STABILIZING PERSISTENT MULTI-AGENT SYSTEM EVOLUTION THROUGH EVIDENCE-GATED REVISION

Anonymous authors
Paper under double-blind review

ABSTRACT

Persistent multi-agent systems should turn experience across recurring tasks into reusable problem-solving procedures that improve over time. Yet every accepted revision is inherited by later tasks and future revisions, so harmful updates can compound alongside useful ones. The central challenge is determining what evidence justifies allowing a revision to shape the system's future. We introduce EMAS, which turns cross-task experience into evidence-gated MAS revisions while keeping the underlying LLM fixed. Its Step graph aligns diagnoses and edits; recurring diagnoses across tasks trigger bounded proposals, and paired validation on separate instances determines retention. After two epochs, validation-selected checkpoints improve task-weighted held-out test accuracy over the initial MAS across four benchmarks by 4.10 and 12.09 percentage points with Kimi K2.6 and Qwen3.6-27B, respectively. Across matched V0–V20 Game24 trajectories, Full EMAS yields the lowest cumulative regression for both backbones. On Qwen, allowing one task to trigger a proposal and removing paired Validation raise this burden to $2.7\times$ and $31.4\times$ the Full-EMAS level, respectively. These results show that evidence-controlled inheritance supports continued improvement while limiting damage along the evolution trajectory. Code is available here.

1 INTRODUCTION

Automated multi-agent system (MAS) design optimizes the executable system surrounding a language model. Prompts assign responsibilities, workflows decompose computation, and topologies route artifacts between agents. Methods that optimize computational graphs or search over agent programs show that this layer substantially affects performance (Zhuge et al., 2024; Hu et al., 2025; Zhang et al., 2025b; Zhou et al., 2026). These methods primarily ask which executable system to construct or select, whereas persistent evolution asks which later evidence should authorize changes to the reusable system inherited by future tasks.

Later executions become especially informative when related tasks recur. Analyses of real-world LLM use reveal recurring task families across large collections of requests (Zheng et al., 2024; Deng et al., 2024; Tamkin et al., 2024). Within a recurring family, a shared prompt-and-topology configuration can serve as a reusable problem-solving procedure. Repeated executions can expose shared defects, missing information flow, or redundant computation, and successful repairs can benefit later instances. This raises the persistent-evolution question of how cross-task experience can revise reusable state while controlling what persists. We study recurring use within supplied task categories with evaluator-provided correctness feedback while keeping the underlying LLM parameters fixed throughout the full evolution trajectory.

In long-running deployments, persistent revision creates a distinct control problem: an accepted change alters the reusable executor inherited by later tasks and becomes the starting point for future revisions, so harmful updates can propagate through the trajectory. Two authorization decisions follow. *Proposal authorization* evaluates whether evidence supports a recurring system defect as the source of failures after considering execution variability, an unusually difficult instance, and instance-specific artifacts. *Retention authorization* asks whether the resulting candidate should replace the current MAS. Even with a recurring diagnosis, the repair may be ineffective or may fix some instances while harming others. These decisions require different evidence because recurrence

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.

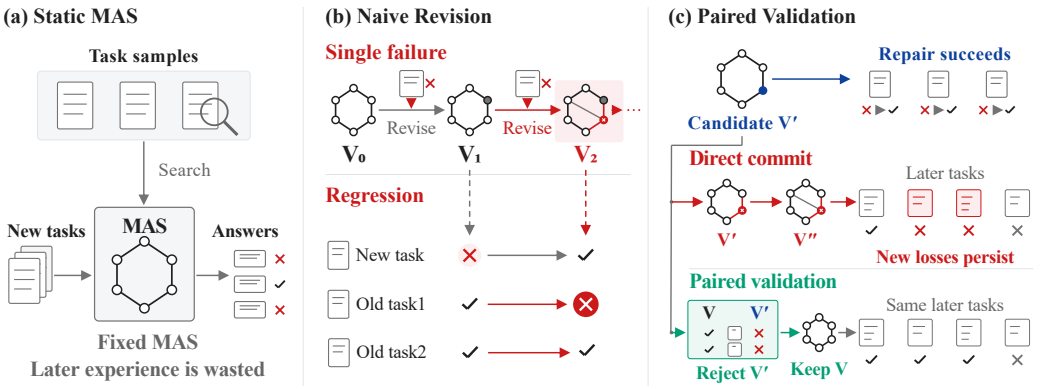

Figure 1: **Motivation for controlling persistent MAS revision. (a)** A static MAS leaves subsequent execution experience unused. **(b)** Revising after an isolated failure can repair a new task while breaking a previously solved task. **(c)** In this illustrative example, a candidate repairs its target cases but introduces other errors. Direct commitment carries these errors into later versions and tasks, while paired Validation detects the regression, rejects the candidate, and retains the current MAS.

can establish that a defect is shared, but not that its proposed repair is beneficial. Figure 1 illustrates the risks of acting on isolated failures and directly committing candidate repairs.

Execution-feedback methods show that agentic programs can be revised and screened (Wang et al., 2024; Ma et al., 2026; He et al., 2026). Sequential inheritance adds a control problem: the observations that motivate a revision and the evidence used to make it the parent of later revisions serve different roles. EMAS makes this distinction explicit through a two-stage protocol for proposal and retention authorization.

We introduce *EMAS*, an Evolving Multi-Agent System that implements this proposal–retention principle for category-specific MAS state. EMAS represents an MAS as a graph of fine-grained *Steps*, providing a shared operation-location space for diagnosis and revision. The Diagnoser maps an execution trace and correctness feedback into localized hypotheses, allowing failures to support the same proposal only when they implicate the same operation at the same system location. Recurring matches across distinct task instances authorize a candidate bounded to one primary operation and its directly affected prompts or connections. Paired Validation then compares the candidate and current MAS on the same category instances that did not trigger the proposal. Only a candidate that improves accuracy, or reduces token use without lowering accuracy, is committed as the system inherited by subsequent tasks.

We evaluate endpoint performance across four benchmarks and analyze regression burden for Kimi and Qwen on Game24. After two evolution epochs, Validation-selected checkpoints improve task-weighted held-out Test accuracy over the Initial MAS by 4.10 and 12.09 percentage points with Kimi K2.6 and Qwen3.6-27B, respectively. Over V0–V20, Full EMAS yields the lowest cumulative Test accuracy regression for both models. On Qwen, it reduces this burden by 62% relative to single-instance proposal authorization and by 97% relative to ungated commitment. On Qwen–MBPP, held-out Test accuracy rises from 55.09% to 81.17% while inference tokens decrease by 90.9%.

Our contributions are threefold.

- We formulate persistent MAS evolution as a sequential inheritance problem that separates proposal from retention authorization.
- EMAS uses recurrence-gated bounded proposals to limit isolated-failure revisions and paired Validation to limit harmful commitments.
- Across evolution trajectories, experiments show accuracy gains, lower regression, and substantial inference-token savings.

2 Related Work

Prior work provides editable programs, experience-driven revision, and persistent-update screening. We organize it by whether evidence supports candidate construction or persistent commitment.

2.1 Editable Representations for Agentic Programs

Automated agent design optimizes prompts, communication structures, workflows, and program code (Zhuge et al., 2024; Zhou et al., 2026; Hu et al., 2025; Zhang et al., 2025b). Related work also studies workflow reuse, joint prompt–topology search, and prompt optimization (Zhang et al., 2025a; Ding et al., 2026; Hong et al., 2026; Bai & Shi, 2026). These methods treat reusable programs as optimization targets. Fine-grained Step graphs localize EMAS cross-task diagnosis and revision.

2.2 Experience-Driven Proposal Formation

Execution experience supports verbal reflection, reusable insights, and structured failure diagnosis (Shinn et al., 2023; Zhao et al., 2024; Zhang et al., 2025c; Yang et al., 2026).

Experience also supports direct program revision. AgentOptimizer revises executable functions from execution histories, and Symbolic Learning updates prompts, tools, and pipeline structure through trajectory-derived feedback (Zhang et al., 2024; Ou et al., 2025). Trace and GEPA use execution feedback to optimize program parameters and prompts, while MASPO jointly optimizes role prompts within a fixed MAS topology (Cheng et al., 2024; Agrawal et al., 2026; Wang et al., 2026b). Semantic Backpropagation aggregates feedback across queries before producing new parameters (Wang et al., 2024). These methods generate revisions from execution feedback. EMAS constructs a candidate bounded to the implicated component only after the same operation-location diagnosis recurs across distinct tasks.

2.3 Persistent Revision and Retention

Persistent systems differ in what an execution changes and what later tasks inherit. TacoMAS adapts capabilities and topology within a test instance but resets scratch memory between instances in its reported implementation (Xu et al., 2026). ANN aggregates multi-task textual gradients into reusable team pools, applies structural and performance screening, and freezes the learned pools and architecture for evaluation (Ma et al., 2026). HERA retains orchestration experience and role prompts, using recurring failures and replay of original trajectories to guide prompt evolution (Li & Ramakrishnan, 2026). MetaGen retains role caches and role/edge scoring priors across instances while constructing collaboration graphs for individual queries (Wang et al., 2026a). EMAS maintains one current Step graph per supplied category, and accepted Versions are inherited and revised further.

Candidate screening is established in AFlow, Semantic Backpropagation, and ANN, which use performance evaluation. Semantic Backpropagation explicitly compares new and current parameters on a validation set before updating (Zhang et al., 2025b; Wang et al., 2024; Ma et al., 2026). Such screening need not separate promotion evidence from proposal evidence.

Recent systems use execution-derived feedback to revise persistent agentic programs and retain improvements across tasks (Ma et al., 2026; Li & Ramakrishnan, 2026; He et al., 2026). EMAS structures this process around localized cross-task evidence. Recurring operation-location diagnoses support bounded candidate construction, while separate category Validation governs whether the resulting revision enters the persistent Version trajectory.

SEAGym provides an evaluation framework for evolving-agent snapshots, transfer, retention on replayed tasks, and cost (Zheng et al., 2026). Because each accepted EMAS Version becomes the next executor and revision parent, retention covers the full inherited trajectory, including its endpoint. Appendix Table 3 summarizes the criteria. Prior methods instantiate different subsets of these evidence roles. EMAS organizes localized recurrence and non-trigger current–candidate comparison into separate proposal and retention authorizations over a sequentially inherited state.

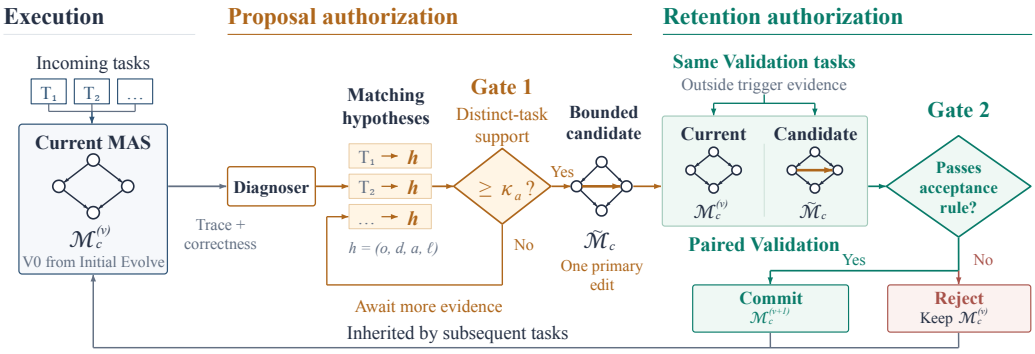

Figure 2: **EMAS evolution through two authorization gates.** Cross-task recurrence authorizes bounded proposals, while paired Validation authorizes retention.

3 METHODOLOGY

3.1 PROBLEM FORMULATION AND EMAS OVERVIEW

EMAS progressively revises the reusable MAS state shared by tasks within each supplied category while keeping the underlying LLM parameters fixed. An accepted revision is inherited by later tasks and becomes the starting point for further revisions. The evolution lifecycle therefore contains two authorization gates. *Proposal authorization* (Gate 1) asks whether cross-task evidence warrants constructing a candidate, whereas *retention authorization* (Gate 2) asks whether that candidate should enter persistent state. EMAS assigns different evidence to these decisions. Recurring structured hypotheses from Evolve traces authorize a proposal, while paired comparison with the current Version on category Validation instances outside the proposal's trigger evidence authorizes retention.

To make proposal evidence comparable and candidate edits localized, EMAS represents each MAS as a graph of atomic LLM-call Steps connected by artifact-flow Edges. It constructs and evolves a separate graph for each supplied task category. We assume evaluator-provided correctness feedback for Evolve executions and a fixed, labeled Validation set per category. The Diagnoser receives execution traces and a binary correctness signal, but reference answers are not directly supplied to the Diagnoser or the Joint Evolver.

For each category c, the current MAS at version v is $\mathcal{M}_c^{(v)} = (G_c^{(v)}, P_c^{(v)})$, where $G_c^{(v)}$ specifies the graph's LLM-call nodes and edges, and $P_c^{(v)}$ specifies the prompt for each node. A version therefore denotes a particular MAS state. Only an accepted revision advances v, whereas a rejected candidate leaves the current Version unchanged. EMAS supports two evolution objectives, namely repairing accuracy failures and reducing execution token cost without decreasing aggregate Validation accuracy. Table 5 and Appendix B detail the data construction and evolution settings.

Initial Evolve constructs the initial MAS $\mathcal{M}_c^{(0)}$. Figure 2 illustrates the subsequent Online Evolve loop governed by these two gates.

3.2 FINE-GRAINED STEP GRAPH AS EVIDENCE AND EDIT SPACE

An *Agent* is a role-bearing component responsible for multiple internal tasks. EMAS decomposes these responsibilities into fine-grained *Steps*, each using one LLM call for a semantically atomic task. An Agent may therefore comprise multiple Steps that expose its internal computation and communication as a set of fine-grained operations.

The graph $G_c^{(v)} = (V_c^{(v)}, E_c^{(v)})$ contains LLM-call Steps $s \in V_c^{(v)}$ and directed artifact-flow Edges $E_c^{(v)}$. A *Phase* groups Steps into a workflow stage with shared instructions. The prompt state $P_c^{(v)}$ comprises system-, category-, phase-, and step-level instructions, enabling fine-grained control over execution. In topological order, Step s receives the original task, its prompt, and the artifacts of its direct predecessors. The resulting MAS is therefore represented as a directed acyclic graph with a

single sink Step whose output provides the final answer. Steps specify computation, Edges specify information flow, and prompts specify instructions. These Step, Edge, and prompt addresses are shared by diagnosis and revision. Cross-instance evidence is aligned at canonical locations, and an authorized candidate is bounded around the location it implicates.

For benchmark b, the maintained state is $\mathcal{M}_b^{(\mathbf{v})} = \{\mathcal{M}_c^{(v_c)} \mid c \in \mathcal{C}_b\}$, where $\mathcal{C}_b$ denotes its categories and $\mathbf{v} = (v_c)_{c \in \mathcal{C}_b}$ records their versions. Accordingly, EMAS evolves one MAS for each task category within a benchmark.

3.3 Initializing the MAS State

EMAS begins from an LLM-designed initial state. Initial Evolve deterministically selects a small set of representative inputs per category, which the category Designer jointly processes into Steps, Edges, and fine-grained prompts. Structural validation and repair enforce valid graph references and topology together with complete prompt coverage, and the resulting design is stored as $\mathcal{M}_c^{(0)}$. Prior work similarly uses LLMs to design agentic systems and workflows (Hu et al., 2025; Zhang et al., 2025b). The resulting state may expose recurring limitations during Online Evolve.

3.4 Structured Operation-Location Diagnosis

For sample i, Online Evolve executes the current $\mathcal{M}_c^{(v)}$ and records its trace. The Diagnoser uses this trace and the binary correctness signal to produce an observation $\tau_i^{(v)}$ containing one or more localized revision hypotheses. Its role is semantic evidence extraction; authorization is a separate function. Proposal and retention are decided downstream. By preserving canonical Step, Edge, Phase, and prompt locations, each system-level outcome becomes an operationally testable hypothesis that can be compared across tasks. Incorrect outputs produce accuracy-oriented hypotheses, whereas correct outputs produce cost-oriented hypotheses. Accuracy diagnosis searches for a recoverable defect that affects correctness. Cost diagnosis searches for avoidable computation, information transfer, or instructions while preserving the correct outcome.

A trace can yield multiple records, each canonicalized as $h = (o, d, a, \ell)$. Here, $o \in \{\text{ACCURACY}, \text{COST}\}$ denotes the objective, d a predefined defect class, a the associated operation, and ℓ its canonical location. Records match only if all four fields are equal. Locations use Step IDs, Edge endpoints, or prompt scope and ID. For ADD-NODE, ℓ combines the Phase ID, optional insertion anchors, and a lower-snake-case label for the missing responsibility.

EMAS can modify three components of a MAS, namely Steps, Edges, and prompts. For Steps, ADD-NODE addresses a missing operation, REMOVE-NODE addresses a redundant or harmful operation, and SPLIT-NODE addresses a non-atomic operation. For Edges, ADD-EDGE supplies missing information to a Step, whereas REMOVE-EDGE eliminates redundant or harmful artifact flow. For prompts, PROMPT-ONLY revises the relevant prompt to provide better instructions.

For example, if a Verifier Step does not receive a Solver artifact, the Diagnoser records a missing-information, ADD-EDGE hypothesis at the canonical Solver-to-Verifier location. Such a hypothesis can support candidate generation when it recurs across enough distinct Evolve samples.

3.5 Proposal Authorization through Recurrence-Gated Bounded Revision

Proposal authorization uses recurrence to distinguish a shared pattern from an isolated diagnosis. For each hypothesis h, EMAS maintains an evidence buffer containing the distinct Evolve samples whose traces support it. Candidate construction is authorized only when this support reaches the operation-specific threshold κ_a.

Operation-specific thresholds encode a design choice to require greater support for removal operations. Removing a Step or Edge can eliminate computation or information flow required by some samples. The threshold settings appear in Appendix B.1.

Once authorized, the Joint Evolver uses the current MAS and supporting diagnoses to generate a complete candidate MAS, which undergoes structural validation before paired Validation. Authorization does not permit an unrestricted rewrite. Each candidate is restricted to one primary change

at ℓ and a narrowly scoped closure of directly affected Edges and prompts. This keeps the candidate aligned with the operation and location supported by the recurring diagnosis. ADD-NODE adds exactly one Step and may add only Edges incident to the new Step and input-aligned prompts, whereas REMOVE-NODE removes exactly one Step and may update only incident or bypass Edges and input-aligned prompts. SPLIT-NODE preserves the target Step, adds same-Phase Steps, and may update only split-group Edges and prompts together with direct-successor prompts. ADD-EDGE and REMOVE-EDGE change exactly one Edge and may revise only the target Step's prompt. PROMPT-ONLY revises exactly the slot encoded by ℓ, leaving topology and all other prompts unchanged.

3.6 Retention Authorization through Paired Validation and Persistent Commit

A recurrence-authorized, structurally valid candidate may still fail to improve the system. Retention authorization therefore uses category instances outside the evidence that triggered the proposal. The candidate and current MAS are evaluated on the same fixed Validation set $\mathcal{D}_c^{\mathrm{val}}$, whose instances do not contribute to proposal construction. Both systems use the same execution and evaluation configuration to ensure a fair comparison. For a system M, define its aggregate correct count as $C(M) = \sum_{i \in \mathcal{D}_c^{\mathrm{val}}} r_i(M)$ and its total execution token cost as $T(M) = \sum_{i \in \mathcal{D}_c^{\mathrm{val}}} \mathrm{tokens}_i(M)$. EMAS uses the following objective-specific acceptance rules.

$$\begin{aligned} \mathrm{Accept}_{\mathrm{acc}}(\widetilde{\mathcal{M}}_c) &\iff C(\widetilde{\mathcal{M}}_c) > C(\mathcal{M}_c^{(v)}), \\ \mathrm{Accept}_{\mathrm{cost}}(\widetilde{\mathcal{M}}_c) &\iff C(\widetilde{\mathcal{M}}_c) \geq C(\mathcal{M}_c^{(v)}) \ \wedge \ T(\widetilde{\mathcal{M}}_c) < T(\mathcal{M}_c^{(v)}). \end{aligned} \tag{1}$$

An accuracy-oriented candidate is accepted only if it strictly increases the aggregate correct count. A cost-oriented candidate is accepted only if it maintains or increases the aggregate correct count while strictly reducing the total execution token cost. These are empirical promotion rules. The accuracy rule accepts any positive correct-count change, while the cost rule requires observed Validation retention together with lower tokens. The paired gate supplies candidate-specific retention evidence; final generalization estimation lies outside its scope. Because the same Validation set is reused across revisions, held-out Test evaluation, accepted-transition transfer, and complete-trajectory ablations separately assess the retained states. Appendix C.3.4 reports how observed correct-count margins align with held-out transfer.

On rejection, EMAS retains the current $\mathcal{M}_c^{(v)}$ and records the rejected candidate. On acceptance, EMAS promotes the candidate to $\mathcal{M}_c^{(v+1)}$, which becomes the current MAS for subsequent category tasks and the starting point for further revisions. In either outcome, EMAS consumes every buffered diagnosis record associated with a sample that supported the processed trigger. Following acceptance, EMAS re-executes the remaining active samples under $\mathcal{M}_c^{(v+1)}$ to obtain up-to-date diagnoses. This replay refreshes the evidence for later proposals so that diagnoses refer to the updated MAS state. To separate online state transition from offline reporting, after a fixed evaluation horizon we distinguish a checkpoint selected from recorded Validation outcomes, the last committed Version, and a retrospective Test-best diagnostic. These views do not alter the evolution trajectory. The last accepted Version remains the current MAS throughout Online Evolve.

4 Evaluation

We ask whether Online Evolve improves beyond model-informed initialization, why endpoint checkpoints are insufficient for assessing persistent evolution, and how recurrence and paired Validation affect proposal and retention. Experiments use two frozen backbones, Qwen3.6-27B (Qwen Team, 2026) and Kimi K2.6 (Moonshot AI, 2026), on MATH (Hendrycks et al., 2021), MBPP (Austin et al., 2021), PlanBench (Valmeekam et al., 2023), and Game24 (Yao et al., 2023). For each supplied category, Evolve provides diagnosed executions, Validation governs promotion and checkpoint selection, and Test remains held out during evolution. We compare against the single-prompt Genesis reference, the Initial MAS, and externally designed systems from GPTSwarm (Zhuge et al., 2024), AFlow (Zhang et al., 2025b), and ADAS (Hu et al., 2025). Appendix B specifies data construction and reporting.

E2 and E15 denote horizons after two and fifteen evolution epochs. A Version is a category-specific MAS state, with V0 as the Initial MAS and its index advancing only after a committed revision. Ver-

Table 1: **Held-out Test accuracy (%).** Overall is task-weighted. Initial MAS and EMAS–Selected@E2 share initialization; Selected uses Validation. Genesis is the pre-construction reference. Search and selection opportunities are not matched across methods. Within each row, bold and underline mark the highest and second-highest reported accuracies; equal displayed values share a rank. EMAS–Oracle@E2 is the retrospective Test-best state in the E2 archive and is excluded from ranking.

Benchmark	Genesis	GPTSwarm	AFlow	ADAS	Initial MAS	EMAS	
						Selected@E2	Oracle@E2
Kimi K2.6							
Math	78.37	84.97	89.80	81.00	**97.43**	<u>97.20</u>	98.83
MBPP	68.91	88.08	89.81	82.73	**95.85**	<u>94.30</u>	95.85
PlanBench	14.40	36.39	60.03	3.49	<u>83.51</u>	**85.69**	88.13
Game24	15.69	81.99	<u>95.10</u>	**96.20**	68.38	<u>95.10</u>	95.34
Overall	54.92	74.81	84.43	67.39	<u>90.11</u>	**94.21**	95.79
Qwen3.6-27B							
Math	38.33	75.83	**97.60**	90.57	<u>97.50</u>	95.47	97.70
MBPP	58.38	82.21	**93.09**	<u>86.70</u>	55.09	81.17	89.12
PlanBench	14.14	14.40	<u>51.66</u>	33.07	35.08	**65.62**	68.67
Game24	14.95	16.54	15.44	<u>67.03</u>	51.72	**79.90**	79.90
Overall	31.98	55.06	<u>75.53</u>	74.81	73.42	**85.51**	88.18

sion counts therefore need not align across trajectories at the same epoch horizon, and V0–V20 denotes 20 committed revisions. Selected@E2 is chosen by recorded Validation outcomes and is used for headline comparisons. Terminal@E2 is the last committed Version at E2, whereas Oracle@H is the retrospective Test-best Version through horizon H and is diagnostic only. Appendix B.3 gives the exact rules.

4.1 Endpoint Effectiveness Beyond Initialization

Selected@E2 raises Overall Test accuracy over Initial MAS by 4.10 points for Kimi (90.11 to 94.21) and 12.09 for Qwen (73.42 to 85.51), with paired Test-task bootstrap 95% confidence intervals excluding zero for both gains (Appendix Table 7). Initial MAS holds model-informed system construction constant, so these results measure the added outcome of Online Evolve's candidate generation, promotion, and checkpoint selection. Separate interventions isolate the two authorization controls.

Genesis is a secondary reference before system construction. Since initialization uses the backbone's task and workflow-design knowledge to construct an executable procedure, Genesis-to-Selected combines system construction with later revision.

The external comparisons place the resulting executor alongside independently designed workflows using the same backbone and Test tasks. Benchmark-dependent rankings make the task mixture part of the aggregate result because EMAS can lead overall while alternatives lead on individual tasks and revisions lose to Initial in some settings. The evidence therefore supports overall competitiveness under the reported protocols, whose search and selection opportunities differ across methods.

Oracle@E2 separates state discovery from checkpoint identification by reporting stronger states reached within the same archive but not selected by Validation. Because V0 remains an eligible non-regressive state, Selected below Initial reflects the checkpoint choice.

These endpoint comparisons leave the temporal scope of improvement open. We use Game24 as a controlled trajectory setting: its single recurring task schema, exact verifier, balanced Validation pools, and headroom under both backbones support matched V0–V20 policy comparisons without category-mixture changes. Cross-benchmark endpoints and accepted-transition statistics provide complementary breadth, while the Game24 interventions probe the two authorization mechanisms under fixed conditions.

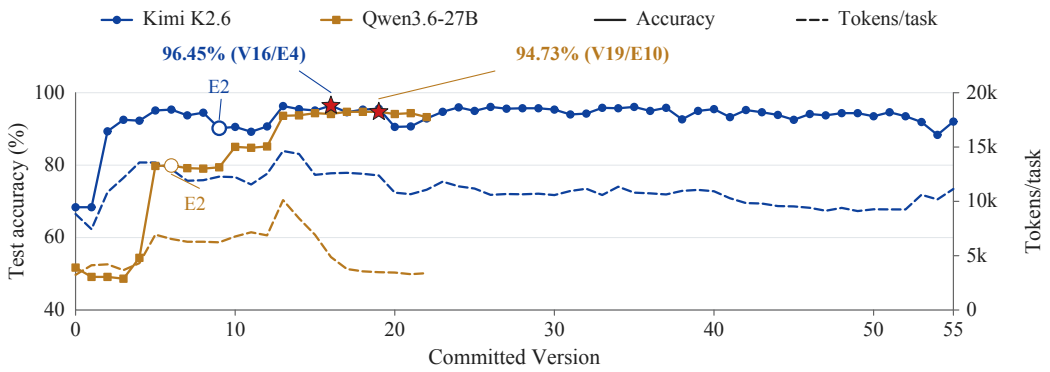

Figure 3: **Game24 evolution through E15.** Colors identify backbones; solid lines show Test accuracy (left axis), and same-color dashed lines show inference tokens per Test task (right axis). Both trajectories end at E15; Version counts do not align epochs across models. Open circles mark the E2 terminal states (Kimi V9; Qwen V6). Stars mark retrospective Oracle@E15 states (Kimi V16/E4; Qwen V19/E10), following Appendix B.3. Figure 4 compares Kimi K2.6 and Qwen3.6-27B over the common V0–V20 window under Full EMAS, $\kappa = 1$, and No Gate.

4.2 Persistent Evolution Beyond Endpoint Selection

Endpoint quality alone does not characterize persistent evolution because the best archived state, the last committed state, and the path between them can diverge. Retaining an Oracle checkpoint preserves a historical peak, whereas the ongoing executor inherits the consequences of each accepted update. Even without archive selection, the Terminal@E2 executor remains above the Initial MAS overall for both backbones, reaching 91.55 versus 90.11 for Kimi and 85.36 versus 73.42 for Qwen (Appendix Table 6). Later regressions therefore matter even when the archive remains strong, and neither an improving archive nor falling token use establishes reliable retention by itself.

The E2 comparison fixes a common reporting horizon but does not establish that revision has exhausted its useful opportunities. The continuations in Figure 3 depend on the executor inherited at E2, with Qwen retaining more accuracy headroom while Kimi has reached a stronger early state. Continued revision can therefore add value at different rates, and a small later gain need not mean that the procedure has stopped changing.

The token trajectories extend this assessment to execution cost, showing that later states can preserve much of an earlier accuracy gain using fewer tokens, so useful revision need not set a new accuracy maximum. Lower-token states can sacrifice accuracy, creating an evolving deployment trade-off that requires separate evidence of reliable retention.

We next use ablations to test how recurrence and paired Validation shape these trajectories.

4.3 Jointly Controlling Proposal and Retention

Persistent revision depends on two sequential decisions: when observed failures justify a candidate revision, and whether that candidate should enter the state inherited by later tasks. Recurrence aggregates cross-task evidence for proposal authorization; paired Validation evaluates candidates before retention. We isolate these roles with complementary interventions: κ=1 allows an isolated diagnosis to trigger a proposal while retaining paired Validation, and No Gate preserves recurrence while removing pre-commit Validation.

Relaxing recurrence changes regression magnitude more strongly than frequency. Relative to κ=1, Full EMAS reduces cumulative regression by 25.7% for Kimi and 62.4% for Qwen, while all four trajectories contain seven or eight regressive steps (Table 2). Recurrence therefore improves proposal quality by limiting high-impact revisions triggered by isolated failures. Kimi further exposes the exploration–stability trade-off: κ=1 reaches a higher V20 than Full (96.69% versus 90.56%) while accumulating more regression. A strong state reached at a lower committed-Version index can come with larger setbacks along the inherited path.

No Gate isolates the retention decision. With recurrence still active, cumulative regression rises to 129.04 pp for Kimi and 157.97 pp for Qwen; Full EMAS reduces these burdens by 87.9% and

Setting	Regressive steps	Cumulative regression (pp)
Kimi K2.6–Game24		
Full EMAS	8/20	**15.56**
$\kappa = 1$	7/20	20.96
No Gate	12/20	129.04
Qwen3.6-27B–Game24		
Full EMAS	7/20	**5.02**
$\kappa = 1$	8/20	13.36
No Gate	11/20	157.97

Table 2: **Regression burden through V20.** On Game24, a regressive step decreases Test accuracy A_v; cumulative regression is $\sum_{v=1}^{20} \max(0, A_{v-1} - A_v)$ in percentage points within each backbone's V0–V20 trajectory.

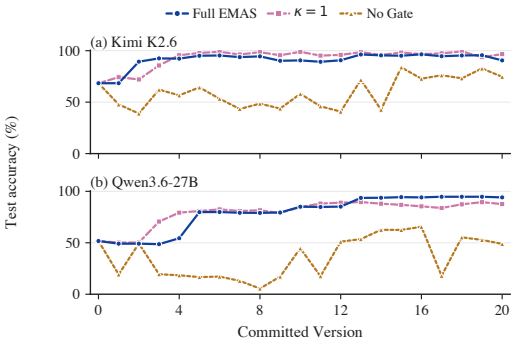

Figure 4: **Game24 accuracy through V20.** Panels show Kimi K2.6 and Qwen3.6-27B under Full EMAS, $\kappa = 1$, and No Gate. Within each backbone, the three settings share V0 and use the common V0–V20 window.

96.8%, respectively. This increase localizes the remaining vulnerability to commitment: useful diagnoses can still produce harmful candidates, and each committed candidate becomes the parent of subsequent revisions. Kimi's No Gate trajectory ends above Qwen's (74.63% versus 48.90%), illustrating how a stronger backbone can reconstruct useful strategies from sparse failure signals. Its 129.04 pp regression burden also shows that recovery capacity cannot safeguard the persistent state without paired Validation.

Backbone dynamics explain why the absolute Full EMAS burden is larger for Kimi. Kimi crosses 90% accuracy at V3, while Qwen reaches the same region at V13. Kimi consequently spends most of V0–V20 maintaining a strong incumbent with limited upward headroom, where even modest disruptions contribute to cumulative regression; Qwen spends more of the window ascending. Together, these results assign complementary roles to the two controls: recurrence strengthens the evidential basis of candidate revisions, and paired Validation protects accumulated progress when those candidates enter the persistent state. Their joint optimization enables continued exploration along a reliable evolutionary path. Appendix Table 9 and Figure 7 further show that restricting revision to topology or prompt alone cannot match joint adaptation.

5 Conclusion and Limitations

Persistent MAS evolution accumulates a reusable problem-solving procedure through revisions inherited by later tasks. EMAS controls this process with two authorization decisions while keeping the underlying LLM fixed. Recurring operation-location diagnoses across tasks authorize bounded topology-and-prompt revisions, and paired Current/Candidate Validation on non-trigger instances determines whether each candidate enters persistent state.

Across four benchmarks, Validation-selected checkpoints after two evolution epochs improve task-weighted Test accuracy over the Initial MAS by 4.10 percentage points with Kimi K2.6 and 12.09 points with Qwen3.6-27B. The long-horizon Game24 trajectories show that reaching a strong archived state and retaining its gains in the ongoing executor are distinct outcomes. Over V0–V20, Full EMAS yields the lowest cumulative Test accuracy regression for both backbones. On Qwen, it reduces this burden by 62.4% relative to κ=1 and by 96.8% relative to No Gate. Recurrence strengthens the evidence behind proposed revisions, and paired Validation protects accumulated progress after commitment.

Our study assumes supplied task categories, evaluator-provided correctness feedback, and a fixed Validation set reused across sequential decisions. Future work should examine automatically discovered task families, refreshed retention evidence over longer horizons, and cheaper screening. EMAS establishes a broader principle for self-evolving agentic systems: evolution quality depends on both what the system discovers and what it allows future tasks to inherit.

AI USE STATEMENT

Generative AI tools were used to assist with editing software code and LaTeX, creating and revising scientific figures and tables, organizing the manuscript, improving readability, and checking formatting and references. All AI-assisted code, text, and artifacts were reviewed by the authors. Numerical results were checked against the underlying experimental records, generated tables, and audit files. The authors retained final control over the scientific claims and take responsibility for the complete content of this work, including all AI-assisted material.

REPRODUCIBILITY STATEMENT

The method and its state-transition procedure are specified in Section 3; the evaluation protocol, main comparisons, and ablations are described in Section 4 and the appendix. The supplementary material includes the recorded result bundle, provenance manifests and checksums, and the scripts used to regenerate and validate the paper-facing tables and figures. These checks reject incomplete records, inconsistent aggregates, unregistered artifacts, and failed numerical invariants.

REFERENCES

Lakshya A. Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J. Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=RQm2KQTM5r.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc V. Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021. URL https://arxiv.org/abs/2108.07732.

Juyang Bai and Laixi Shi. MAS-PromptBench: When does prompt optimization improve multi-agent LLM systems? *arXiv preprint arXiv:2606.23664*, 2026. URL https://arxiv.org/abs/2606.23664.

Ching-An Cheng, Allen Nie, and Adith Swaminathan. Trace is the next AutoDiff: Generative optimization with rich feedback, execution traces, and LLMs. In *Advances in Neural Information Processing Systems*, volume 37, pp. 71596–71642, 2024. doi: 10.52202/079017-2287. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/83ba7056bce2c3c3c27e17397cf3e1f0-Abstract-Conference.html.

Yuntian Deng, Wenting Zhao, Jack Hessel, Xiang Ren, Claire Cardie, and Yejin Choi. WildVis: Open source visualizer for million-scale chat logs in the wild. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 497–506. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-demo.50. URL https://aclanthology.org/2024.emnlp-demo.50/.

Yi Ding, Zijie Xuan, Haowei Zhou, Zhenyu Ju, Xiaoxiao Dong, Jingwen Zhang, Xingyu Zhu, Leixin Sun, and Haochi Zhang. TCP-MCP: Landscape-guided co-evolution of prompts and communication topologies for multi-agent systems. *arXiv preprint arXiv:2605.27850*, 2026. URL https://arxiv.org/abs/2605.27850.

Shan He, Runze Wang, Zhuoyun Du, Huiyu Bai, Zouying Cao, Yu Cheng, and Bo Zheng. Learning to evolve: A self-improving framework for multi-agent systems via textual parameter graph optimization. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 30719–30732. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.1534. URL https://aclanthology.org/2026.findings-acl.1534/.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

Zhi Hong, Qian Zhang, Jiahang Sun, Zhiwei Shang, Mingze Kong, Xiangyi Wang, Yao Shu, and Zhongxiang Dai. MASPOB: Bandit-based prompt optimization for multi-agent systems with graph neural networks. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/65791.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=t9U3LW7JVX.

Sha Li and Naren Ramakrishnan. Experience as a compass: Multi-agent RAG with evolving orchestration and agent prompts. *arXiv preprint arXiv:2604.00901*, 2026. URL https://arxiv.org/abs/2604.00901.

Xiaowen Ma, Yunpu Ma, Chenyang Lin, Sikuan Yan, Jinhe Bi, Zixuan Cao, Yijun Tian, Volker Tresp, and Hinrich Schuetze. Self-evolving multi-agent systems via textual backpropagation. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 9918–9951. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.483. URL https://aclanthology.org/2026.findings-acl.483/.

Moonshot AI. Kimi K2.6: Advancing open-source coding, April 2026. URL https://www.kimi.com/blog/kimi-k2-6.

Yixin Ou, Wangchunshu Zhou, Shengwei Ding, Long Li, Jialong Wu, Tiannan Wang, Jiamin Chen, Shuai Wang, Xiaohua Xu, Ningyu Zhang, Huajun Chen, and Yuchen Eleanor Jiang. Symbolic learning enables self-evolving agents. *AI Open*, 6:314–322, 2025. doi: 10.1016/j.aiopen.2025.11.004. URL https://doi.org/10.1016/j.aiopen.2025.11.004.

Qwen Team. Qwen3.6-27B: Flagship-level coding in a 27B dense model, April 2026. URL https://qwen.ai/blog?id=qwen3.6-27b.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652, 2023. doi: 10.52202/075280-0377. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Alex Tamkin, Miles McCain, Kunal Handa, Esin Durmus, Liane Lovitt, Ankur Rathi, Saffron Huang, Alfred Mountfield, Jerry Hong, Stuart Ritchie, Michael Stern, Brian Clarke, Landon Goldberg, Theodore R. Sumers, Jared Mueller, William McEachen, Wes Mitchell, Shan Carter, Jack Clark, Jared Kaplan, and Deep Ganguli. Clio: Privacy-preserving insights into real-world AI use. *arXiv preprint arXiv:2412.13678*, 2024. URL https://arxiv.org/abs/2412.13678.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. PlanBench: An extensible benchmark for evaluating large language models on planning and reasoning about change. In *Advances in Neural Information Processing Systems*, volume 36, pp. 38975–38987, 2023. doi: 10.52202/075280-1693. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/7a92bcdede88c7afd108072faf5485c8-Abstract.html.

Wenyi Wang, Hisham A. Alyahya, Dylan R. Ashley, Oleg Serikov, Dmitrii Khizbullin, Francesco Faccio, and Jürgen Schmidhuber. How to correctly do semantic backpropagation on language-based agentic systems. *arXiv preprint arXiv:2412.03624*, 2024. URL https://arxiv.org/abs/2412.03624.

Yimeng Wang, Jiaxing Zhao, Hongbin Xie, Hexing Ma, Yuzhen Lei, Shuangxue Liu, Xuan Song, Zichen Zhang, and Haoran Zhang. MetaGen: Self-evolving roles and topologies for multi-agent LLM reasoning. *arXiv preprint arXiv:2601.19290*, 2026a. URL https://arxiv.org/abs/2601.19290.

Zhexuan Wang, Xuebo Liu, Li Wang, Zifei Shan, Yutong Wang, Zhenxi Song, and Min Zhang. MASPO: Joint prompt optimization for LLM-based multi-agent systems. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026b. URL https://icml.cc/virtual/2026/poster/62219.

Chen Xu, Yicheng Hu, Ruizi Wang, Xinyu Lin, Wenjie Wang, Dongrui Liu, and Fuli Feng. TacoMAS: Test-time co-evolution of topology and capability in LLM-based multi-agent systems. *arXiv preprint arXiv:2605.09539*, 2026. URL https://arxiv.org/abs/2605.09539.

Shu-Xun Yang, Cunxiang Wang, Haoke Zhang, Wenbo Yu, Lindong Wu, Jiayi Gui, Dayong Yang, Yukuo Cen, Zhuoer Feng, Bosi Wen, Yidong Wang, Lucen Zhong, Jiamin Ren, Linfeng Zhang, and Jie Tang. TraceSIR: A multi-agent framework for structured analysis and reporting of agentic execution traces. *arXiv preprint arXiv:2603.00623*, 2026. URL https://arxiv.org/abs/2603.00623.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 11809–11822, 2023. doi: 10.52202/075280-0517. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/271db9922b8d1f4dd7aaef84ed5ac703-Abstract.html.

Guibin Zhang, Kaijie Chen, Guancheng Wan, Heng Chang, Hong Cheng, Kun Wang, Shuyue Hu, and Lei Bai. EvoFlow: Evolving diverse agentic workflows on the fly. *arXiv preprint arXiv:2502.07373*, 2025a. URL https://arxiv.org/abs/2502.07373.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xiong-Hui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating agentic workflow generation. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=z5uVAKwmjf.

Shaokun Zhang, Jieyu Zhang, Jiale Liu, Linxin Song, Chi Wang, Ranjay Krishna, and Qingyun Wu. Offline training of language model agents with functions as learnable weights. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 60315–60335. PMLR, 2024. URL https://proceedings.mlr.press/v235/zhang24cd.html.

Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, and Qingyun Wu. Which agent causes task failures and when? on automated failure attribution of LLM multi-agent systems. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 76583–76599. PMLR, 2025c. URL https://proceedings.mlr.press/v267/zhang25cq.html.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024. doi: 10.1609/aaai.v38i17.29936. URL https://ojs.aaai.org/index.php/AAAI/article/view/29936.

Congjie Zheng, Chuanyi Xue, Bin Liang, Jun Yang, and Changshui Zhang. SEAGym: An evaluation environment for self-evolving LLM agents. *arXiv preprint arXiv:2606.17546*, 2026. URL https://arxiv.org/abs/2606.17546.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric Xing, Joseph E. Gonzalez, Ion Stoica, and Hao Zhang. LMSYS-Chat-1M: A large-scale real-world LLM conversation dataset. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=BOfDKxfwt0.

Han Zhou, Xingchen Wan, Ruoxi Sun, Hamid Palangi, Shariq Iqbal, Ivan Vulić, Anna Korhonen, and Sercan Ö. Arık. Multi-agent design: Optimizing agents with better prompts and topologies. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=I05H9RUzHB.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. GPTSwarm: Language agents as optimizable graphs. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62743–62767. PMLR, 2024. URL https://proceedings.mlr.press/v235/zhuge24a.html.

A Reader Guide and Mechanism Positioning

This appendix separates four evidential roles. The remainder of this section positions EMAS at the mechanism level; Appendix B specifies the method and evaluation contract; Appendix C organizes evidence around the main claims; and Appendix D retains retrospective diagnostics and complete records. This hierarchy distinguishes direct intervention evidence from conditional transfer summaries and Test-selected archive diagnostics.

A.1 Mechanism-Level Comparison with Related Work

Table 3 separates mechanisms that are often conflated under broad labels such as feedback, validation, or persistence. Several prior methods instantiate individual components of the EMAS pipeline. In particular, AFlow and Semantic Backpropagation compare proposed and prior systems, while TPGO clusters execution-derived errors, revises a persistent graph, and analyzes its optimization trajectory. Under the table's stricter criteria, however, textual-gradient clustering does not constitute distinct-instance recurrence unless the protocol enforces support from different task instances, and candidate screening does not constitute non-trigger promotion evidence when it reuses the instances that produced the proposal. EMAS combines the two authorization decisions: distinct-instance recurrence governs candidate construction, and a separate category Validation set governs commitment to the persistent state.

Table 3: **Mechanism-level comparison of experience-driven agentic-system revision.** A checkmark indicates that the cited method explicitly implements the criterion defined below; a cross indicates that the criterion is not part of its reported update protocol. Individual ingredients are not claimed to be unique to EMAS.

	Representation		Proposal authorization		Retention authorization		State	Evaluation
Method	Exec. graph	Loc. diagnosis	Distinct recurrence	Bounded edit	Current–candidate	Non-trigger evidence	Sequential commit	Retention path
ADAS (Hu et al., 2025)	×	×	×	×	×	×	×	×
AFlow (Zhang et al., 2025b)	✓	×	×	✓	✓	×	×	×
Semantic BP (Wang et al., 2024)	✓	×	×	×	✓	×	✓	×
ANN (Ma et al., 2026)	✓	✓	×	×	✓	×	×	×
HERA (Li & Ramakrishnan, 2026)	✓	✓	×	✓	×	×	✓	×
TPGO (He et al., 2026)	✓	✓	×	×	✓	×	✓	✓
EMAS (ours)	✓	✓	✓	✓	✓	✓	✓	✓

Criteria. *Exec. graph*: an explicit node–edge execution representation with independently editable components. *Loc. diagnosis*: execution feedback associates an edit direction with a target program component. *Distinct recurrence*: matching evidence from distinct task instances is required before candidate construction. *Bounded edit*: a candidate is restricted to one primary target and limited dependent changes. *Current–candidate*: the proposal is screened against its incumbent. *Non-trigger evidence*: promotion is decided on instances not used to produce the proposal evidence. *Sequential commit*: an accepted state becomes both the next executor and the base for subsequent revision. *Retention path*: evaluation analyzes performance across sequentially retained states rather than only a best or final search output.

B METHOD AND EVALUATION CONTRACT

B.1 PROPOSAL AUTHORIZATION SETTINGS

The operation-specific thresholds in Table 4 require support from distinct samples before a hypothesis can trigger an edit. This support concerns the hypothesis that motivates the edit. Candidate execution provides separate evidence about the program produced in response to that hypothesis.

Table 4: **Operation-specific support thresholds in Full EMAS.** The threshold κ_a is the number of distinct samples that must support the same revision hypothesis before permitted operation a is triggered. The $\kappa = 1$ ablation sets all thresholds to one.

Priority	Permitted operation	Support threshold (κ_a)
1	ADD-NODE	
2	SPLIT-NODE	4
3	ADD-EDGE	
4	PROMPT-ONLY	6
5	REMOVE-EDGE	10
6	REMOVE-NODE	

B.2 DATA AND EXECUTION CONTRACT

Table 5 summarizes the data used for evolution and evaluation. Each benchmark is split into Train and Test. Test is fixed, shared by both backbones, and held out during evolution.

Table 5: **Data partitions for evolution and evaluation.** Each benchmark is split into 80% Train and 20% Test data. For each model and category, the Initial MAS (V0) is run on Train, which is then divided into Evolve and Validation. Validation includes both V0-correct and V0-incorrect tasks when possible, but can be imbalanced when one outcome is scarce and the minimum split size must be maintained. Validation cells report the split size followed by the corresponding correct/incorrect composition in parentheses. Test is shared across models and remains held out throughout evolution.

Benchmark	Model	Category	Train			Test
			Total	Evolve	Validation (V0 R/W)	
Math	Kimi K2.6	algebra	950	760	190 (95/95)	237
		count./prob.	379	329	50 (26/24)	95
		geometry	383	307	76 (38/38)	96
		interm. algebra	722	578	144 (72/72)	181
		number theory	432	382	50 (28/22)	108
		prealgebra	697	563	134 (67/67)	174
		precalculus	437	349	88 (44/44)	109
		Overall	**4,000**	**3,268**	**732 (370/362)**	**1,000**
	Qwen3.6-27B	algebra	950	834	116 (58/58)	237
		count./prob.	379	329	50 (27/23)	95
		geometry	383	307	76 (38/38)	96
		interm. algebra	722	578	144 (72/72)	181
		number theory	432	382	50 (31/19)	108
		prealgebra	697	583	114 (57/57)	174
		precalculus	437	349	88 (44/44)	109
		Overall	**4,000**	**3,362**	**638 (327/311)**	**1,000**
MBPP	Kimi K2.6	all	771	717	54 (27/27)	193
		Overall	**771**	**717**	**54 (27/27)**	**193**
	Qwen3.6-27B	all	771	711	60 (30/30)	193
		Overall	**771**	**711**	**60 (30/30)**	**193**
PlanBench	Kimi K2.6	blocks.	480	430	50 (47/3)	120
		blocks. hard	88	70	18 (12/6)	22
		mystery	480	418	62 (31/31)	120
		randomized	480	384	96 (48/48)	120
		Overall	**1,528**	**1,302**	**226 (138/88)**	**382**
	Qwen3.6-27B	blocks.	480	384	96 (48/48)	120
		blocks. hard	88	70	18 (9/9)	22
		mystery	480	388	92 (46/46)	120
		randomized	480	384	96 (48/48)	120
		Overall	**1,528**	**1,226**	**302 (151/151)**	**382**
Game24	Kimi K2.6	all	1,090	872	218 (109/109)	272
		Overall	**1,090**	**872**	**218 (109/109)**	**272**
	Qwen3.6-27B	all	1,090	872	218 (109/109)	272
		Overall	**1,090**	**872**	**218 (109/109)**	**272**

For each backbone and category, Train is further divided into Evolve and Validation based on V0 outcomes on eligible Train tasks. Initial-design samples are reserved for Evolve. The builder evaluates eligible tasks until its outcome quotas are met or the pool is exhausted; it need not evaluate all of Train. Validation includes both V0-correct and V0-incorrect tasks when possible. For categories large enough, it contains at least 50 tasks. It can therefore be imbalanced when V0 makes very few errors. Strong Initial MAS performance can leave limited repair headroom in individual categories. This regime is most visible for Kimi K2.6 on PlanBench blocks: V0 makes only three errors among the eligible Train tasks, so constructing an exactly outcome-balanced Validation set is infeasible. We retain all three V0-incorrect tasks and fill the Validation set with 47 distinct V0-correct tasks. This design preserves 50 distinct instances and includes every scarce failure. Consequently, this Validation pool provides retention evidence across previously solved tasks and evidence for cost-safe revisions, while providing limited evidence for whether an accuracy revision generalizes across diverse failures. We therefore interpret it primarily as a retention and cost-safety regime with limited observed repair headroom. The remaining Train tasks are used for Evolve.

Evolve provides experience for system revision, Validation is used to accept or reject revisions and to select the primary archived checkpoint after the fixed horizon, and Test remains isolated during evolution. Test is used afterward to evaluate frozen Versions; only the Oracle diagnostics use Test to choose a Version.

Generation settings. The recorded EMAS generation profiles use temperature $= 0$, top-p $= 1$, and request seed 42. Role-specific token limits, timeout and retry settings, V0 component hashes, and exact split IDs are retained in the protocol audit.

Evaluation rules. Train outcome stratification and Online Validation use the Online reward implementation; frozen Test predictions are scored by the recorded offline evaluator and parser. For MATH, Online uses string equivalence whereas Test uses Math-Verify 0.9.0 after final-answer extraction. Thus, Table 5 reports Online V0-correct/incorrect counts; Test-evaluator outcomes are distinct. MBPP executes the benchmark assertions; Game24 checks the extracted expression; PlanBench validates the parsed plan with VAL. Failed and unparseable task executions remain in the accuracy denominator. The audit records the precise adapters and stage-specific execution limits.

B.3 Promotion and Checkpoint Reporting

For each backbone and category, let $\mathcal{V}_c(H)$ contain V0 and every committed Version within horizon H. We report three views of this archive.

Selected@E2. For every committed $v > 0$, $C_c^{\mathrm{Val}}(v)$ and $T_c^{\mathrm{Val}}(v)$ are the candidate's correct count and total tokens in the Validation execution that led to its commit. For V0, we use the earliest event within E2 in which the current V0 enters Validation, including an event whose candidate is subsequently rejected. This rule fixes one V0 observation and bypasses selection among its repeated executions. We select each category's primary E2 checkpoint using its fixed Validation set.

$$v_c^{\mathrm{sel}} = \operatorname*{arg\,max}^{\mathrm{lex}}_{v \in \mathcal{V}_c(E2)} \left(C_c^{\mathrm{Val}}(v), -T_c^{\mathrm{Val}}(v), -v \right), \tag{2}$$

so equal correct counts favor fewer Validation tokens and then the earlier Version. Selected@E2 evaluates v_c^{sel} on Test and task-weights the resulting category states for benchmark and Overall results.

Terminal@E2. Terminal@E2 is the last committed Version in $\mathcal{V}_c(E2)$ and represents the actual system state at the end of the two-epoch evolution horizon.

Oracle@H. Oracle@E2 is the retrospective Test-best state in the same archive. More generally, for a descriptive archive frontier at horizon H, Oracle@H chooses

$$v_c^{\mathrm{oracle}} = \operatorname*{arg\,max}^{\mathrm{lex}}_{v \in \mathcal{V}_c(H)} \left(\overline{A}_c^{\mathrm{Test}}(v), -\overline{T}_c^{\mathrm{Test}}(v), -v \right), \tag{3}$$

where the bars denote means over Test executions.

Reporting scope. Only Selected@E2 is used for headline comparisons; Terminal@E2 and Oracle@E2 are diagnostics. The E15 Game24 summary and the retrospective Test-best columns in the V0–V20 ablation tables below use the explicitly labeled Oracle rule for their respective horizons. The direct gate comparison in Table 10 uses the fixed E2 and V20 states. Test never enters candidate construction, Validation acceptance, or the Selected@E2 rule. The Oracle confidence intervals do not adjust for Test-based selection.

C Evidence for the Main Claims

C.1 Endpoint Effectiveness Beyond Initialization

Genesis uses a generic problem-solving instruction and task-specific output constraints when required. Initial Evolve uses the frozen backbone's task and workflow-design knowledge to generate a standard operating procedure (SOP) and decompose it into an executable MAS. Genesis+SOP adds the same SOP to the Genesis prompt without this decomposition. The Initial MAS additionally provides explicit Steps and Edges on which subsequent diagnosis and editing operate.

The initialization controls in Table 6 distinguish supplying a task procedure from instantiating it as an executable workflow. Their aggregate separation indicates that SOP text alone does not account for the strength of the Initial MAS. The incremental comparison used in Section 4.1 consequently starts after this design contribution has already been realized. Initial-to-Selected measures the added outcome of Online Evolve and checkpoint selection; Genesis-to-Selected measures the complete pipeline.

Table 6: **Complete checkpoint, inference, and search results.** Cells show mean accuracy (%) / tokens per task (k); Overall is task-weighted. Selected, Terminal, and Oracle follow Appendix B.3. Only Selected enters the EMAS ranking. Bold and underline mark first and second rank by accuracy, then tokens. The last column reports recorded search/evolution tokens (M). EMAS rows share the same E1–E2 cost, summed across benchmarks. NR denotes unavailable historical usage; – denotes no search/evolution stage.

	Accuracy (%) ↑ / Tokens per task (k) ↓					Search /
Method	**Overall**	**Math**	**MBPP**	**PlanBench**	**Game24**	**evolve (M)**
Kimi K2.6						
Genesis	54.92 / 0.83	78.37 / 0.61	68.91 / 0.28	14.40 / 2.17	15.69 / 0.12	–
Initial MAS	<u>90.11 / 14.81</u>	**97.43 / 14.19**	**95.85 / 9.53**	<u>83.51 / 23.34</u>	68.38 / 8.84	NR
Genesis+SOP	35.88 / 1.64	39.10 / 1.48	85.66 / 1.35	14.75 / 2.39	18.38 / 1.33	NR
GPTSwarm	74.81 / 18.30	84.97 / 13.78	88.08 / 13.72	36.39 / 30.66	81.99 / 20.82	53.27
AFlow	84.43 / 6.77	89.80 / 8.12	89.81 / 3.69	60.03 / 7.62	<u>95.10 / 2.78</u>	710.67
ADAS	67.39 / 3.61	81.00 / 1.98	82.73 / 4.12	3.49 / 5.26	**96.20 / 6.93**	524.34
EMAS–Selected@E2	**94.21 / 15.02**	<u>97.20 / 11.74</u>	<u>94.30 / 8.25</u>	**85.69 / 28.09**	95.10 / 13.57	2062.82
EMAS–Terminal@E2	91.55 / 14.50	97.33 / 11.72	84.28 / 4.58	81.06 / 28.35	90.20 / 12.28	shared
EMAS–Oracle@E2	95.79 / 17.07	98.83 / 13.84	95.85 / 9.53	88.13 / 32.27	95.34 / 12.97	shared
Qwen3.6-27B						
Genesis	31.98 / 0.30	38.33 / 0.13	58.38 / 0.27	14.14 / 0.90	14.95 / 0.12	–
Initial MAS	73.42 / 7.44	<u>97.50 / 8.22</u>	55.09 / 5.15	35.08 / 9.54	51.72 / 3.24	NR
Genesis+SOP	26.84 / 1.04	28.87 / 0.87	59.93 / 0.94	13.35 / 1.73	14.83 / 0.77	NR
GPTSwarm	55.06 / 10.57	75.83 / 15.40	82.21 / 3.13	14.40 / 8.58	16.54 / 0.91	18.26
AFlow	<u>75.53 / 10.40</u>	**97.60 / 4.63**	**93.09 / 4.52**	<u>51.66 / 33.65</u>	15.44 / 3.16	589.26
ADAS	74.81 / 1.86	90.57 / 1.49	<u>86.70 / 0.73</u>	33.07 / 3.68	<u>67.03 / 1.46</u>	352.01
EMAS–Selected@E2	**85.51 / 5.88**	95.47 / 3.57	81.17 / 0.47	**65.62 / 14.16**	**79.90 / 6.54**	866.08
EMAS–Terminal@E2	85.36 / 5.93	95.37 / 3.61	81.17 / 0.47	65.18 / 14.33	79.90 / 6.54	shared
EMAS–Oracle@E2	88.18 / 8.01	97.70 / 7.02	89.12 / 1.95	68.67 / 14.71	79.90 / 6.54	shared

The baselines and EMAS offer different numbers and types of design and selection opportunities. The external comparison therefore assesses the resulting systems under their reported protocols; it does not isolate an optimizer effect under matched search.

The accuracy and token columns describe different consequences of the selected executor. An accuracy improvement need not require a token increase, and the most accurate state need not minimize inference tokens. Revision-time expenditure is reported separately because it covers the construction and evaluation of state transitions; frozen-system execution falls outside this expenditure.

Paired Initial-to-Selected analysis. Table 7 quantifies the incremental accuracy of Selected@E2 over Initial MAS. Exact Test-ID and category matching yields 1,847 paired tasks per backbone.

Table 7: **Paired Initial-to-Selected accuracy differences at E2.** Accuracy is averaged over Test executions. Δ is Selected minus Initial in percentage points; intervals are pointwise paired-bootstrap 95% CIs. Overall is task-weighted.

Benchmark	N	Initial (%)	Selected (%)	Δ (pp)	95% CI (pp)
Kimi K2.6					
MATH	1000	97.43	97.20	-0.23	[-0.87, +0.43]
MBPP	193	95.85	94.30	-1.55	[-3.28, -0.17]
PlanBench	382	83.51	85.69	+2.18	[+0.00, +4.36]
Game24	272	68.38	95.10	+26.72	[+22.43, +31.00]
Overall	1847	90.11	94.21	+4.10	[+3.21, +4.96]
Qwen3.6-27B					
MATH	1000	97.50	95.47	-2.03	[-2.97, -1.13]
MBPP	193	55.09	81.17	+26.08	[+17.79, +34.02]
PlanBench	382	35.08	65.62	+30.54	[+26.70, +34.38]
Game24	272	51.72	79.90	+28.19	[+22.67, +33.58]
Overall	1847	73.42	85.51	+12.09	[+10.59, +13.59]

Table 8: **Supplementary majority-vote paired analysis.** Task correctness is determined by majority vote over Test executions. W→R counts repairs and R→W regressions. Exact two-sided McNemar p-values and their Holm-adjusted values concern majority-vote outcomes, not the mean-accuracy differences in Table 7. Task counts are shared with that table.

Benchmark	Initial (%)	Selected (%)	W→R	R→W	Exact p	Holm p
Kimi K2.6						
MATH	98.10	98.00	8	9	1.0000	1.0000
MBPP	95.34	94.82	1	2	1.0000	1.0000
PlanBench	85.34	89.01	27	13	0.0385	0.1154
Game24	69.85	97.06	77	3	1.41×10^{-19}	1.13×10^{-18}
Overall	91.01	95.67	113	27	1.04×10^{-13}	6.22×10^{-13}
Qwen3.6-27B						
MATH	97.80	95.60	6	28	1.95×10^{-4}	7.81×10^{-4}
MBPP	55.96	81.35	68	19	1.24×10^{-7}	6.18×10^{-7}
PlanBench	35.34	66.75	129	9	2.36×10^{-28}	2.13×10^{-27}
Game24	52.57	80.51	84	8	4.14×10^{-17}	2.90×10^{-16}
Overall	73.85	85.92	287	64	7.66×10^{-35}	7.66×10^{-34}

We compute 20,000 paired bootstrap draws (seed 45), resampling whole tasks within each model–category group while retaining both states and all executions together. Category counts and the task-weighted Overall mixture remain fixed. The 95% intervals use the 2.5th and 97.5th percentiles of the resampled mean-accuracy differences. This post-hoc analysis of the already-inspected Test split conditions on the observed checkpoints and executions. Its pointwise intervals quantify task-sampling uncertainty. Table 8 provides a supplementary majority-vote analysis with exact McNemar tests and Holm correction across the ten model–benchmark/Overall comparisons.

C.2 Why Endpoint Comparisons Are Insufficient

Version order records the sequence of accepted decisions; performance ordering remains a separate concept. The MBPP trajectories in Figure 5(a) illustrate why an endpoint alone leaves the revision process underdetermined: it can conceal both recovery from earlier setbacks and loss of previously attained accuracy. Since each committed state becomes the starting point for further edits, persistent revision requires assessing the path as well as the endpoint.

The separation between recurring support and candidate acceptance in Figure 5(b) makes their evidential roles operationally distinct. Recurrence supplies a reason to attempt a localized intervention; it does not establish that the generated program meets the promotion rule. Rejection can reflect an

ineffective implementation of a useful hypothesis as well as a poor hypothesis itself, so screening outcomes cannot be read as a measure of diagnostic accuracy.

The improvement retained at Terminal@E2 in Table 6 ties part of the aggregate gain to the executor left by Online Evolve. Archive selection can recover accuracy no longer retained by the ongoing state. Checkpoint quality and ongoing-state quality are therefore distinct, and a strong selected endpoint cannot establish that later revisions preserve earlier gains.

Figure 6 supplies the full category context for the MBPP illustration in Figure 5(a). A common epoch horizon can produce different numbers of committed revisions, because the proposal stream and promotion decisions determine which states enter each archive. Accordingly, Version position indexes committed edits. Evolution effort and compute remain separate quantities.

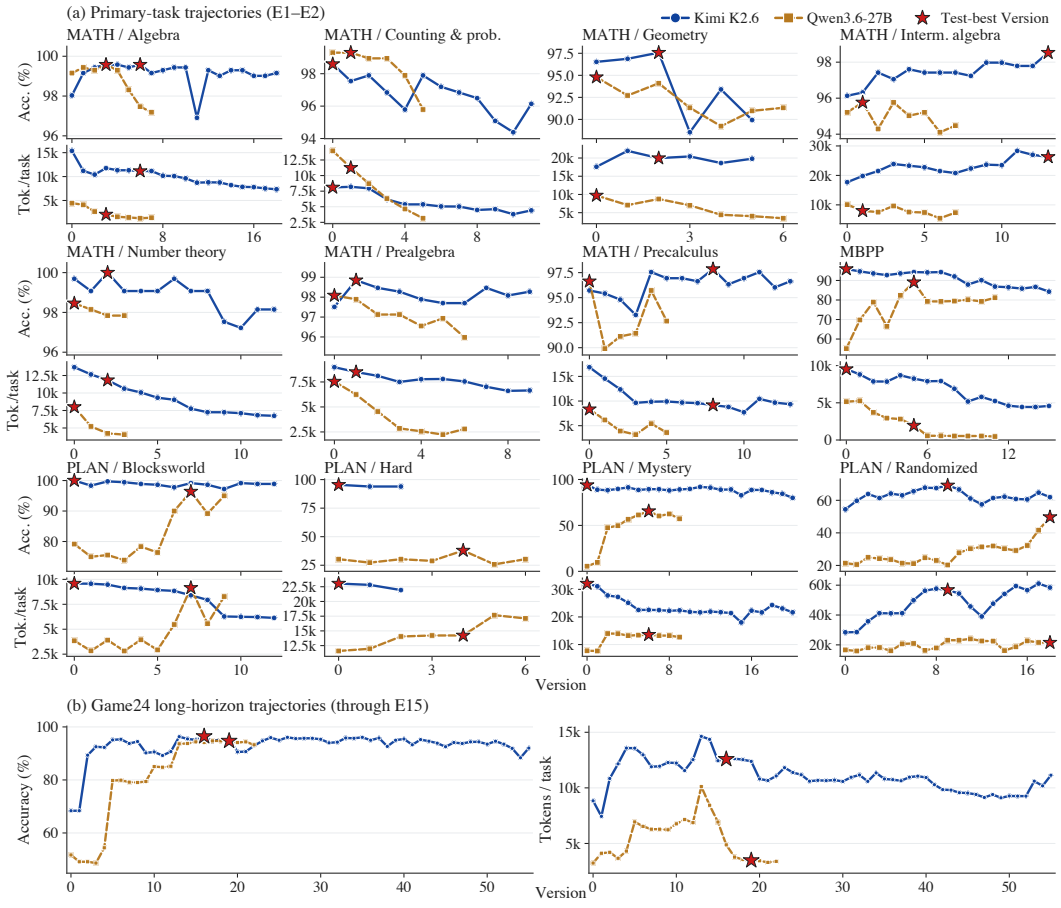

Figure 6: **Held-out accuracy and token trajectories across committed MAS Versions.** Individual revisions can improve or regress either metric. **(a)** Complete trajectories of committed MAS revisions over two evolution epochs for both models on Math, MBPP, and PlanBench, shown by task category. **(b)** Complete Game24 trajectories over 15 evolution epochs, showing the longer observed horizon. Red stars mark the retrospective Test-best Version within each displayed trajectory, not the Validation-selected checkpoint in Table 1.

C.3 Authorization Controls on the Persistent Trajectory

Figure 5(b) audits proposal triggers, candidates entering Validation, and committed states. The main-paper ablations for Kimi and Qwen on Game24 isolate the two authorization decisions: Full versus $\kappa = 1$ changes proposal authorization with paired Validation retained, and Full versus No Gate changes retention authorization with recurrence retained. The complete Qwen–Game24 ablation below adds Topology Only and Prompt Only to examine revision-space coverage.

C.3.1 Proposal Authorization: Recurrence and Revision Severity

Full and $\kappa = 1$ retain paired Validation; $\kappa = 1$ lets an isolated diagnosis authorize a candidate. Across V0–V20, Full reduces cumulative Test accuracy regression from 13.36 to 5.02 percentage points (62.4%), while regressive-transition counts change only from 8/20 to 7/20. The similar frequency and sharply lower burden show that recurrence primarily reduces harmful-revision severity by requiring cross-task support before candidate construction.

Table 9 reports stability and retrospective Oracle reachability across all five arms.

Table 9: **Ablation settings and evolution stability through V20.** Results use Qwen3.6-27B on Game24. Oracle Version identifies the retrospective Test-best state summarized in each row. Regressive steps count accuracy decreases from the preceding Version. Total regression sums these decreases, and Avg. regression averages them over all 20 transitions.

Ablation	Oracle Version	Accuracy (%)	Tokens/task	Regressive steps	Total regression (pp)	Avg. regression (pp)
Full EMAS	V19	94.73	3,481	7/20	5.02	0.25
$\kappa = 1$	V13	89.58	12,075	8/20	13.36	0.67
No gate	V16	65.56	4,262	11/20	157.97	7.90
Topology only	V16	73.41	8,422	9/20	23.53	1.18
Prompt only	V20	62.75	4,179	10/20	12.01	0.60

Figure 7 adds the corresponding accuracy and token trajectories, showing how endpoint quality and inference cost co-evolve under each revision policy.

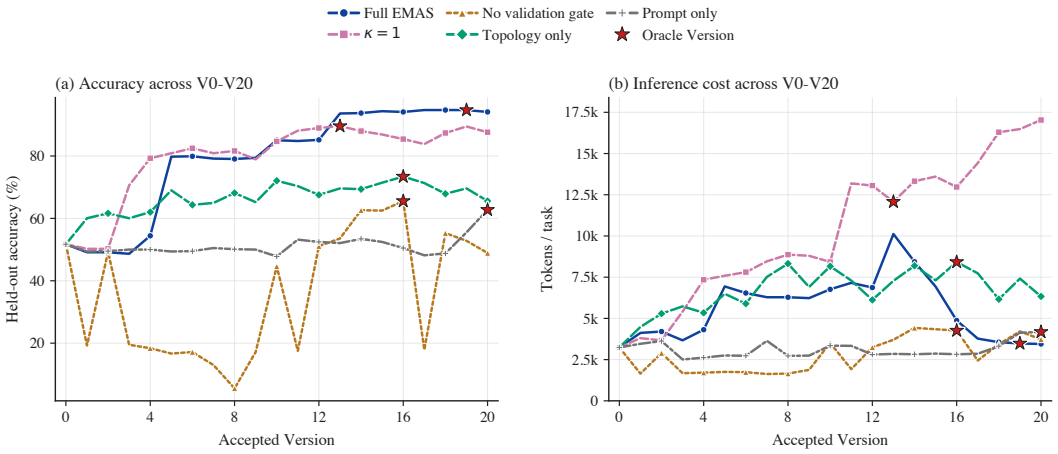

Figure 7: **Accuracy and token trajectories for the EMAS ablations through V20.** Qwen3.6-27B is evolved on Game24 using Full EMAS, $\kappa = 1$, no Validation gate, Topology Only, and Prompt Only. Panel (a) shows held-out accuracy; Panel (b) shows tokens per task. Red stars mark the retrospective Test-best Version in each common V0–V20 trajectory.

C.3.2 Joint Revision of Topology and Prompts

Topology Only and Prompt Only retain recurrence and paired Validation while restricting the primary revision channel. Prompt Only preserves topology and edits an evidence-located prompt. Topology Only restricts primary operations to structural changes, with directly required prompt adjustments permitted by the bounded closure. Their Oracle accuracies reach 73.41% and 62.75%, respectively, compared with 94.73% for Full EMAS; their cumulative regression reaches 23.53 and 12.01 percentage points, compared with 5.02 for Full EMAS. Prompt edits cannot remove structural bottlenecks, and structural primary operations cannot directly target instruction-localized failures. Full EMAS can select and coordinate both channels, expanding the reachable solution space while producing the strongest and most stable trajectory.

C.3.3 Retention Authorization: Paired Validation and Persistent Damage

Full and No Gate retain recurrence; No Gate removes paired Validation before commitment. Across V0–V20, Full EMAS records 5.02 percentage points of cumulative Test accuracy regression, compared with 157.97 for No Gate. This gap shows that paired Validation protects accumulated progress by screening a candidate before it becomes the parent of subsequent tasks and revisions.

C.3.4 Positive Drift and Validation-Margin Transfer

Table 10 summarizes held-out outcomes among the 255 revisions accepted through E2. For the 84 accuracy-oriented commits, positive Test transitions outnumber regressions two to one (48 versus 24), with 12 flat transitions and a +2.75-point mean change. Thus, the accepted path can be locally non-monotonic while remaining positively directed overall.

Table 10: **Accepted-revision transfer through E2.** The table summarizes all exact-predecessor revisions accepted through E2. Gain and loss are conditional magnitudes of positive and negative Test task-majority changes; net change averages all accepted revisions, with 95% intervals obtained by resampling complete model–benchmark–category trajectories.

Accepted-transition balance through E2 **255/1,021 validated candidates accepted (25.0%)**

Objective	Accepted	Test +/Flat/−	Mean gain (pp)	Mean loss (pp)	Net ΔAcc./step [95% CI]	Mean ΔTokens/task
All	255	93/58/104	+4.18	2.43	+0.53 [-0.001, +1.181]	-86
Accuracy	84	48/12/24	+6.59	3.56	+2.75 [+1.395, +4.314]	+1,692
Cost	171	45/46/80	+1.60	2.09	-0.56 [-0.926, -0.214]	-960

+, Flat, and − refer to Test task-majority accuracy; negative token changes denote savings.

EMAS deliberately uses the smallest positive correct-count margin to admit incremental accuracy revisions. Table 11 shows that this permissive rule produces positive aggregate transfer. Under the operational $\Delta C \geq 1$ rule, positive transitions dominate regressions and the mean Test change is positive. Across the nested larger-margin subsets, the positive-transfer rate rises from 57.1% to 77.5%, the regression rate falls from 28.6% to 15.0%, and mean Test improvement grows from +2.75 to +5.92 percentage points.

Table 11: **Held-out transfer strengthens with observed Validation margin.** Among the 84 accuracy revisions accepted under the operational $\Delta C \geq 1$ rule, positive Test transitions outnumber regressions 48 to 24 and the mean Test change is +2.75 percentage points. Nested subsets with larger observed margins show higher positive-transfer rates, lower regression rates, and larger mean Test gains.

Accuracy-oriented commits **84 accepted revisions**

Observed margin	n	Test +/Flat/−	Positive (%)	Regression (%)	Mean ΔAcc. (pp) [95% CI]
Operational $\Delta C \geq 1$	84	48/12/24	57.1	28.6	+2.75 [+1.395, +4.314]
$\Delta C \geq 2$	64	40/7/17	62.5	26.6	+3.43 [+1.618, +5.636]
$\Delta C \geq 3$	53	38/5/10	71.7	18.9	+4.41 [+2.417, +6.441]
$\Delta C \geq 5$	40	31/3/6	77.5	15.0	+5.92 [+3.788, +8.899]

Intervals resample complete model–benchmark–category trajectories. Test outcomes use task-majority correctness. Rows are nested subsets of the realized accuracy commits.

Table 12: **Evolution cost breakdown through Epoch 2.** Token totals are shown by execution component and backbone. Paired Validation is a subtotal of the two Validation rows and is not added again to the total.

Component	Kimi K2.6	Qwen3.6-27B	All	Share
Evolution stream	173.96M	71.76M	245.72M	8.39%
Current Validation	872.88M	419.03M	1.292B	44.11%
Candidate Validation	854.67M	354.77M	1.209B	41.29%
Post-acceptance replay	161.30M	20.52M	181.83M	6.21%
Paired Validation subtotal	*1.728B*	*773.80M*	*2.501B*	*85.40%*
Total	**2.063B**	**866.08M**	**2.929B**	**100.00%**

C.4 Evolution and Deployment Cost

Table 12 accounts for computation incurred during evolution. Its concentration in paired Current/Candidate Validation connects the dominant expenditure to repeated promotion decisions. This distinguishes the expense of authorizing persistent changes from the inference cost of using the state that those decisions produce.

The decomposition should be interpreted alongside the gate ablation: screening consumes computation, but removing it also changes the executor available to later tasks. The present measurements report these two consequences separately. They do not establish a workload length at which the evolution expenditure is amortized.

D Additional Diagnostics and Complete Records

D.1 Retrospective Archive Reachability and Provenance

Tables 13 and 14 provide the Test-best archive diagnostic and its category-level provenance. The distinction from the primary Validation-selected result matters because a reachable state need not be the state identified without Test feedback. Ordinary confidence intervals do not adjust for selecting a Version on Test.

Table 13: **Retrospective Oracle@E2 diagnostic.** The category-wise Test-best EMAS archive composite improves mean accuracy over the Initial MAS in seven of eight model–benchmark settings and preserves it in the remaining setting. Genesis, Initial MAS, and Oracle@E2 are compared with Genesis+SOP and three external baselines on the same Test tasks. This diagnostic characterizes archive reachability and is not used for the main ranking. All competing baselines average repeats 1–3. Cells report mean accuracy and tokens per task with execution-level 95% confidence intervals. Intervals condition on the fixed executor and do not adjust for Test-based selection.

Kimi K2.6

Benchmark	Method	Runs	Accuracy [95% CI]	Tokens/task [95% CI]
Math	Genesis	3	78.37 [75.71, 81.02]	609 [599, 619]
	Initial MAS	3	97.43 [96.31, 98.55]	14186 [14108, 14264]
	Genesis+SOP	3	39.10 [38.20, 40.00]	1485 [1461, 1508]
	GPTSwarm	3	84.97 [84.82, 85.11]	13784 [13697, 13871]
	AFlow	3	89.80 [89.22, 90.38]	8121 [8105, 8137]
	ADAS	3	81.00 [79.45, 82.55]	1983 [1958, 2009]
	EMAS Oracle@E2	3	98.83 [98.45, 99.21]	13842 [13785, 13898]
MBPP	Genesis	3	68.91 [67.62, 70.20]	279 [230, 328]
	Initial MAS	3	95.85 [93.28, 98.43]	9532 [9228, 9836]
	Genesis+SOP	3	85.66 [83.70, 87.63]	1355 [1350, 1359]
	GPTSwarm	3	88.08 [84.22, 91.94]	13718 [13650, 13787]
	AFlow	3	89.81 [88.61, 91.01]	3685 [3645, 3725]
	ADAS	3	82.73 [77.53, 87.93]	4115 [4046, 4185]
	EMAS Oracle@E2	3	95.85 [93.28, 98.43]	9532 [9228, 9836]
PlanBench	Genesis	3	14.40 [13.75, 15.05]	2171 [2094, 2247]
	Initial MAS	3	83.51 [81.16, 85.85]	23345 [23048, 23642]
	Genesis+SOP	3	14.75 [13.39, 16.10]	2392 [2326, 2457]
	GPTSwarm	3	36.39 [34.04, 38.73]	30655 [30538, 30773]
	AFlow	3	60.03 [59.12, 60.95]	7618 [7594, 7642]
	ADAS	3	3.49 [1.03, 5.95]	5260 [5204, 5317]
	EMAS Oracle@E2	3	88.13 [86.78, 89.49]	32265 [31699, 32831]
Game24	Genesis	3	15.69 [14.29, 17.08]	119 [97, 141]
	Initial MAS	3	68.38 [62.39, 74.37]	8838 [8764, 8912]
	Genesis+SOP	3	18.38 [17.47, 19.30]	1335 [1313, 1356]
	GPTSwarm	3	81.99 [77.80, 86.17]	20821 [20606, 21035]
	AFlow	3	95.10 [94.58, 95.62]	2782 [2611, 2952]
	ADAS	3	96.20 [94.81, 97.60]	6927 [6794, 7060]
	EMAS Oracle@E2	3	95.34 [91.65, 99.03]	12971 [12165, 13776]

Qwen3.6-27B

Benchmark	Method	Runs	Accuracy [95% CI]	Tokens/task [95% CI]
Math	Genesis	3	38.33 [38.19, 38.48]	126 [126, 126]
	Initial MAS	3	97.50 [96.84, 98.16]	8224 [8204, 8244]
	Genesis+SOP	3	28.87 [27.99, 29.74]	873 [873, 873]
	GPTSwarm	3	75.83 [75.69, 75.98]	15397 [15335, 15458]
	AFlow	3	97.60 [97.51, 97.69]	4629 [4601, 4658]
	ADAS	3	90.57 [90.19, 90.95]	1494 [1456, 1531]
	EMAS Oracle@E2	3	97.70 [97.04, 98.36]	7025 [6925, 7124]
MBPP	Genesis	3	58.38 [57.63, 59.12]	274 [266, 282]
	Initial MAS	3	55.09 [49.74, 60.45]	5155 [4811, 5498]
	Genesis+SOP	3	59.93 [59.19, 60.67]	945 [940, 950]
	GPTSwarm	3	82.21 [80.24, 84.18]	3130 [2954, 3305]
	AFlow	3	93.09 [91.89, 94.30]	4524 [4485, 4562]
	ADAS	3	86.70 [84.74, 88.67]	730 [680, 780]
	EMAS Oracle@E2	3	89.12 [87.83, 90.41]	1949 [1857, 2041]
PlanBench	Genesis	3	14.14 [14.14, 14.14]	899 [859, 940]
	Initial MAS	3	35.08 [32.24, 37.91]	9536 [9336, 9736]
	Genesis+SOP	3	13.35 [13.35, 13.35]	1727 [1711, 1742]
	GPTSwarm	3	14.40 [13.75, 15.05]	8576 [8379, 8772]
	AFlow	3	51.66 [50.42, 52.90]	33646 [33247, 34045]
	ADAS	3	33.07 [30.61, 35.53]	3683 [3663, 3704]
	EMAS Oracle@E2	3	68.67 [66.58, 70.76]	14713 [14672, 14754]
Game24	Genesis	3	14.95 [13.90, 16.01]	123 [123, 123]
	Initial MAS	3	51.72 [48.26, 55.17]	3238 [3076, 3399]
	Genesis+SOP	3	14.83 [14.30, 15.36]	775 [774, 775]
	GPTSwarm	3	16.54 [14.96, 18.13]	912 [912, 912]
	AFlow	3	15.44 [15.10, 15.78]	3155 [3131, 3179]
	ADAS	3	67.03 [63.34, 70.73]	1457 [1412, 1502]
	EMAS Oracle@E2	3	79.90 [75.59, 84.22]	6539 [6370, 6708]

Table 14: **Category-level provenance of the retrospective Oracle@E2 diagnostic.** For each model and category, the table identifies the Test-best Version within the fixed two-epoch evolution budget. Version numbers index committed revisions; V0 denotes the Initial MAS. These Versions are not used for the primary ranking.

Kimi K2.6

Benchmark	Category	Oracle Version	Accuracy (%)	Tokens/task
Math	algebra	V6	99.58	11,153
	count./prob.	V0	98.60	8,095
	geometry	V2	97.57	19,971
	interm. algebra	V13	98.53	26,280
	number theory	V2	100.00	11,833
	prealgebra	V1	98.85	8,494
	precalculus	V8	97.86	9,173
MBPP	all	V0	95.85	9,532
PlanBench	blocks.	V0	100.00	9,581
	blocks. hard	V0	95.45	23,040
	mystery	V0	93.89	32,099
	randomized	V9	69.17	56,807
Game24	all	V6	95.34	12,971

Qwen3.6-27B

Benchmark	Category	Oracle Version	Accuracy (%)	Tokens/task
Math	algebra	V3	99.58	2,071
	count./prob.	V1	99.30	11,242
	geometry	V0	94.79	9,719
	interm. algebra	V1	95.76	7,996
	number theory	V0	98.46	8,004
	prealgebra	V0	98.08	7,541
	precalculus	V0	96.64	8,343
MBPP	all	V5	89.12	1,949
PlanBench	blocks.	V7	96.39	9,151
	blocks. hard	V4	37.88	14,228
	mystery	V6	65.56	13,568
	randomized	V18	49.72	21,509
Game24	all	V6	79.90	6,539

Long-horizon archive frontier. Table 15 gives the E2 and E15 archive diagnostics for both backbones, complementing the continuations examined in Section 4.2. Since the best-so-far accuracy cannot decrease when an archive is enlarged, the relevant interpretation is the additional empirical

reachability within this longer observation window. This diagnostic supplies neither monotonic Version performance nor a Test-independent rule for identifying the added states, and the longer-horizon evidence is confined to Game24.

Table 15: **Retrospective Game24 archive frontier at two horizons.** For each model, E2 and E15 report the Test-best Version within the corresponding evolution horizon, with accuracy ties resolved by fewer tokens and then earlier Version. Oracle Version is formatted as Version/Epoch. These descriptive results are not Test-independent checkpoint selections.

Model	Endpoint	Oracle Version	Accuracy (%)	Tokens/task
Kimi	E2	V6/E1	95.34	12,971
	E15	V16/E4	96.45	12,585
Qwen	E2	V6/E2	79.90	6,539
	E15	V19/E10	94.73	3,481

D.2 Accuracy–Token Archive Diagnostics

Figure 8 treats accuracy and token use jointly. When several Versions lie on a category's empirical frontier, selecting an operating state requires specifying how accuracy and token use are prioritized. Version recency alone does not resolve that choice. As with the Oracle summaries, these frontiers are descriptive because they use Test outcomes.

D.3 Full Trigger Records

Table 16 records how this policy translates into revision activity. Triggers, candidates entering Validation, and accepted candidates have different denominators and should not be conflated into a single acceptance rate. Backbone comparisons are descriptive because each backbone also uses its own model-conditioned Validation split.

D.4 Post-hoc Headroom Diagnostics

Table 17 concerns the 93 accepted Epoch-2 revisions, a subset of the E1–E2 population above. Conditioning on successful accuracy revisions relates available headroom to the magnitude of a gain; it does not determine the probability that a candidate will transfer. The V0 grouping and exact-predecessor grouping answer different questions: the former describes the initial task regime, while the latter describes the state on which the particular edit acts.

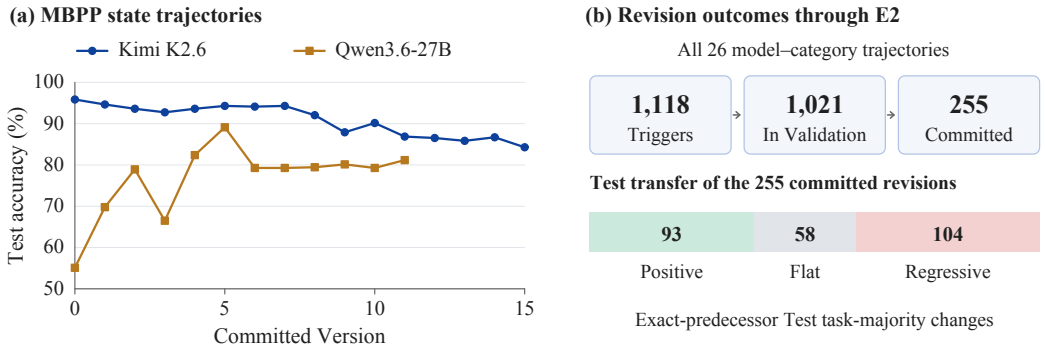

Figure 5: **Version trajectories and revision outcomes.** **(a)** Complete E1–E2 MBPP accuracy trajectories for both backbones, selected as an illustration with opposite net changes; points show mean Test accuracy. The horizontal axis counts committed revisions. Full category trajectories appear in Appendix Figure 6. **(b)** Counts across all 26 model–category trajectories through E2. Validation candidates are the subset entering paired evaluation; accepted candidates become committed Versions. Test outcomes classify all accepted revisions against their exact predecessors using task-majority correctness. These counts include both revision objectives and are not restricted to MBPP.

Table 16: **A selective subset of trigger events becomes committed MAS revisions across two evolution epochs.** For each benchmark–category–model trajectory, the table reports the number of triggers by permitted operation and whether each trigger is ultimately accepted or rejected. Rows aggregate each trajectory from E1 through E2; the six operation columns sum to Total, Acc. and Rej. partition Total, and Acc. rate is Acc./Total. Across all trajectories, EMAS activates 1,118 triggers, of which 255 become committed revisions (22.8%).

Benchmark	Category	Model	Prompt	−Node	Split	−Edge	+Node	+Edge	Acc.	Rej.	Total	Acc. rate
Math	algebra	Kimi K2.6	42	1	5	0	0	0	18	30	48	37.5%
		Qwen3.6-27B	21	32	0	0	0	0	7	46	53	13.2%
	count./prob.	Kimi K2.6	21	1	0	0	0	0	11	11	22	50.0%
		Qwen3.6-27B	15	7	0	0	0	0	5	17	22	22.7%
	geometry	Kimi K2.6	5	0	14	0	0	0	5	14	19	26.3%
		Qwen3.6-27B	8	8	4	0	0	0	6	14	20	30.0%
	interm. algebra	Kimi K2.6	25	0	11	0	1	1	13	25	38	34.2%
		Qwen3.6-27B	13	19	6	0	0	0	7	31	38	18.4%
	number theory	Kimi K2.6	24	0	0	0	0	0	12	12	24	50.0%
		Qwen3.6-27B	2	22	0	0	0	0	3	21	24	12.5%
	prealgebra	Kimi K2.6	27	0	0	9	0	0	9	27	36	25.0%
		Qwen3.6-27B	19	15	2	0	0	0	6	30	36	16.7%
	precalculus	Kimi K2.6	16	4	2	0	0	0	13	9	22	59.1%
		Qwen3.6-27B	7	3	12	0	0	0	5	17	22	22.7%
MBPP	all	Kimi K2.6	42	0	0	4	0	0	15	31	46	32.6%
		Qwen3.6-27B	36	6	1	0	3	0	11	35	46	23.9%
PlanBench	blocks.	Kimi K2.6	70	8	0	9	0	0	12	75	87	13.8%
		Qwen3.6-27B	4	46	4	0	6	0	9	51	60	15.0%
	blocks. hard	Kimi K2.6	7	4	0	0	0	0	2	9	11	18.2%
		Qwen3.6-27B	14	0	1	0	0	0	6	9	15	40.0%
	mystery	Kimi K2.6	38	18	4	14	1	1	20	56	76	26.3%
		Qwen3.6-27B	50	22	9	0	0	0	9	72	81	11.1%
	randomized	Kimi K2.6	26	12	17	7	4	8	18	56	74	24.3%
		Qwen3.6-27B	62	9	11	0	4	0	18	68	86	20.9%
Game24	all	Kimi K2.6	52	0	3	0	0	1	9	47	56	16.1%
		Qwen3.6-27B	54	2	0	0	0	0	6	50	56	10.7%
Kimi K2.6 overall			395	48	56	43	6	11	157	402	559	28.1%
Qwen3.6-27B overall			305	191	50	0	13	0	98	461	559	17.5%
Overall			**700**	**239**	**106**	**43**	**19**	**11**	**255**	**863**	**1118**	**22.8%**

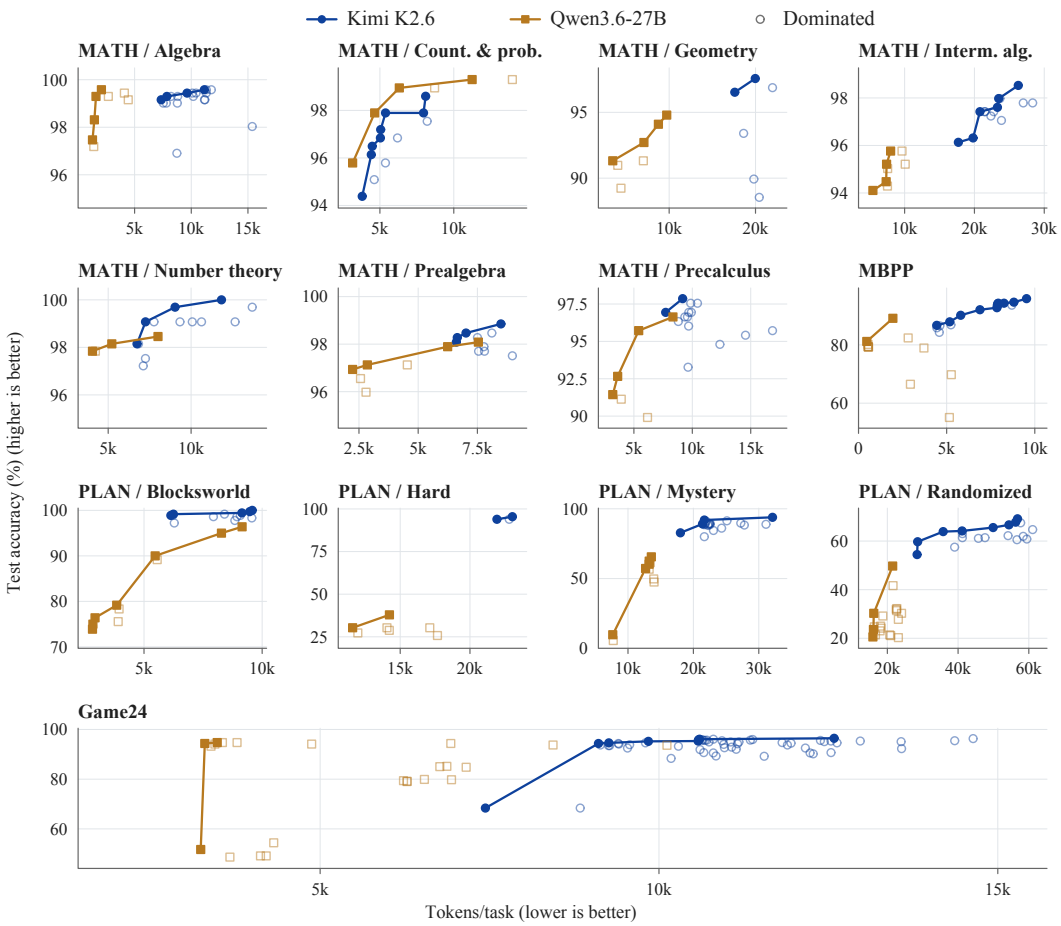

Figure 8: **Accuracy–token trade-offs across evaluated MAS Versions.** Each panel shows one benchmark category for both models. Math, MBPP, and PlanBench include Versions from two evolution epochs; Game24 includes Versions through Epoch 15. Filled markers and connecting lines show each model's category-specific Pareto frontier; open markers indicate dominated Versions. Higher accuracy and fewer tokens are preferred.

Table 17: **Accuracy headroom and held-out outcomes of committed revisions.** The 93 Epoch-2 revisions are grouped by V0 accuracy in Panel A and by preceding-Version accuracy in Panel B; all accuracy quantities use Test task-majority correctness. Lower accuracy indicates greater headroom. Regressed counts revisions that lower Test accuracy. Successful means higher accuracy for accuracy-directed revisions, or fewer tokens without an accuracy decrease for cost-directed revisions. Losses average over regressed revisions; gains and token savings condition on successful accuracy-directed and cost-directed revisions, respectively.

Panel A: V0 accuracy (global headroom)

V0 accuracy bin	n	Regressed n (%)	Avg. loss (pp)	Successful n (%)	Acc. gain (pp)	Tokens saved/task
Low (<50)	15	6 (40.0%)	5.81	8 (53.3%)	6.88	1,072
Medium (50–80)	19	9 (47.4%)	4.32	9 (47.4%)	3.32	32
High (80–95)	2	1 (50.0%)	1.04	1 (50.0%)	–	372
Saturated (≥ 95)	57	23 (40.4%)	1.89	30 (52.6%)	3.01	453

Panel B: exact-predecessor accuracy (local headroom)

Predecessor bin	n	Regressed n (%)	Avg. loss (pp)	Successful n (%)	Acc. gain (pp)	Tokens saved/task
Low (<50)	12	4 (33.3%)	5.80	8 (66.7%)	6.88	1,072
Medium (50–80)	17	8 (47.1%)	4.96	8 (47.1%)	3.48	32
High (80–95)	19	10 (52.6%)	2.11	9 (47.4%)	3.90	705
Saturated (≥ 95)	45	17 (37.8%)	2.02	23 (51.1%)	1.35	398