TGO-MAS: TEMPORAL GRAPH ORGANIZATION FOR MULTI-AGENT SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model systems with multiple agents must decide both which agents interact and when each interaction becomes available during execution over multiple steps. Workflow descriptions and static communication graphs can encode endpoints or order, but they do not make the activation step of a typed interaction an explicit decision variable. We therefore formulate temporal graph organization that is aware of execution steps around the insight that *interaction timing is part of organization rather than execution metadata*. TGO-MAS instantiates this formulation as a closed scheduling loop. It represents candidates that pair typed interactions with activation steps, scores them from the observed execution prefix, selects a feasible set for the next step, and revises the next decision after delivery, failure, or completion. Structural analysis establishes several consequences of this construction. Static projections can discard temporal distinctions, static schedules are contained in the action space for the next step, and the optional accounting and event record extensions satisfy stated feasibility and reconstruction conditions. TGO-MAS outperforms the strongest baseline, the DyTopo reimplementation, by 2.94% on HumanEval-heldout-124 and 1.88% on MMLU-validation-1531.

1 INTRODUCTION

Large language model (LLM) multiagent systems (MASs) coordinate specialized agents through messages, workflows, and graph shaped communication (Wu et al., 2024; Hong et al., 2024; Li et al., 2023; Zhuge et al., 2024; Zhang et al., 2025d). During an execution with multiple steps, however, a handoff or report is useful only when its payload arrives at the right point. Organization must therefore decide both which typed interaction occurs and when it activates.

A workflow with a planner, coder, and verifier illustrates this timing dependence. Planning advice delivered before implementation can narrow the task and prevent rework, but the same advice may disrupt progress once coding has begun. Verification cannot guide implementation before an artifact exists, while a correction delivered after integration may arrive too late or supersede an earlier report. Thus, even with a fixed roster and fixed interaction types, the useful schedule changes as outputs become available, dependencies resolve, and failures alter the execution state.

Existing interfaces expose only part of this decision. Roles and workflows distribute interaction choices across prompts and controllers, while static graphs expose endpoints but not early, late, or single activation of the same typed edge. Dynamic organizers update connectivity during execution (Lu et al., 2026; Tian et al., 2026), yet an interface for step aware scheduling must also distinguish readiness from estimated value, preserve order, enforce resource feasibility, and replan after delivery, failure, or completion. Execution logs record what happened after the fact but do not specify the next feasible interaction.

The resulting research question asks whether a MAS organizer can use only the observed execution history to schedule which typed interactions should activate next. It may use the current phase, available outputs, unresolved dependencies, prior message and tool events, and remaining resources, but not evaluator labels, latent future availability, or future outcomes. This information boundary specifies what the organizer may observe and excludes future outcomes from the schedule. It also motivates a closed design in which the organizer encodes the observed prefix, assesses readiness separately from estimated value, selects a feasible set for the next step, and revises that decision from observed lifecycle feedback.

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.

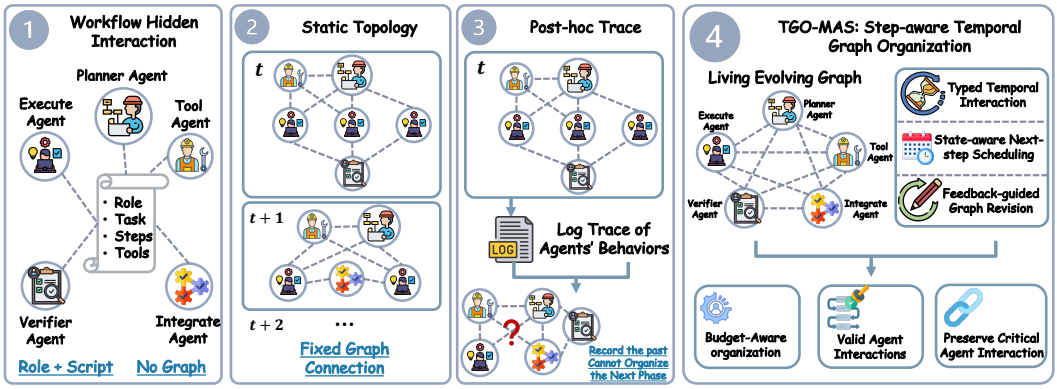


Figure 1: TGO-MAS replaces hidden workflows, static topology, and retrospective traces with step aware temporal graph organization.

Figure 1 summarizes TGO-MAS (Temporal Graph Organization for Multi-Agent Systems). Its **observable state encoder** summarizes the prefix. The **typed temporal candidate assessor** separates readiness from value. The **feasible interaction scheduler** selects the next set under resource constraints. An **observed feedback update** records lifecycle events before replanning. The core schedules the next step, while an optional extension reserves later interactions.

The formulation fixes the decision object and information boundary. The method instantiates both, and theory establishes representation, containment, feasibility, and reconstruction. Empirical Studies test temporal information, while research questions RQ1 through RQ5 cover quality, efficiency, attribution, timing interventions, robustness, and controls. Our contributions are exactly threefold.

- **Problem Formalization.** We formulate MAS organization as prefix constrained, step aware scheduling of typed interactions, making activation time, readiness, feasibility, and feedback revision explicit beyond static endpoints and retrospective traces.
- **Method Design.** TGO-MAS couples prefix encoding, separate readiness and value assessment, feasible temporal selection, and feedback based replanning, directly instantiating the formal requirements.
- **Empirical Evaluation.** In matched comparisons, TGO-MAS exceeds DyTopo by 2.94% on HumanEval-heldout-124 and 1.88% on MMLU-validation-1531 while using 4.21% fewer HumanEval tokens. Without inferential intervals, these results remain descriptive.

2 Related Work

MAS organization and graph design. Roles and workflows prescribe interactions (Wu et al., 2024; Hong et al., 2024; Li et al., 2023; Zhuge et al., 2024; Zhang et al., 2025d), while graph design methods optimize topology (Zhang et al., 2025c;b; Jiang et al., 2026). Both organize roles or endpoints without necessarily making each typed interaction's activation step selectable.

Dynamic and temporal communication. DyTopo reconstructs connectivity, QueenBee produces temporal DAGs, and learned communication studies whom and when to connect under resource constraints (Lu et al., 2026; Tian et al., 2026; Jiang & Lu, 2018; Singh et al., 2019; Kim et al., 2019; Das et al., 2019; Ding et al., 2020; Liu et al., 2023). Because these precedents establish time varying organization, TGO-MAS is distinguished by a decision interface that schedules typed interactions with explicit activation steps from an observed prefix, separates readiness from value, enforces feasibility, and replans after feedback.

Temporal representation and provenance. Temporal graph models index relations by time, and provenance standards order records and derivations (Rossi et al., 2020; Trivedi et al., 2019; Kazemi et al., 2020; Lebo et al., 2013). TGO-MAS uses these ideas for activation timing, lifecycle records, feasibility, and reconstruction, not task superiority, interpretability, or universal diagnostic value. Appendix B provides comparisons, proofs, protocols, and results.

3 PROBLEM FORMULATION

Conventional MAS graphs specify who may interact, whereas our decision object also specifies when each typed interaction activates. We schedule online from the observable prefix while excluding future outcomes and evaluator information unavailable during execution.

3.1 TYPED TEMPORAL DECISION OBJECT

Let the temporal schedule, fixed execution contract, and typed interaction space be

$$\begin{aligned} &\Sigma = (E_1, \dots, E_T), \quad E_a \subseteq \mathcal{E}, \quad q = (u, v, r, a), \\ &\Gamma = (V, \mathcal{R}, T, \Pi, B_0, L_{1:T}), \quad \mathcal{E} = \{(u, v, r) : u, v \in V,\ u \neq v,\ r \in \mathcal{R}\}. \end{aligned} \tag{1}$$

Here, V, $\mathcal{R}$, T, Π, B_0, and $L_{1:T}$ denote the roster, interaction types, horizon, fixed policies and interfaces, episode budget, and step capacities, respectively. Membership $(u, v, r) \in E_a$ activates an interaction from u to v with type r at step a. The type identifies the receiver's function, such as request, proposal, critique, or verification, allowing one agent pair to serve distinct roles. Hence $q = (u, v, r, a)$ is the elementary organizational decision and includes activation time, while Γ fixes all other factors for direct comparisons.

A static topology fixes interactions independently of the prefix. Our temporal representation preserves activation order, whose loss under static projection is characterized theoretically.

3.2 PREFIX CONSTRAINED ONLINE SCHEDULING

At step t, $\mathcal{O}_{\le t}$ denotes the observable prefix of outputs, task phase, unresolved dependencies, delivered messages, tool observations, failures, prior decisions, and remaining resources. It excludes future outcomes, evaluator labels, latent true dependency windows, and future agent states, preventing use of information unavailable at deployment. The method independent organizer interface and admission constraints are

$$\begin{aligned} &\mathcal{C}_t \subseteq \mathcal{E} \times \{t+1\}, \qquad A_t = \mathrm{Org}(\mathcal{O}_{\le t}) \in \mathcal{F}_t, \quad A_t \subseteq \mathcal{C}_t, \\ &\mathcal{F}_t = \{A \subseteq \mathcal{C}_t : C(A) \le B_t,\ W_{t+1}(A) \le L_{t+1},\ \mathrm{Valid}(A) = 1\}. \end{aligned} \tag{2}$$

The candidate set contains typed interactions for the next step. The feasible family admits sets whose cost $C(A)$ fits budget B_t, whose load $W_{t+1}(A)$ fits capacity L_{t+1}, and whose prerequisites, roles, and endpoints are valid. It thereby defines an executable action space without specifying how a method scores or solves candidates.

After A_t executes, observed delivery, failure, cancellation, and completion events extend the prefix to $\mathcal{O}_{\le t+1}$ and trigger replanning, so earlier decisions cannot use information revealed later.

3.3 TASK OBJECTIVE AND SCOPE

For an unseen task x, the organizer and fixed contract jointly produce the MAS output. With evaluation set $\mathcal{D}_{\mathrm{eval}} = \{(x_i, \nu_i)\}_{i=1}^{N}$ and evaluator ν_i specific to each task, system quality is

$$\begin{aligned} \hat{y} &= F_{\mathrm{Org},\Gamma}(x) = \mathrm{MASExec}_{\Gamma}(x; A_0, \dots, A_{T-1}), \\ \mathrm{Score}(F_{\mathrm{Org},\Gamma}; \mathcal{D}_{\mathrm{eval}}) &= \tfrac{1}{N} \textstyle\sum_{i=1}^{N} \nu_i(F_{\mathrm{Org},\Gamma}(x_i)). \end{aligned} \tag{3}$$

Policies fixed in Γ generate and process payloads, while the organizer controls their delivery and timing. Thus generation, prompting, or tool changes are not credited to scheduling. Under one information contract, task evaluators cover code generation, answer selection, and timing diagnostics without entering organizer inputs.

Step Aware Temporal Organization Problem. Given x and Γ, choose at each step a feasible typed interaction set A_t from $\mathcal{O}_{\le t}$ to achieve high utility under shared resource and execution constraints. The fixed roster excludes self interaction. Persistent channels, agent creation, asynchronous dispatch, payload semantics, and estimate correctness are outside scope. TGO-MAS instantiates Org without changing the objective or information boundary. Appendix C gives extended notation and cancellation semantics.

4 EMPIRICAL STUDIES

We conduct three registered diagnostics to test whether timing, recipient identity, dependency-window alignment, and ordered lifecycle history preserve distinctions discarded by a static endpoint projection. Studies A and B measure required-window coverage retention under controlled interventions, while Study C compares history views for choosing the next interaction. These results motivate the representation rather than serve as premises of TGO-MAS. Section 6 evaluates downstream task utility, and Appendix J provides the complete numeric backing.

Study A on timing and recipient identity. Both interventions reduce coverage for every organizer in Table 1. Recipient shuffling is consistently more disruptive than time permutation, with decreases of 36.95–47.10 points versus 9.13–37.66 points. Window width specifically moderates timing: AgentPrune loses 37.66 points under the tight window but 12.58 points under the wide window, whereas its recipient-shuffle decrease remains 44.61–47.10 points. DyTopo records the highest clean, time-permuted, and recipient-shuffled values at 78.45%, 69.32%, and 36.12%. The pattern separates when an interaction occurs from whom it reaches, but does not identify either intervention as a general cause of downstream quality.

Table 1: Coverage under timing and recipient interventions (%).

Method	Window	Clean	Time permuted	Recipient shuffled
G-Designer	tight, wide	61.87	47.23	23.91
AgentPrune	tight	75.04	37.38	27.94
AgentPrune	wide	73.12	60.54	28.51
CARD	tight, wide	64.29	50.18	27.34
DyTopo	tight, wide	78.45	69.32	36.12

Study B on the dependency-window boundary. Figure 2 shows a boundary-sensitive response to signed temporal shifts. At $\delta = 0$, every method retains 100% coverage. Under tight windows, retention after a one-step shift spans 52.89–68.75% and falls to 23.56–42.34% after a two-step shift. The corresponding ranges under wide windows are 90.67–95.12% and 72.14–85.32%. Earlier and later shifts both reduce retention, while wider windows absorb more displacement before a required interaction crosses the registered boundary. The comparison therefore supports timing alignment, not a universal preference for earlier communication. Lines connect the five registered shift settings only and do not imply continuity between them.

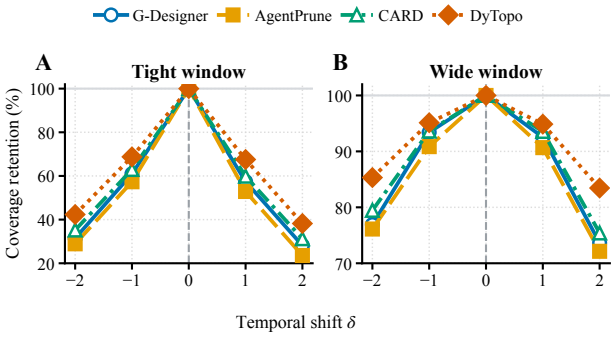


Figure 2: Required-window retention under temporal shifts. Panels use window-specific y-axis intervals.

Study C on history views for organization. Table 2 exposes a trade-off obscured by success alone. Structured lifecycle history records 92.54% success, 87.65% ranking, and 8.45% redundancy. Against raw ordered history, it adds 21.09 success and 19.81 ranking points while cutting redundancy by 14.67 points. Static top-1 eliminates redundancy but remains near chance; activating both reaches 100.00% success with 50.76% redundancy and undefined ranking. Only structured history exceeds 90% success and 80% ranking below 10% redundancy, supporting lifecycle-aware replanning in this task without establishing downstream utility or interpretability.

Table 2: History-view trade-offs for next-edge selection (%).

History view	Next-edge success ↑	Redundancy rate ↓	Ranking accuracy ↑
Static top-1	50.82	0.00	49.43
Activate both	100.00	50.76	N/A
Raw ordered history	71.45	23.12	67.84
Structured lifecycle history	92.54	8.45	87.65

5 TGO-MAS

Figure 3 summarizes TGO-MAS as a closed scheduling loop. At each execution step, the organizer encodes the observed MAS state, scores typed temporal candidates, selects and activates a feasible interaction set, and replans from delivery outcomes and ordered lifecycle transitions.

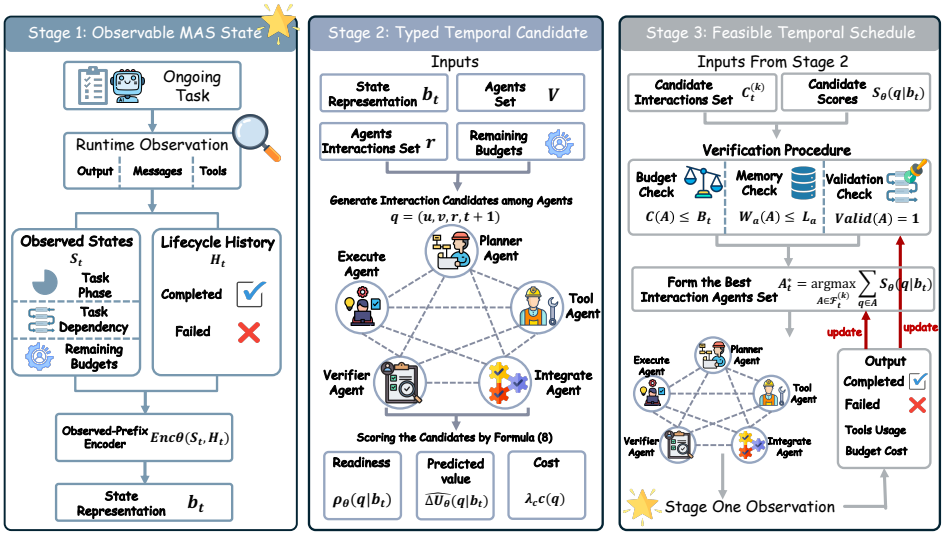


Figure 3: TGO-MAS selects feasible typed interactions from the observed MAS state and replans after feedback. Advance commitments, reservations, and state reconstruction are optional extensions.

Observable MAS state. The first module collects the task phase, available outputs, unresolved dependencies, delivered messages, tool observations, remaining budget, and declared capacities available at decision step t. We group these quantities as

$$S_t = (P_t, Z_t, D_t, M_{\le t}, O_{\le t}, B_t, L_{t:T}), \tag{4}$$

where P_t is the task phase, Z_t contains available agent outputs, D_t records unresolved dependencies, and $M_{\le t}$ and $O_{\le t}$ are the delivered message and tool observation histories. The scalar B_t is the remaining episode budget, while the sequence $L_{t:T}$ gives the capacities declared for steps t through T. Together with the ordered lifecycle event history H_t, this state yields the decision representation

$$b_t = \mathrm{Enc}_\theta(S_t, H_t), \tag{5}$$

where Enc_θ is the organizer's encoder, b_t is its observed prefix representation, and θ denotes the parameters shared by the encoder and assessor. In particular, H_t distinguishes an interaction that failed before delivery from one that arrived and was later replaced. Neither future states, evaluator labels, nor latent true dependency windows are available to the encoder.

Typed temporal candidates. The second module enumerates eligible typed interactions and their activation steps. Let $K \ge 1$ be the maximum explicit activation horizon, with $K = 1$ for the core organizer. Specializing the decision object in Eq. (1), each candidate and its descriptor are

$$q = (u, v, r, a) \in \mathcal{C}_t^{(K)}, \quad a \in \{t+1, \dots, \min(t+K, T)\}, \quad \phi(q) = \phi(u, v, r, a), \tag{6}$$

where u and v are the source and target agents, $r \in \mathcal{R}$ is the interaction type, and a is the activation step. The set $\mathcal{C}_t^{(K)}$ contains candidates that pass endpoint, prerequisite, and observed state checks. Its core specialization $\mathcal{C}_t^{(1)}$ is exactly the next step set $\mathcal{C}_t$ in Eq. (2). The feature vector $\phi(q)$ describes the four candidate fields. From b_t and $\phi(q)$, the assessor computes

$$\rho_\theta(q \mid b_t) = \sigma\Big(f_\theta^{\mathrm{ready}}(b_t, \phi(q))\Big), \qquad \widehat{\Delta U}_\theta(q \mid b_t) = f_\theta^{\mathrm{value}}(b_t, \phi(q)), \tag{7}$$

where σ is the logistic sigmoid, $f_\theta^{\mathrm{ready}}$ is the readiness head, and $f_\theta^{\mathrm{value}}$ is the value head. The quantity $\rho_\theta(q \mid b_t) \in [0, 1]$ estimates present readiness, while $\widehat{\Delta U}_\theta(q \mid b_t) \in \mathbb{R}$ estimates relative

value for ranking. The selection score is

$$s_\theta(q \mid b_t) = \rho_\theta(q \mid b_t)\widehat{\Delta U}_\theta(q \mid b_t) - \lambda_c c(q), \tag{8}$$

where $c(q) > 0$ is candidate cost and $\lambda_c \geq 0$ controls its penalty. The product assigns a high benefit term only when a candidate is both presently ready and estimated to be useful, while the cost penalty discourages spending on weak gains. Readiness and value remain separate because a correction from a verifier to a coder can be useful after a draft exists but unavailable before inspection, whereas an early planner handoff can be ready yet add little after the relevant coding step. These quantities are selection surrogates, not calibrated marginal utilities.

Feasible temporal schedule. The third module enforces budget, validity, and capacity before it compares candidate subsets. For any $A \subseteq \mathcal{C}_t^{(K)}$, expand the cost and load terms from Eq. (2) as

$$C(A) = \sum_{q \in A} c(q), \qquad W_a(A) = R_t(a) + \sum_{q \in A:\, a(q)=a} w(q), \tag{9}$$

where $C(A)$ is total admission cost, $w(q)$ is the capacity used by q, $a(q)$ is its activation step, and $R_t(a)$ is capacity already reserved at step a. Thus, $W_a(A)$ is the total capacity that would be committed at step a after admitting A. With remaining budget B_t and declared capacity L_a, the detailed feasible family is

$$\mathcal{F}_t^{(K)} = \left\{ A \subseteq \mathcal{C}_t^{(K)} : C(A) \leq B_t,\ \mathrm{Valid}(A) = 1, W_a(A) \leq L_a \text{ for every } a \in \{a(q) : q \in A\} \right\}. \tag{10}$$

The validity predicate retains the prerequisite, role, and endpoint safeguard from Eq. (2). Candidate enumeration screens individual interactions, while this predicate also excludes incompatible combinations within a selected set. The standalone core has $K = 1$ and no pending future reservation, so $\mathcal{F}_t^{(1)}$ recovers the family $\mathcal{F}_t$ in Eq. (2). TGO-MAS selects the temporal interaction set

$$A_t^\star = \arg\max_{A \in \mathcal{F}_t^{(K)}} \sum_{q \in A} s_\theta(q \mid b_t). \tag{11}$$

Thus, the selected typed interactions and activation steps in Figure 3 are precisely the members of $A_t^\star$. The reported solver enumerates feasible subsets and applies a deterministic identifier rule when estimated scores are equal. It optimizes only the displayed additive objective over $\mathcal{F}_t^{(K)}$. A deployment solver may approximate this optimization while retaining the same constraints.

Multi agent execution and feedback. The fourth module admits $A_t^\star$ and activates the members due at the next execution step while the agents continue their assigned planning, solving, tool use, verification, and integration work. TGO-MAS controls which information is delivered at the scheduled step. It neither replaces agent policies nor determines payload correctness. If o_t denotes the resulting delivery, failure, cancellation, completion, and resource use observations, state and history evolve as

$$(S_{t+1}, H_{t+1}) = \mathrm{Update}(S_t, H_t, o_t), \tag{12}$$

where Update applies the observed lifecycle transitions in order. The associated budget and reservation ledgers evolve as

$$(B_{t+1}, R_{t+1}) = \mathrm{Account}(B_t, R_t, A_t^\star, o_t), \tag{13}$$

where R_t is the capacity reservation map. The operator $\mathrm{Account}$ debits $C(A_t^\star)$ exactly once when a set is admitted and adds $w(q)$ to the reservation for each future activation step. Activation and terminal lifecycle transitions release the associated reservation according to the declared capacity policy. Under the default policy, cancellation releases unused reserved capacity but does not refund the admission charge. A retry or replacement receives a new identifier and undergoes a new admission decision. These rules make each budget change and reservation change traceable to one recorded transition. A successful verifier delivery may resolve a coder dependency and make an integration interaction ready, while a failed delivery leaves that dependency open. The next decision uses the updated prefix and repeats Eq. (4) through Eq. (11), closing the loop.

Table 3: Matched effectiveness and total system token use on HumanEval-heldout-124 and MMLU-validation-1531. Entries report means and standard deviations.

	HumanEval-heldout-124		MMLU-validation-1531	
Method	pass@1 ↑	Total tokens (k) ↓	Accuracy ↑	Total tokens (k) ↓
G-Designer	71.98 ± 0.62	$\underline{20.84 \pm 0.35}$	75.46 ± 0.72	$\underline{12.93 \pm 0.41}$
AgentPrune	72.85 ± 0.68	$\mathbf{20.53 \pm 0.42}$	76.01 ± 0.75	$\mathbf{12.74 \pm 0.38}$
CARD	73.59 ± 0.71	21.14 ± 0.39	76.48 ± 0.78	13.04 ± 0.45
GTD	73.94 ± 0.75	21.65 ± 0.48	76.69 ± 0.81	13.35 ± 0.49
DyTopo reimplementation	$\underline{74.45 \pm 0.82}$	21.85 ± 0.52	$\underline{76.94 \pm 0.85}$	13.54 ± 0.55
TGO-MAS	$\mathbf{76.64 \pm 0.85}$	20.93 ± 0.44	$\mathbf{78.39 \pm 0.88}$	12.94 ± 0.46

Advance commitments and realization. For interactions beyond $t + 1$, the optional $K > 1$ extension admits a contract recording its stable identifier, typed endpoints, activation step, capacity use, and cost. Equation (10) checks feasibility before $\mathrm{Account}$ charges the budget and reserves capacity. Cancellation releases unused capacity; retries, failures, and replacements append linked records rather than overwrite commitments. These ordered records reconstruct the pending queue and ledgers by Proposition 4. Provider, server, and evaluator logic remains external to the scheduler.

Theoretical support. Appendix G proves four structural properties: static projections can erase temporal distinctions; next-step organization contains static schedules and $K > 1$ adds feasible future actions; admission and atomic updates preserve nonnegative budgets and bounded reservations; and stable identifiers with ordered records reconstruct pending state. These results justify the temporal representation and ledger design, but do not guarantee task accuracy or learned utility.

6 Experiments

Datasets and tasks. We evaluate TGO-MAS on two tasks that follow established benchmarks. HumanEval (Chen et al., 2021) measures functional Python code generation on a held out split of 124 problems that is disjoint from the optimization split of 40 problems. MMLU (Hendrycks et al., 2021b) measures multiple choice knowledge and reasoning on all 1,531 validation records from 57 subjects. Synthetic dependency window tasks are used only for the timing diagnostics in RQ3 and do not replace the two primary benchmarks.

Baselines and controls. The matched comparison includes G-Designer, AgentPrune, CARD, and GTD as organizers defined at the task level or held fixed during execution, together with the DyTopo reimplementation that updates at each step as the closest dynamic baseline. The necessity study additionally evaluates no communication, random and complete graphs, a fixed workflow, a rule scheduler, and a compute matched individual agent. These controls distinguish the contributions of communication, connectivity, adaptation, and collaboration among agents.

Protocol and metrics. The controlled lane fixes the roster of five agents, model snapshot, prompts, tools, execution steps, call and token limits, edge caps, paired cells for each item and seed, stopping rule, and evaluator. Only the organization mechanism varies. HumanEval uses `pass@1`, MMLU uses normalized exact match accuracy, and total tokens measure resource use for the complete system. Failures remain in the denominator. We report means and standard deviations descriptively and keep reproductions that preserve author settings separate. The complete protocol appears in Appendix I.

6.1 Matched performance and component ablations

RQ1. Matched effectiveness and efficiency. Table 3 gives TGO-MAS the highest reported means at 76.64 on HumanEval and 78.39 on MMLU. They exceed the DyTopo reimplementation by 2.94% and 1.88% while using 4.21% and 4.43% fewer tokens. AgentPrune uses fewer tokens but has lower task means, so quality and resources require joint interpretation. The pattern supports selection based on execution state, while RQ2 and RQ3 examine relevant inputs without attributing the DyTopo difference to one factor.

Table 4: Component ablations of TGO-MAS. Entries report means and standard deviations.

Variant	HE pass@1 ↑	ΔHE (pp)	MMLU acc. ↑	ΔMMLU (pp)	Δtokens
TGO-MAS	76.64 ± 0.85	$+0.00$	78.39 ± 0.88	$+0.00$	$+0.0\%$
Without timestamps	74.15 ± 0.88	-2.49	77.03 ± 0.82	-1.36	$+3.4\%$
Stale state	74.86 ± 0.91	-1.78	77.31 ± 0.85	-1.08	$+0.8\%$
History removal	74.85 ± 0.93	-1.79	77.17 ± 0.86	-1.22	$+2.6\%$

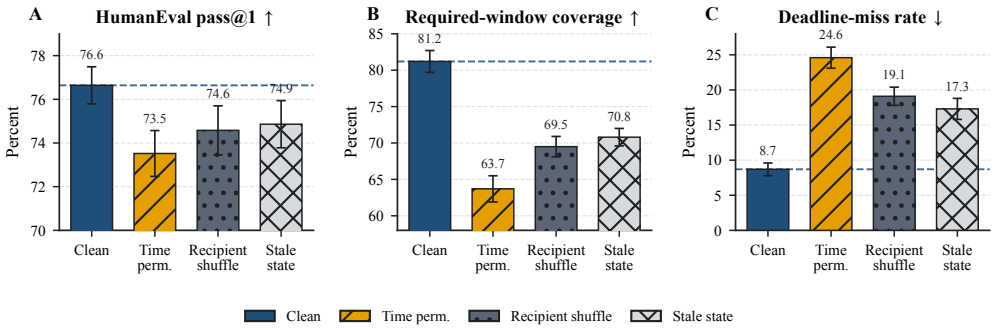


Figure 4: Mechanism interventions across complete executions. Bars report means with standard deviation whiskers. Dashed lines mark clean TGO-MAS for HumanEval `pass@1`, required window coverage, and deadline miss rate.

RQ2. Component necessity. Table 4 shows that every ablation lowers both task means and raises token use. Timestamp removal causes the largest drops, 2.49 points on HumanEval and 1.36 points on MMLU.

Timestamps locate dependency windows, while state and history distinguish arrived, failed, or obsolete inputs. Similar state and history losses support complementary replanning cues. These results concern tested inputs, not commitments over a longer horizon or reservation semantics.

6.2 Mechanism, Robustness, and Necessity

RQ3. Mechanism and boundary. Figure 4 shows that time permutation causes the largest changes, with HumanEval decreasing by 3.12 points, coverage by 17.5 points, and deadline misses increasing by 15.9 points. Because endpoints and interaction types remain fixed, the joint deterioration connects mistimed delivery to an end to end consequence. Recipient shuffle and stale state probe destination selection and current feedback, respectively, yielding smaller changes in the same direction. The timing intervention covers 91.2% of episodes in Table 12, supporting temporal placement as one mechanism without identifying its unique contribution to the clean gain.

RQ4. Robustness. TGO-MAS records silent agents, corrupted messages, and delays of one step before replanning from the observed state. Figure 5 shows higher HumanEval means and smaller degradation than the DyTopo reimplementation under all three matched perturbations. The degradation gaps are 1.94, 2.44, and 2.58 points, respectively. TGO-MAS retains from 94.2% to 96.2% of its clean score, while recovery ranges from 59.7% to 71.3%. Table 13 reports the complete results. They support resilience rather than immunity and do not isolate why DyTopo degrades more.

RQ5. Necessity controls. Figure 6 and Table 14 test no communication, arbitrary connectivity, fixed or rule based organization, and an individual agent with matched compute. TGO-MAS has the highest reported mean on both datasets without using the most tokens. The complete graph and compute matched individual agent are lower despite comparable or higher token use. The rule scheduler is closest but trails by 2.44 HumanEval points and 1.53 MMLU points. This ordering argues against communication volume or matched compute as sufficient explanations. Since each control changes an entire organization policy, the gap supports adaptation to observed state but cannot separate scheduling from agent specialization.

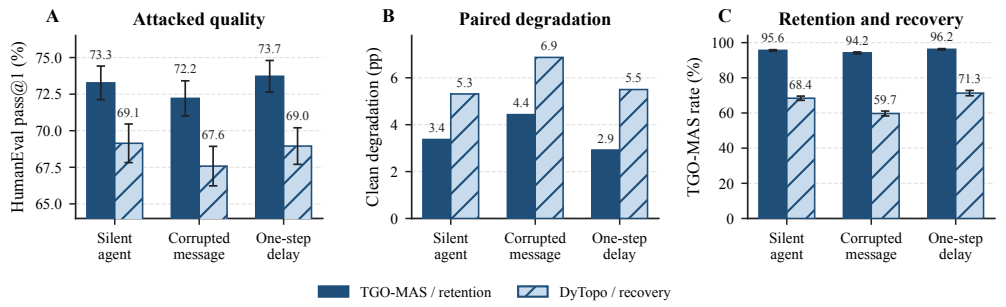


Figure 5: Robustness under silent agent, corrupted message, and delayed delivery perturbations. Panels A and B compare attacked quality and degradation relative to clean execution for TGO-MAS and DyTopo. Panel C reports TGO-MAS retention and recovery.

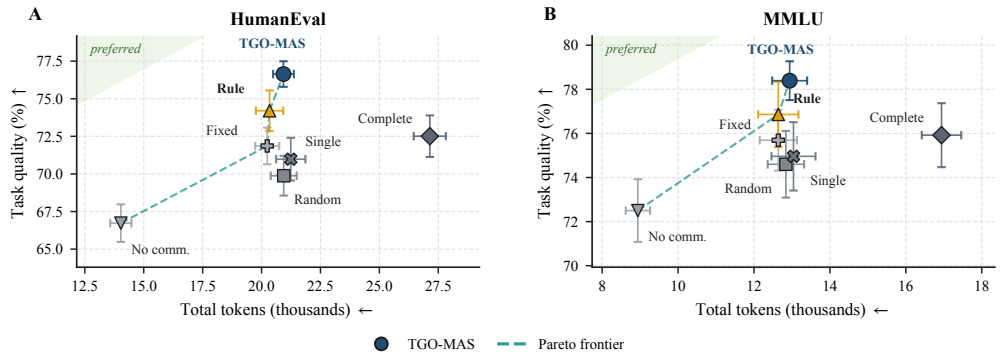


Figure 6: Necessity controls compare mean task quality with total tokens. Error bars denote standard deviations. Dashed curves connect nondominated points for quality and token use. The shaded corner indicates the preferred direction rather than a decision threshold.

RQ1 and RQ2 test matched outcomes and organizer inputs, while RQ3 through RQ5 test mechanism, robustness, and necessity.

Across the five questions, the results favor selective temporal organization over simply increasing communication or compute. TGO-MAS improves both benchmark means with fewer tokens than DyTopo, while the complete graph and compute matched individual agent do not reproduce that tradeoff. The larger changes under timestamp removal and time permutation further align benchmark quality with when interactions occur. The smaller recipient, state, and robustness effects show that timing operates together with destination choice and observed feedback.

Comparisons remain descriptive because the aggregate tables omit bootstrap intervals, paired win, tie, and loss summaries, and corrected tests. Controlled evaluations remain separate from reproductions that preserve author settings. The protocol in Appendix I leaves blinded trace diagnosis and state reconstruction unevaluated.

7 Conclusion

TGO-MAS treats interaction timing as an organizational variable: it selects typed interactions and activation steps from the observed prefix, separates readiness from value, enforces feasibility, and replans after feedback. Against DyTopo, its controlled means are 2.94% and 1.88% higher on HumanEval-heldout-124 and MMLU-validation-1531 with 4.21% and 4.43% fewer tokens; ablations, interventions, robustness tests, and controls provide descriptive support. Evidence is limited to five-agent runs, synthetic windows, and the next-step core, establishing neither significance, causality, universal necessity, nor broader superiority. Future work should test larger populations and longer horizons under matched resources and registered inference, and whether observable-prefix replanning and advance commitments justify their reservation and accounting costs.

Reproducibility Statement

We provide the implementation details of TGO-MAS in the main text and appendix. All experiments use publicly available datasets. Code and configurations of TGO-MAS will be released for reproduction.

AI Use Statement

Generative AI tools were not used in the preparation of this manuscript. All technical contents, method details, and experimental results were designed and verified by the authors.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C. Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *Advances in Neural Information Processing Systems*, volume 34, pp. 29304–29320, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/f514cec81cb148559cf475e7426eed5e-Abstract.html.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. Tarmac: Targeted multi-agent communication. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1538–1546, 2019. URL https://proceedings.mlr.press/v97/das19a.html.

Ziluo Ding, Tiejun Huang, and Zongqing Lu. Learning individually inferred communication for multi-agent cooperation. In *Advances in Neural Information Processing Systems*, volume 33, pp. 22069–22079, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/fb2fcd534b0ff3bbed73cc51df620323-Abstract.html.

Yali Du, Bo Liu, Vincent Moens, Ziqi Liu, Zhicheng Ren, Jun Wang, Xu Chen, and Haifeng Zhang. Learning correlated communication topology in multi-agent reinforcement learning. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 456–464, 2021. doi: 10.5555/3463952.3464010. URL https://www.ifaamas.org/Proceedings/aamas2021/pdfs/p456.pdf.

Bradley Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7 (1):1–26, 1979. doi: 10.1214/aos/1176344552. URL https://doi.org/10.1214/aos/1176344552.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Daoguang Zan, Shanghaoran Quan, Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. Omni-MATH: A universal olympiad level mathematic benchmark for large language models. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=yaqPf0KAlN.

Piotr Gmytrasiewicz. How to do things with words: A bayesian approach. *Journal of Artificial Intelligence Research*, 68:753–776, 2020. doi: 10.1613/jair.1.11951. URL https://doi.org/10.1613/jair.1.11951.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. doi: 10.1609/aaai.v32i1.11694. URL https://ojs.aaai.org/index.php/AAAI/article/view/11694.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. Measuring coding challenge competence with APPS. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021a. URL https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/hash/c24cd76e1ce41366a4bbe8a49b02a028-Abstract-round2.html.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

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, 2021c. URL https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979. URL https://www.jstor.org/stable/4615733.

Petter Holme and Jari Saramäki. Temporal networks. *Physics Reports*, 519(3):97–125, 2012. doi: 10.1016/j.physrep.2012.03.001. URL https://doi.org/10.1016/j.physrep.2012.03.001.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. Metagpt: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VtmBAGCN7o.

Guangzheng Hu, Yuanheng Zhu, Dongbin Zhao, Mengchen Zhao, and Jianye Hao. Event-triggered communication network with limited-bandwidth constraint for multi-agent reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, 34(8):3966–3978, 2023. doi: 10.1109/TNNLS.2021.3121546. URL https://doi.org/10.1109/TNNLS.2021.3121546.

Shengchao Hu, Li Shen, Ya Zhang, and Dacheng Tao. Learning multi-agent communication from graph modeling perspective. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Qox9rO0kN0.

Eric Hanchen Jiang, Levina Li, Frank Wan, Xiao Liang, Sophia Yin, Yuchen Wu, Xinfeng Li, Yizhou Sun, Wei Wang, Kai-Wei Chang, and Ying Nian Wu. Dynamic generation of multi llm agents communication topologies with graph diffusion models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, pp. 38042–38060, 2026. doi: 10.18653/v1/2026.acl-long.1764. URL https://aclanthology.org/2026.acl-long.1764/.

Jiechuan Jiang and Zongqing Lu. Learning attentional communication for multi-agent cooperation. In *Advances in Neural Information Processing Systems*, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/6a8018b3a00b69c008601b8becae392b-Abstract.html.

Anirudh Kakarlapudi, Gayathri Anil, Adam Eck, Prashant Doshi, and Leen-Kiat Soh. Communication and coordination in bandwidth-limited multiagent systems. In *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence*, volume 180 of *Proceedings of Machine Learning Research*, pp. 938–948, 2022. URL https://proceedings.mlr.press/v180/kakarlapudi22a.html.

Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akhil Sethi, Peter Forsyth, and Pascal Poupart. Representation learning for dynamic graphs: A survey. *Journal of Machine Learning Research*, 21(70):1–73, 2020. URL https://jmlr.org/papers/v21/19-447.html.

Lukas Kesper, Sebastian Trimpe, and Dominik Baumann. Learning event-triggered control from data through joint optimization. In *Proceedings of the 5th Annual Learning for Dynamics and Control Conference*, volume 211 of *Proceedings of Machine Learning Research*, pp. 1072–1085, 2023. URL https://proceedings.mlr.press/v211/kesper23a.html.

Daewoo Kim, Sangwoo Moon, David Hostallero, Wan Ju Kang, Taeyoung Lee, Kyunghwan Son, and Yung Yi. Learning to schedule communication in multi-agent reinforcement learning. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HUAnBToP_a.

Srijan Kumar, Xikun Zhang, and Jure Leskovec. JODIE: Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1269–1278, 2019. doi: 10.1145/3292500.3330895. URL https://dblp.org/rec/conf/kdd/KumarZL19.

Timothy Lebo, Satya Sahoo, and Deborah McGuinness. Prov-o: The prov ontology. W3c recommendation, World Wide Web Consortium, 2013. URL https://www.w3.org/TR/prov-o/.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for "mind" exploration of large language model society. In *Advances in Neural Information Processing Systems*, volume 36, pp. 51991–52008, 2023. doi: 10.52202/075280-2264. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/a3621ee907def47c1b952ade25c67698-Abstract-Conference.html.

Shiyuan Li, Yixin Liu, Qingsong Wen, Chengqi Zhang, and Shirui Pan. Assemble your crew: Automatic multi-agent communication topology design via autoregressive graph generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 23142–23150, 2026a. doi: 10.1609/aaai.v40i28.39481. URL https://ojs.aaai.org/index.php/AAAI/article/view/39481.

Shiyuan Li, Yixin Liu, Yu Zheng, Mei Li, Quoc Viet Hung Nguyen, and Shirui Pan. OFA-MAS: One-for-all multi-agent system topology design based on mixture-of-experts graph generative models. In *Proceedings of the ACM Web Conference 2026*, pp. 1333–1344, 2026b. doi: 10.1145/3774904.3792537. URL https://dblp.org/rec/conf/www/LiLZLNP26.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/aca97732e30bcf1303bc22ac3924fd16-Abstract-Conference.html.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pp. 158–167, 2017. doi: 10.18653/v1/P17-1015. URL https://aclanthology.org/P17-1015/.

Yang Liu, Jinxuan Cai, Yishen Li, Qi Meng, Zedi Liu, Xin Li, Chen Qian, Chuan Shi, and Cheng Yang. MASFactory: A graph-centric framework for orchestrating LLM-based multi-agent systems with vibe graphing. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pp. 357–366, 2026. doi: 10.18653/v1/2026.acl-demo.35. URL https://aclanthology.org/2026.acl-demo.35/.

Zeyang Liu, Lipeng Wan, Xue Sui, Zhuoran Chen, Kewu Sun, and Xuguang Lan. Deep hierarchical communication graph in multi-agent reinforcement learning. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pp. 208–216, 2023. doi: 10.24963/ijcai.2023/24. URL https://www.ijcai.org/proceedings/2023/24.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. A dynamic llm-powered agent network for task-oriented agent collaboration. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=XII0Wp1XA9.

Yuxing Lu, Yucheng Hu, Xukai Zhao, and Jiuxin Cao. Dytopo: Dynamic topology routing for multi-agent reasoning via semantic matching, 2026. URL https://arxiv.org/abs/2602.06039. Preprint.

Andrew Mellor. The temporal event graph. *Journal of Complex Networks*, 6(4):639–659, 2018. doi: 10.1093/comnet/cnx048. URL https://dblp.org/rec/journals/compnet/Mellor18.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2080–2094, 2021. doi: 10.18653/v1/2021.naacl-main.168. URL https://aclanthology.org/2021.naacl-main.168/.

Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs, 2020. URL https://arxiv.org/abs/2006.10637. Preprint.

Maayan Roth, Reid Simmons, and Manuela Veloso. What to communicate? execution-time decision in multi-agent POMDPs. In *Distributed Autonomous Robotic Systems 7*, pp. 177–186. Springer Japan, 2006. doi: 10.1007/4-431-35881-1_18. URL https://doi.org/10.1007/4-431-35881-1_18.

Subhro Roy and Dan Roth. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1743–1752, 2015. doi: 10.18653/v1/D15-1202. URL https://aclanthology.org/D15-1202/.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 2186–2188, 2019. URL https://www.ifaamas.org/Proceedings/aamas2019/pdfs/p2186.pdf.

Jari Saramäki, Mikko Kivelä, and Márton Karsai. Weighted temporal event graphs. In *Temporal Network Theory*, pp. 107–128. Springer International Publishing, 2019. doi: 10.1007/978-3-030-23495-9_6. URL https://doi.org/10.1007/978-3-030-23495-9_6.

Junjie Sheng, Xiangfeng Wang, Bo Jin, Junchi Yan, Wenhao Li, Tsung-Hui Chang, Jun Wang, and Hongyuan Zha. Learning structured communication for multi-agent reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 36(2):50, 2022. doi: 10.1007/s10458-022-09580-8. URL https://doi.org/10.1007/s10458-022-09580-8.

Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. Learning when to communicate at scale in multiagent cooperative and competitive tasks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rye7knCqK7.

Shoucheng Song, Youfang Lin, Sheng Han, Chang Yao, Hao Wu, Shuo Wang, and Kai Lv. CoDe: Communication delay-adaptive multi-agent collaboration via dual alignment of intent and timeliness. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 23304–23312, 2025. doi: 10.1609/aaai.v39i22.34497. URL https://ojs.aaai.org/index.php/AAAI/article/view/34497.

Haochen Sun, Shuwen Zhang, Lujie Niu, Lei Ren, Hao Xu, Hao Fu, Fangkun Zhao, Caixia Yuan, and Xiaojie Wang. Collab-overcooked: Benchmarking and evaluating large language models as collaborative agents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 4922–4951, 2025. doi: 10.18653/v1/2025.emnlp-main.249. URL https://aclanthology.org/2025.emnlp-main.249/.

Congjia Tian, Yuhang Yao, and Jiaming Cui. Queenbee planner: Skill-evolving communication topologies for token-efficient llm multi-agent systems, 2026. URL https://arxiv.org/abs/2606.27492. Preprint.

Qi Tian, Kun Kuang, Baoxiang Wang, Furui Liu, and Fei Wu. CGIBNet: Compact graph information bottleneck for multi-agent communication, 2021. URL https://arxiv.org/abs/2112.10374. Preprint.

Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. DyRep: Learning representations over dynamic graphs. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyePrhR5KX.

Yuanfei Wang, Fangwei Zhong, Jing Xu, and Yizhou Wang. ToM2C: Target-oriented multi-agent communication and cooperation with theory of mind. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=2t7CkQXNpuq.

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 conversations. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=BAakY1hNKS.

Tongtong Wu, Yanming Li, Ziye Tang, Chen Jiang, Linhao Luo, Guilin Qi, Shirui Pan, and Gholamreza Haffari. Card: Towards conditional design of multi-agent topological structures. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=JgvJdICc6P.

Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Inductive representation learning on temporal graphs. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rJeW1yHYwH.

Tingting Yuan, Hwei-Ming Chung, Jie Yuan, and Xiaoming Fu. DACOM: Learning delay-aware communications for multi-agent reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 11763–11771, 2023. doi: 10.1609/aaai.v37i10.26389. URL https://ojs.aaai.org/index.php/AAAI/article/view/26389.

Sukwon Yun, Jie Peng, Pingzhi Li, Wendong Fan, Jie Chen, James Zou, Guohao Li, and Tianlong Chen. Graph-of-agents: A unified framework for dynamic multi-agent system construction and orchestration, 2026. URL https://arxiv.org/abs/2604.17148. Preprint.

Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, Lei Bai, and Xiang Wang. Multi-agent architecture search via agentic supernet. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 75834–75852, 2025a. URL https://proceedings.mlr.press/v267/zhang25bi.html.

Guibin Zhang, Yanwei Yue, Zhixun Li, Sukwon Yun, Guancheng Wan, Kun Wang, Dawei Cheng, Jeffrey Xu Yu, and Tianlong Chen. Cut the crap: An economical communication pipeline for llm-based multi-agent systems. In *International Conference on Learning Representations*, 2025b. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/bbc461518c59a2a8d64e70e2c38c4a0e-Abstract-Conference.html.

Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang, Tianlong Chen, and Dawei Cheng. G-designer: Architecting multi-agent communication topologies via graph neural networks. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 76678–76692, 2025c. URL https://proceedings.mlr.press/v267/zhang25cu.html.

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 *International Conference on Learning Representations*, 2025d. URL https://openreview.net/forum?id=z5uVAKwmjf.

Jintian Zhang, Xin Xu, Ningyu Zhang, Ruibo Liu, Bryan Hooi, and Shumin Deng. Exploring collaboration mechanisms for LLM agents: A social psychology view. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 14544–14607, 2024. doi: 10.18653/v1/2024.acl-long.782. URL https://aclanthology.org/2024.acl-long.782/.

Yue Zhao. Grade: Graph representation of llm agent dependency and execution, 2026. URL https://arxiv.org/abs/2606.22741. Preprint.

Changxi Zhu, Mehdi Dastani, and Shihan Wang. A survey of multi-agent deep reinforcement learning with communication. *Autonomous Agents and Multi-Agent Systems*, 38(1):4, 2024. doi: 10.1007/s10458-023-09633-6. URL https://doi.org/10.1007/s10458-023-09633-6.

Kunlun Zhu, Hongyi Du, Zhaochen Hong, Xiaocheng Yang, Shuyi Guo, Zhe Wang, Zhenhailong Wang, Cheng Qian, Xiangru Tang, Heng Ji, and Jiaxuan You. Multiagentbench: Evaluating the collaboration and competition of llm agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pp. 8580–8622, 2025. doi: 10.18653/v1/2025.acl-long.421. URL https://aclanthology.org/2025.acl-long.421/.

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, 2024. URL https://proceedings.mlr.press/v235/zhuge24a.html.

A Appendix Roadmap and Notation

This appendix expands the compact presentation of TGO-MAS in the main text. Appendix B situates the framework within research on multi-agent workflows, communication control, temporal graphs, and evaluation; Appendices C and D provide the extended notation and the details of the motivating empirical analyses; and Appendices E and F specify the method components and closed-loop scheduling procedure. Appendices G and H present the structural results and their complete proofs, while Appendices I and J document the evaluation protocol and the full numerical results and diagnostics. Finally, Appendix K records the reproducibility artifacts, implementation status, and conformance boundaries of the supplied source package. This organization separates the conceptual and formal foundations from the method specification, theoretical justification, empirical evaluation, and reproducibility record.

Table 5: Core notation used by TGO-MAS. The temporal-graph symbols (V, $\mathcal{R}$, $\mathcal{E}$, T, Σ) describe the persistent interaction structure of the multi-agent system, whereas runtime quantities (E_t, S_t, H_t, $\mathcal{Q}_t$, B_t, R_t) change step by step as agents execute and contracts pend or resolve.

Symbol	Meaning
$V, \mathcal{R}, \mathcal{E}$	agents, interaction types, potential typed edges
T, Σ, E_t	horizon, temporal schedule, active set at step t
S_t, H_t, b_t	observable MAS state, lifecycle history, and prefix representation
$A_t, \mathcal{Q}_t$	selected core actions and optional pending contracts
K	maximum explicit contract horizon ($K = 1$ for the core)
$B_t, c(q)$	remaining budget and contract cost
$L_\tau, w(q), R_t(\tau)$	capacity, weight, and reserved capacity

B Extended Related Work

Section 2 keeps only the systems nearest to the temporal scheduling interface; the paragraphs below expand that discussion along seven threads: workflow and graph organization; the closest temporal precedents, DyTopo and QueenBee; communication control in MARL together with temporal-graph representations; additional graph and coordination mechanisms; complementary communication-control perspectives; temporal-network, provenance, and evaluation foundations; and the benchmarks and statistical protocol that shape the evaluation discipline of Section 6.

Workflow and graph organization. Systems organized by roles, executable workflows, and graph construction methods all encode useful portions of an MAS control policy (Wu et al., 2024; Hong et al., 2024; Li et al., 2023; Zhuge et al., 2024; Zhang et al., 2025d;c;b; Jiang et al., 2026). Their differences concern when topology is chosen, which execution state it can inspect, and whether delivery timing is independently represented. A topology chosen before an episode, a frozen spatial and temporal mask, and a graph tailored to each task are not equivalent to a schedule for the next step that responds to execution, but neither interface is automatically superior.

Closest temporal precedents. DyTopo reconstructs sparse directed graphs at reasoning rounds from manager goals and agent need/offer descriptions (Lu et al., 2026). QueenBee creates temporal communication DAGs and updates reusable rules from execution records (Tian et al., 2026). They are direct precedents for temporal organization in LLM MASs. TGO-MAS is positioned only by its explicit decision interface. Candidates that pair typed events with steps are scored from observable execution state, selected under feasibility constraints, and revised after feedback. Advance commitment, reservation accounting, and lifecycle transitions are optional extensions of this interface. This distinction at the formulation level is not a claim of firstness.

Communication control and temporal graphs. MARL communication research studies communication necessity, recipient selection, gating, and constrained medium access (Jiang & Lu, 2018; Singh et al., 2019; Hu et al., 2023; Kim et al., 2019; Das et al., 2019; Ding et al., 2020; Liu et al., 2023). Temporal graph work represents relations as events indexed by time (Rossi et al., 2020; Trivedi et al., 2019; Kazemi et al., 2020). TGO-MAS adopts these ideas as representation and scheduling

vocabulary, while its claims about task performance remain subject to the controlled experiment protocol.

Additional graph and coordination mechanisms. Graph-of-Agents and MASFactory make graph structure a reusable orchestration interface (Yun et al., 2026; Liu et al., 2026). MAAS automates the design of systems with multiple agents (Zhang et al., 2025a). ArgDesigner, OFA-MAS, and GRADE explore autoregressive design, mixtures of experts, and execution graphs that model dependencies (Li et al., 2026a;b; Zhao, 2026). DyLAN dynamically selects agents, CommFormer jointly learns communication graphs and architecture, and CARD conditions topology design on context (Liu et al., 2024; Hu et al., 2024; Wu et al., 2026). Social psychology analyses complement these methods by examining how collaboration structures affect interaction among LLM agents (Zhang et al., 2024).

Complementary communication control perspectives. Correlated communication, recipient modeling, graph information bottlenecks, and structured communication each constrain different aspects of message exchange (Du et al., 2021; Wang et al., 2022; Tian et al., 2021; Sheng et al., 2022). Formulations for constrained media, event triggers, and delays additionally distinguish who may communicate, when an update is warranted, and when a message remains timely (Kakarlapudi et al., 2022; Kesper et al., 2023; Yuan et al., 2023; Song et al., 2025). Decision theoretic accounts of communication and message reasoning further ground policies for recipients (Roth et al., 2006; Gmytrasiewicz, 2020). A recent survey consolidates the broader MARL communication landscape (Zhu et al., 2024).

Temporal, provenance, and evaluation foundations. Temporal network theory, temporal event graphs, and weighted temporal event graphs formalize the ordering and timing of interactions (Holme & Saramäki, 2012; Mellor, 2018; Saramäki et al., 2019). JODIE and TGAT complement DyRep and TGN with recurrent and attentional representations of evolving interactions (Kumar et al., 2019; Xu et al., 2020). PROV-O provides a standard vocabulary for entities, activities, agents, and derivations in execution records (Lebo et al., 2013).

Benchmarks and statistical protocol. SMAC, MultiAgentBench, and Collab-Overcooked cover cooperative MARL and collaboration among LLM agents (Samvelyan et al., 2019; Zhu et al., 2025; Sun et al., 2025). HumanEval, MMLU, GSM8K, MATH, and APPS provide established programming, knowledge, and reasoning evaluations (Chen et al., 2021; Hendrycks et al., 2021b; Cobbe et al., 2021; Hendrycks et al., 2021c;a). SVAMP, MultiArith, AQuA, process verification data, and Omni-MATH broaden mathematical reasoning coverage (Patel et al., 2021; Roy & Roth, 2015; Ling et al., 2017; Lightman et al., 2024; Gao et al., 2025). Reproducible reinforcement learning evaluation and reliable aggregate comparison motivate the reporting discipline (Henderson et al., 2018; Agarwal et al., 2021). Holm's correction and the bootstrap support familywise comparison and interval estimation, respectively (Holm, 1979; Efron, 1979).

C Extended Formalization and Notation

The following identities extend the compact notation of Section 3 and Table 5 to realized events, observable state, and reserved capacity. Equation (6) defines the core candidate $q = (u, v, r, a)$, with $a = t + 1$ for scheduling at the next step. The optional commitment extension augments this candidate to $\bar{q} = (\iota, u, v, r, a, w, c)$, where ι is an identifier, w is capacity use, and c is an episode cost. A realized event can be represented as

$$e_k = (\iota_k, u_k, v_k, r_k, p_k, a_k, d_k, z_k, x_k), \tag{14}$$

where ι_k is a unique identifier. The quantities p_k, a_k, and d_k are the proposal step, activation step, and step at which delivery becomes available. The quantity z_k is lifecycle status, and x_k is a payload reference, hash, or safe summary. Scheduling, delivery, and completion are distinct transitions so that a commitment is not mistaken for a successful payload.

Recalling Eq. (4), the observable state has the form

$$S_t = (P_t, Z_t, D_t, M_{\le t}, O_{\le t}, B_t, L_{t:T}), \tag{15}$$

where P_t is the task phase, Z_t contains available agent outputs, D_t records unresolved dependencies, and $M_{\le t}$ and $O_{\le t}$ are the histories of delivered messages and tool observations. The remaining

budget is B_t, while $L_{t:T}$ gives declared capacities from the current step through the horizon. As recapped from Eq. (5), the encoder may use only this observed prefix and ordered lifecycle events H_t to produce $b_t = \mathrm{Enc}_\theta(S_t, H_t)$. Before decision t, pending contracts in the optional extension define

$$R_t(\tau) = \sum_{q \in \mathcal{Q}_t : a(q) = \tau} w(q). \tag{16}$$

For the optional ledger semantics, acceptance is charged exactly once. Under the default semantics, cancellation does not refund a charge, while retry or rescheduling creates a linked contract with a new identifier and charge. Alternative refund policies require explicit ledger transitions and a correspondingly revised feasibility invariant. Persistent channels, dynamic agent creation, and asynchronous wall clock scheduling are outside the present formulation.

D Empirical Studies Details

Section 4 motivates the temporal interface with three registered studies; the paragraphs below record the construction shared by all three and then the invariants and accounting of Study A (Table 1), Study B (Figure 2 and Table 10), and Study C (Tables 2 and 11).

Controlled construction. The three studies use registered dependency windows and interventions that change one aspect of an interaction schedule or history view. A required interaction contributes to coverage only when its source, target, type, and delivery step satisfy the registered window. The supplied aggregates do not include item-level records, denominators, or dispersion estimates, so all comparisons are descriptive.

Study A invariants and results. Time permutation changes delivery time while preserving the directed typed interactions and their counts. Recipient shuffle preserves source, type, activation step, interaction count, and budget but changes targets. Table 1 reports all supplied clean and intervened coverage values. Recipient shuffling causes the larger decrease for every organizer, whereas AgentPrune's wide window attenuates the decrease under time permutation.

Study B accounting. Signed temporal shifts preserve the organizer and registered window while moving activation by $\delta \in \{-2, -1, 0, +1, +2\}$. Table 10 gives the exact values underlying Figure 2. The five shift settings are discrete interventions, and the connecting lines do not define intermediate schedules. The wider window retains more coverage at every nonzero shift for every organizer.

Study C metrics. The comparison reports next-edge success, redundant-edge rate, and pairwise ranking accuracy as separate outcomes. Table 11 reproduces the supplied aggregate values without adding dispersion estimates. Activating both candidates illustrates why success alone is insufficient: it reaches 100.00% success with 50.76% redundancy, while pairwise ranking accuracy is not defined for that control. Structured lifecycle history provides the strongest joint trade-off among the evaluated views.

E TGO-MAS Method Details

This appendix makes the core/extension boundary operational. The core organizer receives only the observed state and history, produces an action for the next step, and replans after execution. Recalling S_t, H_t, b_t, and q from Eqs. (4)–(6), a compatible parameterization of the encoder and assessor is

$$\begin{aligned} b_t &= \mathrm{Enc}_\theta(S_t, H_t), \\ (\rho_{q,t}, \widehat{\Delta U}_{q,t}) &= f_\theta(b_t, \phi(q)), \qquad q \in \mathcal{C}_t^{(1)}, \end{aligned} \tag{17}$$

Here $\rho_{q,t} \equiv \rho_\theta(q \mid b_t)$ and $\widehat{\Delta U}_{q,t} \equiv \widehat{\Delta U}_\theta(q \mid b_t)$ are compact aliases for the readiness and value estimates in Eq. (7). The descriptor $\phi(q)$ follows Eq. (6). An organizer governed by rules may replace f_θ with transparent eligibility and priority functions. In either case, scores are selection surrogates, not calibrated marginal utility estimates.

Algorithm 1 Core TGO-MAS with observed prefix scheduling for the next step and replanning

Require: horizon T, episode budget B, capacities for the next steps $L_{1:T}$

Initialize observed history $H_0 \leftarrow \emptyset$ and state S_0
for $t = 0$ to $T - 1$ **do**
  Form $b_t \leftarrow \mathrm{Enc}_\theta(S_t, H_t)$ from observed information only
  Enumerate candidates $q = (u, v, r, t + 1)$ and apply hard eligibility checks
  Estimate $\rho_\theta(q \mid b_t)$ and $\widehat{\Delta U}_\theta(q \mid b_t)$
  Select $A_t \in \mathcal{F}_t$ by Eq. (11)
  Activate A_t and observe deliveries, failures, completions, and resource use o_t
  Update $S_{t+1}, H_{t+1} \leftarrow \mathrm{Update}(S_t, H_t, o_t)$
end for
return final task output and the observed decision record

Learning and calibration. If registered window labels are available in a controlled environment, they can supervise a readiness head. If episode outcomes are available, they can train a ranking or value surrogate. Neither route identifies a universal dependency window or establishes transfer across environments. Deployment must retain the restriction against anticipation. Labels used for offline supervision cannot be exposed as online features.

Advance commitment extension. To schedule beyond $t + 1$, replace the candidate horizon in Algorithm 1 with $a \in \{t + 1, \ldots, \min(t + K, T)\}$. Atomically assign an identifier, debit the episode budget, reserve $R_t(a)$, and insert the accepted contract into $\mathcal{Q}_t$. Cancellation, retry, failure, and rescheduling must then be explicit state transitions. These extra operations are required for Proposition 3 and Proposition 4, but not for the core feedback loop.

Implementation status. Algorithm 2 specifies typed candidates, an observed prefix state encoder, a readiness/value estimator interface, additive surrogate selection, feedback updates, an optional delayed contract queue, debit at admission, future capacity reservation, lifecycle transitions, cancellation without refunds, and schedule state reconstruction. The source paper package does not include the previously referenced scheduler repository, so conformance of an executable implementation to this specification is not independently verifiable here. Provider, server, and evaluator integration likewise remain responsibilities of the execution layer outside the present specification.

F Closed Loop Scheduling Algorithm

Algorithm 2 turns the method equations of Section 5 (Equations (4)–(13)) into an executable reference procedure. The core specification uses $K = 1$. The branches marked "optional" apply only to advance commitments. The inputs to Enc_θ contain an observed execution prefix only, so the procedure cannot consume evaluator labels, true dependency windows, or future execution outcomes.

The exact subset solver uses a deterministic identifier rule to break ties when equal surrogate values are feasible. Under the default policy, cancellation releases a future reservation but does not refund budget charged at admission. Replay starts from B_0, $L_{0:T}$, and an empty ledger, then processes the same ordered acceptance and lifecycle records. It reconstructs the pending queue, reservations, and due set when the trace satisfies Proposition 4.

G Theoretical Analysis

The four results below explain why the representation and accounting choices in TGO-MAS are structurally appropriate. The complete proofs appear in Appendix H. Proposition 1 shows that discarding activation steps erases distinctions that the typed temporal candidate in Eq. (6) is designed to preserve. Proposition 2 places the core for the next step and the advance commitment extension in one action hierarchy, clarifying what the horizon parameter K adds to the scheduling design. Proposition 3 connects the admission test and ledger update in Eqs. (10) and (13) to nonnegative budget and bounded capacity reservations. Proposition 4 connects stable identifiers and ordered lifecycle transitions to the state reconstruction mechanism used by the optional ledger.

Algorithm 2 TGO-MAS closed loop scheduling with optional $K > 1$ commitments

Require: initial budget B_0, declared capacities $L_{0:T}$, commitment horizon $K \geq 1$, encoder Enc_θ, assessor $(\rho_\theta, \widehat{\Delta U}_\theta)$

```
Initialize empty lifecycle history H_0 and contract ledger Q_0
for t = 0 to T − 1 do
    Receive only observed state S_t and observed execution feedback o_{t−1}
    Append valid activation, delivery, completion, failure, and cancellation transitions from o_{t−1} to H_t
    Update pending contracts, remaining budget B_t, and reservations R_t from Q_t
    b_t ← Enc_θ(S_t, H_t)
    C_t ← {q = (u, v, r, a) : u, v ∈ V, t + 1 ≤ a ≤ min(t + K, T), Eligible(q, b_t)}
    for each q ∈ C_t do
        ρ_q ← ρ_θ(q | b_t) and ΔÛ_q ← ΔÛ_θ(q | b_t)
        s_q ← ρ_q ΔÛ_q − λ_c c(q)
    end for
    A*_t ← arg max_{A⊆C_t} Σ_{q∈A} s_q subject to Eq. (11)
    Atomically assign identifiers, debit Σ_{q∈A*_t} c(q), and record acceptance events
    if K > 1 then
        Reserve w(q) at each future activation step a(q) and insert contracts into Q_t
    end if
    Activate interactions due at step t + 1, whose realized outcomes form the next o_t
end for
return final task output, admitted schedules, and ordered lifecycle trace
```

For a schedule $\Sigma = (E_1, \ldots, E_T)$, define the topology and multiset projections that discard activation steps:

$$P_{\mathrm{top}}(\Sigma) = \bigcup_{t=1}^{T} E_t, \qquad P_{\mathrm{multi}}(\Sigma) = \biguplus_{t=1}^{T} E_t. \tag{18}$$

Here E_t is the set of typed edges active at step t. The union records edge presence, while the multiset union also records repeated occurrence but not its step.

Proposition 1 (Loss of injectivity)**.** *For $T \geq 2$ and nonempty space $\mathcal{E}$ of typed edges, neither projection in Eq.* (18) *is injective.*

Lemma 1 (Temporal shift witness)**.** *For any $e \in \mathcal{E}$ and distinct steps i, j, the schedules containing $\{e\}$ only at i and only at j have identical topology and multiset projections.*

Proof sketch of Proposition 1 and Lemma 1. The two schedules are unequal because their active sets differ at steps i and j. Each projection nevertheless contains exactly one occurrence of e, so Lemma 1 supplies a collision for both maps.

Proposition 2 (Containment)**.** *Every static schedule is representable by an organizer for the next step. Moreover, whenever a feasible activation in $\{t + 2, \ldots, \min(t + K, T)\}$ exists, the $K = 1$ contract action space is strictly contained in that for $K > 1$.*

Lemma 2 (Action embedding)**.** *A fixed edge set G is realized by choosing $A_t = G$ at every step, and every feasible $K = 1$ contract remains available when the admissible horizon is enlarged to $K > 1$.*

Proof sketch of Proposition 2 and Lemma 2. Repetition gives the static realization in Lemma 2. Enlarging K preserves all actions for the next step and adds the assumed feasible action with activation $a > t + 1$, which establishes strict containment without asserting any utility improvement.

Proposition 3 (Dynamic feasibility)**.** *Assume positive costs, a single debit upon admission, atomic queue and reservation updates, and reapplication of the admission test after every replan. Then the remaining budget is nonnegative and reserved capacity at every future step a is at most L_a.*

Lemma 3 (Single step invariant preservation)**.** *If $B_t \geq 0$ and $R_t(a) \leq L_a$ before admission, then admitting any $A_t \in \mathcal{F}_t$ and atomically applying the specified debit and reservation update preserves both inequalities.*

Proof sketch of Proposition 3 and Lemma 3. Membership in $\mathcal{F}_t$ bounds the new debit by B_t and each reservation after admission by L_a. Lemma 3 therefore applies at every decision, and induction from the initialized ledger proves both invariants.

Proposition 4 (Schedule state reconstruction)**.** *Given an empty initial queue and a complete, totally ordered event trace with unique contract identifiers, the pending queue, reservations, and due edge set are uniquely reconstructible.*

Lemma 4 (Deterministic trace transition)**.** *For a reconstructed ledger state and the next valid uniquely identified record, the acceptance or terminal transition semantics determine at most one successor ledger state.*

Proof sketch of Proposition 4 and Lemma 4. Starting from the empty queue, apply Lemma 4 to each record in trace order. Induction gives a unique ledger after every prefix, from which the activation steps uniquely determine the due edge set.

These results have deliberately bounded scope. Proposition 1 establishes representation loss rather than timing utility, and Proposition 2 compares action spaces rather than task outcomes or stateful trajectories. Propositions 3 and 4 require their stated accounting and trace assumptions and address schedule state rather than payload semantics, diagnostic value, or task quality. Likewise, Eq. (11) optimizes only its additive selection estimate. The results thus support the method's representation and ledger design without extending to empirical performance claims.

H Complete Theoretical Proofs

This section proves the four pairs of propositions and lemmas in Section G. Recall that $P_{\mathrm{top}}(\Sigma) = \bigcup_{t=1}^{T} E_t$ and $P_{\mathrm{multi}}(\Sigma) = \biguplus_{t=1}^{T} E_t$.

Lemma 1 (restated). *For any $e \in \mathcal{E}$ and distinct steps i, j, the schedules containing $\{e\}$ only at i and only at j have identical topology and multiset projections.*

Proof of Lemma 1. Choose $e \in \mathcal{E}$ and distinct $i, j \in \{1, \dots, T\}$. Define $\Sigma^{(i)} = (E_1^{(i)}, \dots, E_T^{(i)})$ by $E_i^{(i)} = \{e\}$ and $E_t^{(i)} = \varnothing$ for $t \neq i$. Define $\Sigma^{(j)}$ analogously. Because $i \neq j$, $\Sigma^{(i)} \neq \Sigma^{(j)}$. However,

$$P_{\mathrm{top}}(\Sigma^{(i)}) = P_{\mathrm{top}}(\Sigma^{(j)}) = \{e\}, \qquad P_{\mathrm{multi}}(\Sigma^{(i)}) = P_{\mathrm{multi}}(\Sigma^{(j)}) = \{\{e\}\},$$

where $\{\{e\}\}$ is the singleton multiset. This proves Lemma 1. □

Proposition 1 (restated). *For $T \geq 2$ and nonempty space $\mathcal{E}$ of typed edges, neither projection in Eq.* (18) *is injective.*

Proof of Proposition 1. The collision constructed in Lemma 1 proves that neither projection is injective. □

Lemma 2 (restated). *A fixed edge set G is realized by choosing $A_t = G$ at every step, and every feasible $K = 1$ contract remains available when the admissible horizon is enlarged to $K > 1$.*

Proof of Lemma 2. Let a static organization be specified by a fixed edge set G. An organizer for the next step that chooses $A_t = G$ at every decision realizes the same active set at every execution step, proving the first part of the lemma. A $K = 1$ action has activation $a = t + 1$, which belongs to $\{t + 1, \dots, \min(t + K, T)\}$ for every $K > 1$. Hence every feasible $K = 1$ action is also available under the larger horizon. □

Proposition 2 (restated). *Every static schedule is representable by an organizer for the next step. Moreover, whenever a feasible activation in $\{t + 2, \dots, \min(t + K, T)\}$ exists, the $K = 1$ contract action space is strictly contained in that for $K > 1$.*

Proof of Proposition 2. The first statement is Lemma 2. If a feasible action with $a \in \{t + 2, \ldots, \min(t + K, T)\}$ exists, it belongs to the $K > 1$ action space but not the $K = 1$ space, so the containment is strict. This relation concerns action languages and does not compare task utilities or trajectories. □

Lemma 3 (restated). *If $B_t \geq 0$ and $R_t(a) \leq L_a$ before admission, then admitting any $A_t \in \mathcal{F}_t$ and atomically applying the specified debit and reservation update preserves both inequalities.*

Proof of Lemma 3. Suppose the ledger before admission satisfies $B_t \geq 0$ and $R_t(a) \leq L_a$ for every represented future step a. For an admitted set $A_t \in \mathcal{F}_t$, Eq. (10) gives $C(A_t) \leq B_t$ and

$$R_t(a) + \sum_{q \in A_t :\, a(q) = a} w(q) \leq L_a \quad \text{for every represented step } a.$$

The single debit therefore yields $B_{t+1} = B_t - C(A_t) \geq 0$. Because the reservation update is atomic, its value after admission is exactly the left side of the displayed capacity inequality and is thus at most L_a. Terminal transitions may release reservations but cannot increase them under the specified policy. This proves preservation over a single step. □

Proposition 3 (restated). *Assume positive costs, a single debit upon admission, atomic queue and reservation updates, and reapplication of the admission test after every replan. Then the remaining budget is nonnegative and reserved capacity at every future step a is at most L_a.*

Proof of Proposition 3. Lemma 3 preserves both inequalities at each admission. The initialized ledger has $B_0 \geq 0$ and no reservations. Induction over admissions proves Proposition 3. Telescoping the budget update also gives $\sum_t C(A_t) \leq B_0$. □

Lemma 4 (restated). *For a reconstructed ledger state and the next valid uniquely identified record, the acceptance or terminal transition semantics determine at most one successor ledger state.*

Proof of Lemma 4. Let the initial pending queue and reservation map be empty. Consider a reconstructed ledger after a trace prefix and the next record. An acceptance record inserts the contract named by its unique identifier and adds its recorded reservation. A cancellation, failure, completion, or supersession record applies the specified terminal transition to the contract with that identifier. Rescheduling terminates the old contract and inserts the linked new contract. Total order fixes which transition is applied next, while unique identifiers fix its target, so the successor ledger is unique whenever the record is valid. This proves Lemma 4. □

Proposition 4 (restated). *Given an empty initial queue and a complete, totally ordered event trace with unique contract identifiers, the pending queue, reservations, and due edge set are uniquely reconstructible.*

Proof of Proposition 4. Lemma 4 applies to each record in trace order. Induction over the complete trace yields a unique pending queue and reservation map after every prefix. At execution step t, filtering that queue by recorded activation step $a = t$ uniquely determines the due contracts and hence their typed due edge set, proving Proposition 4. □

Scope. These proofs do not establish task quality, efficiency, robustness, identification of timing effects, learnability of readiness or value estimates, semantic replay, diagnostic benefit, or novelty against all prior scheduling approaches. Schedule state reconstruction also does not recover prompts, model or tool versions, payload meaning, randomness, or environment state.

I Experimental Setup and Evaluation Protocol

This appendix specifies the datasets, tasks, baselines, and measurements used to organize the reported controlled aggregates. It also records evaluation fields that are not present in the supplied aggregate sheet and therefore remain protocol requirements rather than reported outcomes.

Table 6: Registered datasets and task definitions. "Selected by the plan" denotes a split chosen for the controlled protocol rather than inherited from a baseline paper.

Dataset	Task and unit of evaluation	Controlled comparison population and split role	Evaluator and primary outcome
HumanEval (Chen et al., 2021)	Generate one Python function from a specification; one programming problem is one item.	Official universe of 164 tasks partitioned by canonical order; optimization split of 40 items selected by the plan; disjoint HumanEval-heldout-124 confirmatory split; confirmatory items excluded from prompt, threshold, hyperparameter, and baseline selection.	Same functional correctness harness, container image, extraction rule, and timeout for all methods; `pass@1` is the proportion of items for which one generated solution passes all tests; invalid code, refusal, timeout, missing response, and budget exceedance count as failures.
MMLU (Hendrycks et al., 2021b)	Select one answer from `{A, B, C, D}` for a multiple choice question; one question is one item.	Confirmatory set: all 1,531 records from the 57 validation CSV files, parsed with a reader that handles CSV quoting; separate development set of 50 items, seed 42, stratified by supercategory, reserved for organizer fitting or threshold selection; the 1,530-item count reported by GTD is retained only for auditing fidelity to the authors when its excluded item can be identified.	Fixed normalizer extracts one answer label; normalized exact match accuracy is the item-weighted proportion of predictions containing one valid label equal to the gold label; ambiguous, multiple, or missing labels are failures; supercategory macro accuracy is supplementary.
Synthetic timing dependency tasks	Deliver registered typed edges to required recipients within registered execution step windows; one workflow motif is one episode.	Used only in Empirical Studies and RQ3 diagnostics; motifs, seeds, role roster, edge counts, budgets, and dependency windows fixed before execution.	Success is the proportion of episodes satisfying all required dependencies; required window coverage is the fraction of required edges delivered in their valid windows; wasted edge rate is the fraction of deliveries outside a useful registered window.

I.1 Evaluation Lanes, Datasets, and Tasks

Evaluation lanes. The author fidelity lane assesses whether an adapter can reproduce a baseline under that method's documented model, prompts, data subset, budget, and evaluator. It is a study of adapter correctness and is never combined with the controlled comparison lane in a ranking table. The controlled comparison lane fixes the model snapshot, roster of five agents, shared task envelope, decoding configuration, tools, logical execution steps, edge/call/token caps, paired item and seed cells, stopping rule, and evaluator. Only the organization or scheduling mechanism changes.

Datasets and tasks. The controlled lane has two primary tasks aligned with benchmarks, and they carry the effectiveness outcomes of RQ1, RQ2, and RQ5; the synthetic timing dependency tasks serve the RQ3 diagnostics, and RQ4 reuses the HumanEval-heldout-124 split under paired perturbations. HumanEval (Chen et al., 2021) evaluates functional Python code generation, and MMLU (Hendrycks et al., 2021b) evaluates multiple choice knowledge and reasoning. Table 6 records the evaluation population and split role.

The synthetic timing dependency tasks used by the preceding Empirical Studies are diagnostic tasks with registered windows for required edges. They are reported separately and never replace the two primary benchmarks.

I.2 Baselines and Fairness Conditions

Baseline families. Table 7 separates controls from baseline adapters; the families below supply the matched comparators for RQ1, the intact reference for the RQ2 ablations, and the necessity tests of

Table 7: Baseline families and their abstractions in the controlled comparison. Notes about provenance qualify each adapter and must not be read as claims of official reproduction.

Family / method	Role in the controlled comparison	Registered abstraction and fairness boundary	Provenance status
No communication, complete graph, and random graph	Necessity and capacity controls	No communication: no delivery among agents; complete graph: all valid directed deliveries enabled; random graph: the same number of valid directed edges as the matched method; these conditions test the necessity of communication and recipient selection under shared caps.	Controls selected by the plan
Fixed workflow, rule temporal scheduler, and single agent with matched compute	Workflow and controls for the necessity of MAS	Fixed workflow: a registered role sequence; rule scheduler: deterministic phase/dependency rules in the same action space as learned scheduling; single agent: matched aggregate output token and call allowances through serial self revision, with sequential path depth reported separately.	Controls selected by the plan
G-Designer (Zhang et al., 2025c)	Static organizer tailored to each task	Directed graph tailored to the task generated at episode start and reused at later execution steps; the controlled adapter cannot inspect the execution trajectory when updating topology.	Controlled aggregate certified by authors and based on official code
AgentPrune (Zhang et al., 2025b)	Frozen spatial and temporal organizer	Spatial and temporal mask learned or selected on the optimization split, then frozen for confirmatory items; trajectory information at test time cannot update the mask.	Controlled aggregate certified by authors and based on official code
CARD (Wu et al., 2026)	Organizer based on context	Graph generated at episode start from task and registered exogenous condition information; recomputation allowed only when the registered condition changes, not in response to unregistered endogenous execution history.	Controlled aggregate certified by authors and based on official code
GTD (Jiang et al., 2026)	Graph generator tailored to each task	Candidate graph generated for the task from its registered proxy/generation interface, then held fixed within the episode.	Controlled aggregate certified by authors with unresolved adapter provenance
DyTopo (Lu et al., 2026)	Dynamic routing comparator at each step	Directed graph reconstructed at each execution step from the manager goal and need/offer descriptors defined in the paper, then projected to the shared action space containing the top k entries.	Independent reimplementation that is not an official reproduction
TGO-MAS	Proposed organizer aware of execution steps	Typed directed edges for the next step selected from the observable MAS execution state and history, subject to the same projection and resource constraints; the reported evaluation concerns the core organizer and does not establish optional $K > 1$ commitments.	Controlled aggregate certified by authors

RQ5. Every method that produces graphs in the controlled lane uses the same projection to the top k entries, deterministic rule for breaking ties, stopping rule, worker call cap, token cap, and evaluator. Sparsity and early stopping behavior specific to each method are reported in a separate author fidelity or supplementary view, not as superiority under matched budgets. The provenance status is kept explicit. An official repository is not equivalent to an official reproduction, and a paper description is not equivalent to a complete adapter.

Table 8: Evaluation metrics and their intended interpretation. "Higher"/"lower" and ↑/↓ identify the preferred direction only within a matched protocol.

Metric family	Measure	Direction	Meaning
Effectiveness	HumanEval `pass@1` and MMLU normalized exact match accuracy	Higher	Correctness on the benchmarks. This is the share of evaluated items solved by one code sample or answered with the correct normalized option label, respectively. These are the RQ1 headline metrics.
Efficiency	Worker, controller, and evaluator calls, prompt, cached input, completion and total tokens, unique messages, active and delivered edges, critical path calls, service and end to end latency, and monetary cost	Lower, conditional on quality	Total system resource use. Worker and controller overhead are reported separately and also included in system totals. Critical path calls measure sequential model call depth, whereas end to end latency includes queueing and transport.
Scheduling diagnostics	Activation window precision/recall, deadline misses, capacity use at each step, interaction before a valid window, wasted edge rate, and schedule stability	Depends on context	Measures whether a schedule activates useful interactions at registered execution steps and avoids unnecessary deliveries. These diagnostics are used where a registered oracle window or deterministic dependency construction exists. They are not a replacement for task correctness.
Robustness	Attacked task quality, paired clean degradation, retention, recovery, and failure type distribution	Quality ↑ and degradation ↓	Compares each perturbed execution with its matched clean execution. Clean degradation is $y_{\text{clean}} - y_{\text{attack}}$. Retention is attacked quality relative to clean quality when defined. Failure types explain, rather than erase, unsuccessful episodes.
Trace utility and replay	Blinded localization accuracy for fault type, faulty agent, and faulty step, diagnosis time, and replay conformance	Accuracy ↑ and time ↓	The blinded comparison tests whether a structured lifecycle trace helps an auditor diagnose a registered fault relative to the same raw log record. Replay conformance separately tests whether recorded lifecycle events reconstruct the registered schedule state. It does not by itself support semantic replay or interpretability.

I.3 Metrics and Their Interpretation

Metric policy. The five metric families below are the outcome vocabulary for RQ1 through RQ5. Correctness on the benchmarks is the primary outcome. Resource, schedule, robustness, and trace metrics make the associated tradeoffs or mechanisms auditable. None is combined into a new weighted score. Table 8 defines the direction and interpretation of each reported quantity. All terminal outcomes, including failures, refusals, timeouts, invalid outputs, and budget exceedances, remain in their relevant denominators.

I.4 RQ1–RQ5: Task and Metric Map

Table 9 states the estimand mapping. The comparison unit for task outcomes is the paired item and seed cell. The registered item→seed analysis calls for 10,000 hierarchical bootstrap replicates, 95% confidence intervals, paired win/tie/loss counts, and Holm correction within each RQ family. The supplied aggregate sheet reports means and standard deviations but not these inferential outputs. Accordingly, this manuscript does not make significance claims. Seeds remain nested stochastic repeats rather than independent task samples.

Table 9: Registered task and metric mapping for the reported aggregate results from RQ1 through RQ5.

RQ	Evaluation material	Primary outcomes	Required comparison or interpretation boundary
RQ1	HumanEval-heldout-124 and MMLU-validation-1531	`pass@1` or normalized exact match accuracy, together with total system efficiency measures	Matched comparison of effectiveness and efficiency against the registered controls and adapters. Quality and resource tradeoffs are reported without a composite score.
RQ2	The same primary datasets, restricted to implemented components	Paired change in the RQ1 benchmark and total system metrics	Each ablation removes one input or operation defined by the scheduling requirements. The Empirical Studies probe related distinctions separately, and an unimplemented component cannot be claimed necessary.
RQ3	Synthetic timing dependency tasks and registered HumanEval timing diagnostics	Success, required window coverage, and wasted edges for synthetic tasks, together with paired task quality change and applicability fraction for diagnostics across the full pipeline	Valid time permutation is separated from recipient rewiring, stale state input, and replay with fixed payloads. Replay supports a diagnostic claim about delivery timing, not the full estimand for the adaptive organizer.
RQ4	HumanEval-heldout-124 paired clean and perturbed records	Attacked task quality and paired clean degradation, together with retention, recovery, resource deltas, and failure type distribution	Interventions involving a silent agent, a corrupted message, and a delay of one step preserve their registered caps. Any recovery that uses extra overhead is a system tradeoff.
RQ5	The RQ1 primary datasets with necessity controls	The same benchmark effectiveness and total system efficiency outcomes as RQ1	No communication, complete/random graph, fixed workflow, rule scheduler, and a single agent with matched compute test communication, temporal organization, and the necessity of MAS separately.

I.5 Trace Diagnostic Protocol in the Appendix

Diagnostic comparison. The trace utility and replay measures of Table 8 sit outside RQ1–RQ5. This protocol in the appendix compares an ordered raw message log with a structured lifecycle trace derived from the same frozen execution record with an injected fault. The two blinded materials preserve the task identifier, fault injection, scoring rubric, and time limit while hiding method identity and view provenance. Auditors report the fault type, faulty agent, faulty execution step, and diagnosis completion time. These outcomes evaluate the diagnostic usefulness of the record format. They are not an RQ about task effectiveness.

Replay conformance. Separately, the ordered lifecycle events are used to reconstruct the registered queue, reservation, due schedule, and delivery state. Exact conformance and the first mismatch event are reported apart from blinded diagnostic outcomes. Before this appendix protocol is run, the auditor population, fault taxonomy and injection procedure, target sample size, time limit, compensation or ethics handling, and threshold for practical effects must be frozen. Until then, neither diagnostic utility nor interpretability is an empirical claim.

The complete execution contract, manifest hashes, staged gates, cost controls, and rejection criteria cannot be independently inspected because the corresponding plan and reproducibility checklist

artifacts are absent from the supplied source package. This appendix therefore records the intended protocol boundary and does not present any unrun component as a completed result.

J Additional Results and Diagnostics

Tables 10–14 give complete numeric backing for the result figures and compact tables in the main text: Table 10 backs Figure 2 of Study B, Table 11 expands the compact Table 2 of Study C, Table 12 backs Figure 4 of RQ3, Table 13 backs Figure 5 of RQ4, and Table 14 backs Figure 6 of RQ5. The tables reproduce the supplied aggregates without adding dispersion estimates or interpretive cells.

Table 10: Required-window coverage retention under signed temporal shifts (%). Entries are supplied aggregate percentages without dispersion estimates; the five shift settings are discrete interventions and do not define intermediate schedules. Numeric backing for Figure 2.

	Tight window					Wide window				
Method	-2	-1	0	$+1$	$+2$	-2	-1	0	$+1$	$+2$
G-Designer	31.45	60.98	100.00	57.42	28.12	76.92	93.41	100.00	92.54	73.65
AgentPrune	28.91	57.34	100.00	52.89	23.56	76.12	90.85	100.00	90.67	72.14
CARD	35.21	62.94	100.00	59.87	31.22	79.45	93.62	100.00	93.56	75.41
DyTopo	42.34	68.75	100.00	67.54	38.21	85.32	95.12	100.00	94.87	83.45

Table 11: History-view results for next-edge selection (%), expanding the compact Table 2. ↑/↓ mark the preferred direction, and an em dash marks a quantity that is not defined for that control; entries are supplied aggregates without dispersion estimates.

History view	Next-edge success ↑	Redundant-edge rate ↓	Pairwise ranking accuracy ↑
Static top-1	50.82	0.00	49.43
Activate both	100.00	50.76	—
Raw ordered history	71.45	23.12	67.84
Structured lifecycle history	92.54	8.45	87.65

Table 12: Mechanism interventions across the full pipeline. Entries are mean ± standard deviation where supplied, paired changes are relative to clean TGO-MAS, and "Applicable" is the fraction of episodes to which the intervention applies. Numeric backing for Figure 4.

Condition	HE pass@1 (%)	Paired change (pp)	Applicable (%)	Coverage (%)	Deadline misses (%)
Clean	76.64 ± 0.85	$+0.00$	100.0	81.2 ± 1.5	8.7 ± 0.9
Time permutation	73.52 ± 1.05	-3.12	91.2 ± 0.5	63.7 ± 1.8	24.6 ± 1.5
Recipient shuffle	74.58 ± 1.12	-2.06	100.0	69.5 ± 1.4	19.1 ± 1.3
Stale state	74.86 ± 1.08	-1.78	100.0	70.8 ± 1.2	17.3 ± 1.5

Table 13: Robustness under paired perturbations with degradation relative to clean runs and retention from attacked to clean runs. Entries are mean ± standard deviation where supplied, and degradation is the paired clean minus attacked change in points. Numeric backing for Figure 5.

Condition	TGO-MAS pass@1	DyTopo pass@1	TGO-MAS degradation (pp)	DyTopo degradation (pp)	TGO-MAS retention (%)	TGO-MAS recovery (%)
Clean	76.64 ± 0.85	74.45 ± 0.82	0.00	0.00	100.0	100.0
Silent agent	73.27 ± 1.15	69.14 ± 1.32	3.37	5.31	95.6 ± 0.4	68.4 ± 1.2
Corrupted message	72.21 ± 1.20	67.58 ± 1.35	4.43	6.87	94.2 ± 0.5	59.7 ± 1.4
Delay of one step	73.72 ± 1.08	68.95 ± 1.25	2.92	5.50	96.2 ± 0.4	71.3 ± 1.5

Table 14: Effectiveness and total token results for communication, organization, and controls with matched compute. Entries are mean ± standard deviation, and tokens are system totals in thousands. Numeric backing for Figure 6.

	HumanEval-heldout-124		MMLU-validation-1531	
Control / method	pass@1 (%)	Tokens (k)	Accuracy (%)	Tokens (k)
No communication	66.73 ± 1.25	14.03 ± 0.45	72.50 ± 1.42	8.94 ± 0.32
Random graph	69.88 ± 1.32	20.94 ± 0.55	74.60 ± 1.51	12.84 ± 0.48
Complete graph	72.51 ± 1.38	27.14 ± 0.68	75.92 ± 1.45	16.94 ± 0.52
Fixed workflow	71.86 ± 1.22	20.24 ± 0.51	75.69 ± 1.38	12.64 ± 0.49
Rule scheduler	74.20 ± 1.35	20.34 ± 0.58	76.86 ± 1.48	12.64 ± 0.53
Single agent	70.98 ± 1.42	21.24 ± 0.62	74.96 ± 1.55	13.04 ± 0.58
TGO-MAS	76.64 ± 0.85	20.93 ± 0.44	78.39 ± 0.88	12.94 ± 0.46

No blinded trace diagnostic outcome is included because the supplied result set contains none. The material on trace utility and replay conformance in Appendix I remains a protocol rather than a completed result.

K Reproducibility and Conformance

Table 15 records, for each mechanism of the scheduling specification, the artifact status of the supplied source package and the verification boundary that remains because no executable artifact, test fixture, or conformance receipt is included.

Table 15: Specification status in the supplied source package. "Specified" marks mechanisms defined in the manuscript; "Protocol only" and "Specified only" mark the absence of an executable artifact, test fixture, or conformance receipt for that mechanism.

Mechanism	Package status	Verification boundary
Typed edges indexed by step and immutable schedules	Specified	No executable artifact is included for implementation validation
Edge count capacity at each step	Specified	The manuscript defines the capacity cap and the policy that disallows carryover
Timing/recipient preservation checks	Protocol only	No test fixture or conformance receipt is included
State encoder restricted to the observed prefix and readiness/value interface	Specified	Learned estimators must obey the observed prefix interface
Exact selection from the feasible set and feedback replanning	Specified	The objective concerns the stated surrogate, not true task utility
Delayed contracts with $K > 1$	Specified only	Neither runner integration nor executed validation is included
Episode debit and reservation ledger	Specified only	The cancellation policy without refunds is defined but not verified by an artifact
Complete lifecycle trace and schedule state replay	Specified only	Schedule state reconstruction is stated. No replay receipt is included

The intended run record contains immutable protocol and environment hashes, baseline and source status, item, seed, condition, prediction, resource use, edge events, and terminal failure status. The supplied package contains no executable implementation, run manifest, test fixture, or conformance receipt. Consequently, this appendix documents the scheduling specification and its intended checks. It does not independently verify implementation, empirical effectiveness, robustness, replay, or diagnostic utility.