GAMMA-LLM: GAME-THEORETIC ADAPTIVE MULTI-AGENT MESSAGE ALLOCATION FOR LLM REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-agent large language model systems can improve reasoning by combining independent hypotheses and intermediate feedback, but additional communication is not uniformly beneficial: redundant or poorly targeted interactions increase inference-time token usage and can propagate incorrect rationales. We propose GAMMA, a game-theoretic framework for adaptive multi-agent message allocation that estimates which agents should receive greater influence and which communication edges have positive marginal value under a specified communication objective. GAMMA models agents as players in a graph-restricted cooperative game, constructs offline supervision from node-level Myerson contributions and cost-aware conditional edge values, and amortizes these quantities into a lightweight graph policy for sparse communication and contribution-weighted aggregation. Across nine reasoning benchmarks, GAMMA achieves higher macro-averaged accuracy than the evaluated adaptive multi-agent baselines while using fewer online inference tokens. Ablation, calibration, and robustness analyses further show improved prediction of the game-derived targets, lower error propagation, and fewer wrong-consensus failures.

1 INTRODUCTION

Large language models increasingly operate as components of multi-agent reasoning systems rather than as isolated predictors. Multiple agents can generate independent hypotheses, critique intermediate reasoning, and aggregate complementary evidence, motivating debate-based systems (Du et al., 2024; Liang et al., 2024), solver–critic loops (Wu et al.), role-specialized agent societies (Qian et al., 2024; Hong et al., 2023), dynamic agent selection (Liu et al., 2023), and learned communication topologies (Zhang et al.). Yet communication is not uniformly useful (Huang et al.). Many instances do not require additional interaction, making extra discussion an unnecessary inference-time cost (Snell et al., 2024; Wang et al., 2025); on harder instances, poorly targeted communication can also propagate incorrect rationales or reinforce brittle consensus (Xie et al., 2026). The central question is therefore not simply whether agents should communicate, but *who should influence the final decision and which messages are worth transmitting*.

Existing adaptive methods partially address this question by learning which agents or communication edges to retain (Shen et al., 2025; Zhang et al., 2025). However, optimizing graph-level performance or pruning a topology does not by itself specify how the resulting utility should be credited to individual agents and messages. This distinction matters because communication value is inherently context dependent: the contribution of an agent depends on what other agents already provide, while the value of an edge depends on the communication structure already present. Unlike graph-level reward regression or similarity-based scoring, GAMMA supervises two conditionally distinct quantities: an agent's marginal contribution to coalition utility and an edge's cost-adjusted marginal gain given the current communication graph. The former depends on which other agents already contribute to the coalition, while the latter depends on which communication paths are already available. Thus, neither agent value nor edge value is an intrinsic score that can be recovered from confidence, similarity, or a single graph-level reward alone.

This structure motivates a game-theoretic credit-assignment formulation. Cooperative-game theory defines contribution through marginal changes relative to coalitions (Shapley et al., 1953; Ghorbani & Zou, 2019), while graph-restricted variants incorporate communication constraints (Myerson, 1977).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

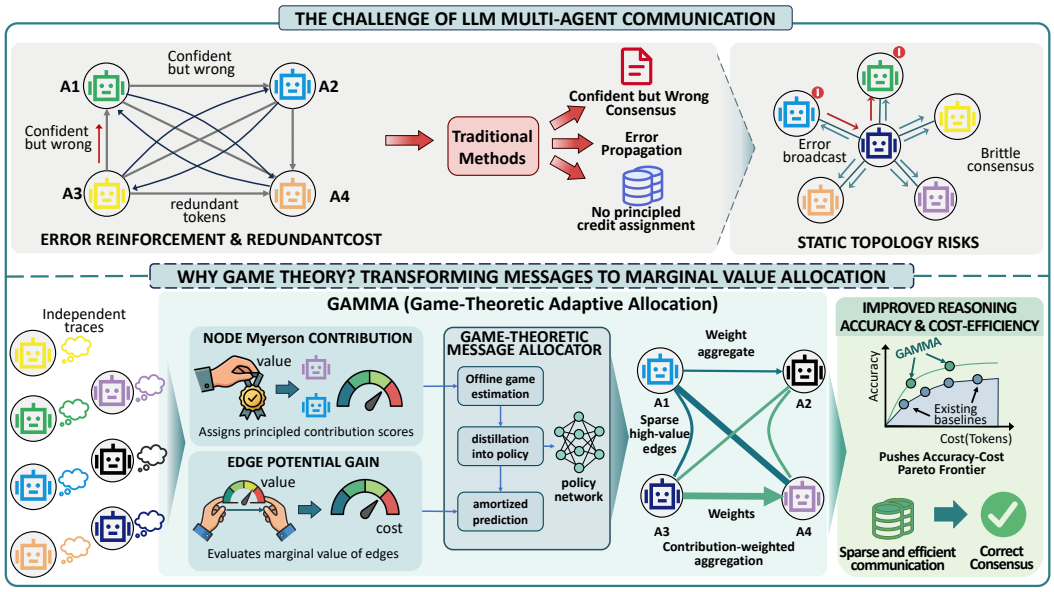


Figure 1: Motivation and overview of GAMMA. GAMMA reframes LLM multi-agent communication from dense or fixed topology design into game-theoretic marginal-value allocation, using node contributions and conditional edge values to enable sparse, online communication-efficient aggregation.

These tools align directly with the two decisions required at inference time: *node credit* estimates how much influence an agent should receive under graph-restricted cooperation, while *edge credit* estimates the incremental value of adding a candidate message under the current communication context and its cost. Crucially, the two cannot be collapsed into a single topology score: a valuable agent need not communicate densely, and an individually strong agent can still participate in a redundant or harmful exchange.

We propose GAMMA (Game-Theoretic Adaptive Multi-Agent Message Allocation), which operationalizes this two-level view. At the node level, GAMMA constructs offline Myerson contribution targets under graph-restricted cooperation. At the edge level, it constructs graph-conditional, cost-adjusted targets that measure the marginal value of adding a candidate communication edge. These game-derived quantities are computed only during offline supervision and amortized into a lightweight graph neural policy, avoiding coalition enumeration at inference time (Wang et al., 2026). Given independent agent traces, the online policy predicts node contributions and candidate-edge gains, executes only sparse edges with positive predicted cost-adjusted value, and aggregates answers using contribution-weighted voting. Across nine reasoning benchmarks, GAMMA achieves higher average accuracy than the evaluated adaptive multi-agent baselines while using fewer online inference tokens.

Our contributions are:

1. We formulate adaptive LLM communication as game-theoretic message allocation, separating agent-level contribution from edge-level marginal value.
2. We derive node-level Myerson and cost-aware edge supervision from offline graph rollouts and amortize them into a sparse online graph policy.
3. Across nine reasoning benchmarks, GAMMA improves average accuracy over evaluated adaptive baselines while using fewer online tokens, with ablations and diagnostics supporting the proposed allocation signals.

2 Related Work

Single-agent reasoning. Chain-of-thought prompting, self-consistency, tree search over thoughts, and graph-structured reasoning show that additional inference-time computation can improve LLM

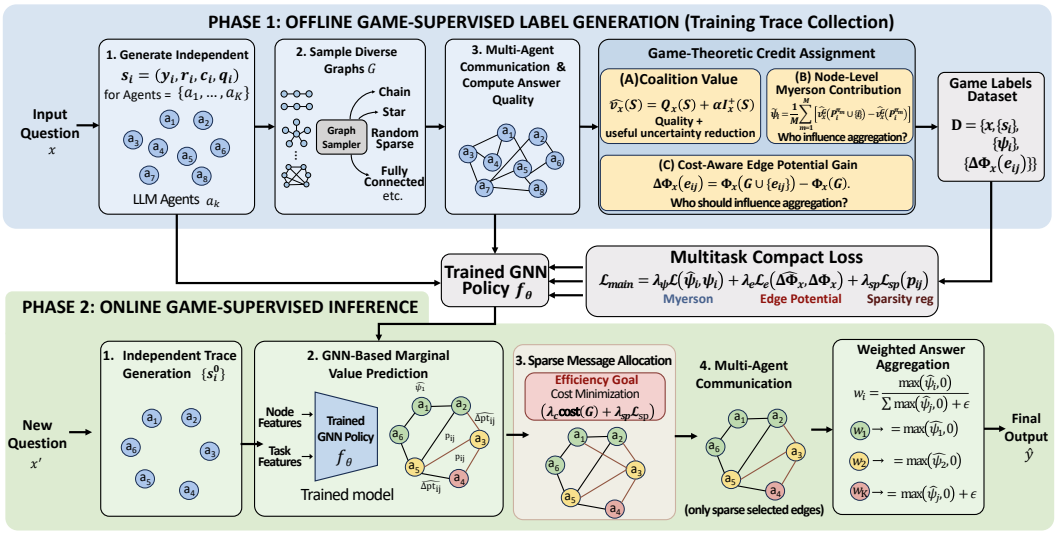


Figure 2: Overview of GAMMA. Phase 1 performs offline game-supervised label generation. Phase 2 performs amortized online inference.

reasoning (Wei et al., 2022; Wang et al.; Yao et al., 2023; Besta et al., 2024). These methods are important baselines because they spend additional tokens without introducing inter-agent communication. In our experiments, they test whether gains come from structured message allocation rather than token count alone.

LLM multi-agent collaboration. Multi-agent systems such as CAMEL, AutoGen, AgentVerse, MetaGPT, debate frameworks, and reflective collaboration introduce role specialization, dialogue protocols, critic–solver loops, and agent societies (Li et al., 2023; Wu et al.; Chen et al.; Hong et al., 2023; Du et al., 2024; Bo et al., 2024). These systems demonstrate that multiple agents can produce complementary reasoning, but many rely on fixed or manually designed communication patterns (Wang et al., 2024). GAMMA focuses on deciding which communication events have positive marginal value.

Dynamic agent selection and topology learning. DyLAN performs inference-time agent selection with early stopping (Liu et al., 2023). GPTSwarm represents LLM workflows as optimizable computational graphs (Zhuge et al., 2024). G-Designer learns task-adaptive communication topologies with a graph neural network, while AGP jointly learns hard agent pruning and soft edge pruning (Zhang et al.; Li et al., 2025; Liu et al., 2024). AMAS and guided topology diffusion further study adaptive topology determination and generative topology synthesis (Leong et al., 2025; Jiang et al., 2025). These are the closest baselines. GAMMA differs in making this credit decomposition explicit: its policy is supervised with separate game-derived targets for agent contribution and candidate-edge marginal value, rather than relying solely on a graph-level objective for communication selection.

3 Problem Formulation

We study adaptive communication in a multi-agent LLM reasoning system. Given an input task x and a pool of K agents $\mathcal{A} = \{a_1, \ldots, a_K\}$, each agent first produces an independent reasoning trace $s_i = (y_i, r_i, c_i, q_i, \tau_i)$, where y_i is the parsed answer, r_i is the rationale, c_i is self-reported confidence when available, q_i is an optional inference-available auxiliary quality signal (e.g., a task-specific verifier score when such a verifier is part of the evaluation protocol), and τ_i is the token cost. We distinguish such inference-available signals from gold correctness labels used only for offline target construction.

A directed communication graph $G = (\mathcal{N}, \mathcal{E})$ specifies message flow, where $e_{ij} \in \mathcal{E}$ denotes a message from agent i to agent j. For graph-restricted coalition valuation, $\mathrm{Comp}(G[S])$ denotes the

weakly connected components of the directed subgraph induced by S. We use weak connectivity only for the cooperative-game restriction: it captures whether members of a coalition belong to the same underlying communication component, abstracting away edge orientation at the node-credit level. Directionality is preserved in actual message execution and in the graph-conditional edge value, separating coalition feasibility for node-level credit assignment from directional information flow for edge selection.

After communication under G, agents produce graph-conditioned states $\{s_i^G\}_{i=1}^K$, and an aggregation rule returns the final answer $\hat{y}$.

The goal is to learn a graph policy π_θ that selects useful communication while controlling cost:

$$\max_{\pi_\theta} \ \mathbb{E}_{x\sim p(x)} \left[Q(\hat{y}, x) - \lambda_c \mathcal{C}(G, x) \right], \tag{1}$$

where $Q(\hat{y}, x)$ measures task quality, $\mathcal{C}(G, x)$ denotes the communication-related online inference cost induced by G, and λ_c controls the quality–communication tradeoff.

4 The GAMMA Framework

Figure 2 gives an overview of GAMMA. The framework has two phases. In the offline phase, GAMMA collects multi-agent traces under diverse candidate graphs and computes game-theoretic labels for agents and edges. In the online phase, a trained graph neural policy predicts these values from independent traces, selects a sparse communication graph, and aggregates answers using contribution-aware weights.

4.1 Phase 1: Offline Game-Supervised Label Generation

The purpose of the offline phase is to convert expensive cooperative-game reasoning into two deployable forms of supervision. At the node level, GAMMA estimates which agents make positive marginal contributions and should therefore receive influence during aggregation. At the edge level, GAMMA estimates which communication links create enough cooperative value to justify their token cost.

Offline supervision data. Let $\mathcal{D}_{\text{train}}$ denote the designated training instances used for game-label construction in each reported benchmark setting. All graph rollouts, coalition-value computation, and graph-policy fitting are performed only on training data; evaluation instances are not used to construct game labels or fit the policy. The fitted policy is frozen for evaluation.

For each training instance, we first collect independent traces $\{s_i\}_{i=1}^K$ from all agents and then run multi-agent communication under a diverse set of candidate graphs, including an empty graph, a chain, a star, random sparse graphs, a fully connected graph, and value-guided sparse graphs produced by preliminary graph policies. The value-guided graphs are used only to diversify the offline rollout distribution. These rollouts provide post-communication states, final answers, verifier scores, and token costs, which are used to derive node-level Myerson contributions and cost-aware conditional edge-value targets.

4.1.1 Coalition Value

For a coalition $S \subseteq \mathcal{N}$, GAMMA defines a training-time coalition value as

$$\widehat{v}_x(S) = Q_x(S) + \alpha I_x^+(S), \tag{2}$$

where $Q_x(S)$ denotes the task-specific quality of the answer produced by coalition S under the offline rollout protocol, and $I_x^+(S)$ measures useful uncertainty reduction. For exact-answer tasks, $Q_x(S)$ is computed from correctness; for tasks with executable or task-specific verifiers, it is computed from the corresponding verifier feedback. By definition, $Q_x(S)$ is evaluated on the coalition output rather than on the full-agent output. Gold annotations, when available, are used only for offline target construction and are not exposed to the policy at inference time. Offline label generation therefore requires labeled training data or an informative task-specific verifier.

The information term is defined as

$$I_x^+(S) = g_x(S) \cdot \Delta\mathrm{Ent}_x(S), \tag{3}$$

where

$$\Delta\mathrm{Ent}_x(S) = H(p_x^{\mathrm{pre}}(S)) - H\big(p_x^{\mathrm{post}}(S)\big)\,, \tag{4}$$

and $p_x^{\mathrm{pre}}(S)$ and $p_x^{\mathrm{post}}(S)$ denote the coalition's empirical answer distributions before and after communication, respectively. The gate $g_x(S) \in [0,1]$ is instantiated by the task-quality signal used for offline supervision (binary correctness for exact-answer tasks or a normalized task-specific verifier score when applicable). Multiplying entropy reduction by this quality gate avoids rewarding reductions in uncertainty that accompany an incorrect or low-quality coalition output.

The value in Eq. equation 2 is deliberately simple. It defines contribution relative to task quality and useful uncertainty reduction. Communication cost is not mixed into this coalition value; instead, cost is introduced separately when constructing the edge-selection target.

4.1.2 Node-Level Myerson Contribution

Coalition value alone tells us whether a group of agents is useful, but it does not tell us which individual agent should receive credit. GAMMA therefore derives a node-level Myerson contribution for each agent. Since communication is constrained by a graph, we first use a graph-restricted version of the coalition value. If a coalition S spans multiple weakly connected components under G, those components are treated as separate cooperative units for node-level credit assignment. Using $\mathrm{Comp}(G[S])$ as defined in Section 3, we define

$$\widehat{v}_x^G(S) = \sum_{C \in \mathrm{Comp}(G[S])} \widehat{v}_x(C). \tag{5}$$

This is not an additional supervision target. It is an intermediate transformation that makes the Myerson contribution respect the communication graph. For a random permutation π of agents, let P_i^π be the set of agents that appear before agent i. The graph-restricted marginal contribution of agent i is

$$\Delta_i^\pi = \widehat{v}_x^G(P_i^\pi \cup \{i\}) - \widehat{v}_x^G(P_i^\pi). \tag{6}$$

We approximate the Myerson contribution using M Monte Carlo permutations:

$$\widetilde{\psi}_i^G = \frac{1}{M}\sum_{m=1}^{M}\left[\widehat{v}_x^G(P_i^{\pi_m} \cup \{i\}) - \widehat{v}_x^G(P_i^{\pi_m})\right]. \tag{7}$$

The resulting label measures agent contribution relative to the value function in Eq. equation 2 and the sampled graph context. A confident agent can receive low contribution if it adds little value under this objective, while a low-confidence agent can receive higher contribution when its inclusion improves the coalition value. Thus, $\widetilde{\psi}_i^G$ provides a structured supervision target for contribution-aware aggregation.

4.1.3 Cost-Aware Conditional Edge Value

To construct an edge-selection target, GAMMA uses the graph-conditioned outcome of the offline communication rollout. We define the cost-aware graph potential

$$\Phi_x(G) = Q_x(G) + \alpha I_x^+(G) - \lambda_c \mathcal{C}(G, x), \tag{8}$$

where $Q_x(G)$ is the task quality after executing communication under G and $I_x^+(G)$ is the corresponding useful uncertainty reduction, defined analogously to Eq. equation 3 from the pre- and post-communication answer distributions under G. This graph-conditioned potential is used for edge labeling; the graph-restricted coalition value in Eq. equation 5 remains the quantity used for node-level Myerson credit.

For a candidate edge $e_{ij} \in \mathcal{E}_{\mathrm{cand}}$, its graph-conditional edge gain is

$$\Delta\Phi_x(e_{ij} \mid G) = \Phi_x(G \cup \{e_{ij}\}) - \Phi_x(G). \tag{9}$$

The quantity is explicitly graph-context dependent: the same edge can have different marginal value under different existing communication structures. For example, a message from agent i to agent j may be useful when no equivalent information path exists, but become redundant once another path already conveys the same evidence. A positive value means that adding the edge increases the

defined cost-adjusted graph potential in that context, whereas a negative value means that it decreases the potential. We therefore use $\Delta\Phi_x(e_{ij} \mid G)$ as a graph-conditioned supervision target for edge selection. Because Eq. equation 9 is already a finite difference of graph potential, GAMMA does not require an additional graph-potential regression loss.

4.1.4 GAME-SUPERVISED GRAPH POLICY TRAINING

After generating node and edge labels, GAMMA trains a graph neural policy f_θ to predict amortized game-derived quantities from independent traces. Given trace-derived node, edge, and task features, the policy predicts $\hat{\psi}_i$ and $\widehat{\Delta\Phi}_{ij}$, where $\hat{\psi}_i$ is the predicted node contribution and $\widehat{\Delta\Phi}_{ij}$ is the predicted edge gain for candidate edge e_{ij}. We derive a soft edge-selection score as $p_{ij} = \sigma(\widehat{\Delta\Phi}_{ij})$, which is used by the sparsity regularizer. The training objective is intentionally compact:

$$\mathcal{L}_{\text{main}} = \lambda_\psi \underbrace{\frac{1}{|\mathcal{N}|}\sum_{i\in\mathcal{N}}\left\|\hat{\psi}_i - \widetilde{\psi}_i^G\right\|_2^2}_{\mathcal{L}_\psi} + \lambda_e \underbrace{\frac{1}{|\mathcal{E}_{\text{cand}}|}\sum_{e_{ij}\in\mathcal{E}_{\text{cand}}}\left\|\widehat{\Delta\Phi}_{ij} - \Delta\Phi_x(e_{ij} \mid G)\right\|_2^2}_{\mathcal{L}_e} + \lambda_{sp} \underbrace{\frac{1}{|\mathcal{E}_{\text{cand}}|}\sum_{e_{ij}\in\mathcal{E}_{\text{cand}}} p_{ij}}_{\mathcal{L}_{sp}} \tag{10}$$

The first term, $\mathcal{L}_\psi$, supervises node-level contribution prediction by regressing $\hat{\psi}_i$ to the graph-conditioned offline Myerson target $\widetilde{\psi}_i^G$. The second term, $\mathcal{L}_e$, supervises edge-level value prediction by regressing $\widehat{\Delta\Phi}_{ij}$ to the graph-conditional edge target $\Delta\Phi_x(e_{ij} \mid G)$. The third term, $\mathcal{L}_{sp}$, penalizes the average soft edge-selection score and encourages sparse communication. The coefficients λ_ψ, λ_e, and λ_{sp} control the relative importance of node contribution regression, edge-value regression, and sparsity regularization, respectively. Cost awareness enters through the offline edge target in Eq. equation 9, whereas $\mathcal{L}_{sp}$ imposes a generic preference for sparse selection. These mechanisms are complementary: the former favors communication with positive cost-adjusted marginal value, while the latter discourages unnecessary dense connectivity. The complete offline label-generation and training procedure is provided in Appendix H.

4.2 PHASE 2: AMORTIZED ONLINE INFERENCE

At inference time, GAMMA does not enumerate coalitions or compute Myerson values. The expensive game-theoretic credit assignment has already been distilled into the trained policy f_θ. Given a new question x', agents first generate independent traces $\{s_i\}_{i=1}^K$. The trained policy predicts amortized estimates of the offline node and edge targets. GAMMA then selects a sparse graph by retaining edges whose predicted cost-adjusted gain exceeds the edge threshold, subject to the communication budget:

$$\mathcal{E}^* = \left\{ e_{ij} \in \mathcal{E}_{\text{cand}} : \widehat{\Delta\Phi}_{ij} > \tau_\Phi \right\}, \tag{11}$$

where the default threshold is $\tau_\Phi = 0$, with additional budget constraints applied when required.

Communication is executed only along the selected edges $\mathcal{E}^*$. The final answer is aggregated using normalized nonnegative contribution weights:

$$w_i = \begin{cases} \dfrac{\max(\hat{\psi}_i, 0)}{\sum_{j=1}^K \max(\hat{\psi}_j, 0)}, & \text{if } \sum_{j=1}^K \max(\hat{\psi}_j, 0) > 0, \\ \dfrac{1}{K}, & \text{otherwise.} \end{cases} \tag{12}$$

Thus, if all predicted contributions are non-positive, GAMMA falls back to uniform aggregation rather than using an ill-defined normalization. For multiple-choice and discrete-answer tasks, GAMMA uses contribution-weighted voting. For numerical reasoning, it groups equivalent normalized answers before voting. For code generation, inference-available verifier feedback is used as an additional filter before contribution-weighted selection. The complete online inference procedure is provided in Appendix H.

Table 1: Main benchmark results on all nine datasets using Qwen-3.5-9B.

	Math		Knowledge		Code		QA		Long		
Method	GSM8K	MATH	MMLU	GPQA	HEval	MBPP	SQA	HQA	LBR	Avg.	Tokens
Single-agent reasoning											
Single	84.6	63.8	73.9	39.4	71.2	66.7	73.1	59.8	54.6	65.2	285
CoT	87.3	67.5	76.2	42.1	73.8	69.0	75.6	62.4	57.3	67.9	448
Self-Consistency	89.1	70.2	77.8	44.3	76.0	71.4	77.3	64.7	59.5	70.0	2 310
Tree-of-Thought	88.6	69.5	77.3	43.7	75.4	70.8	76.8	64.0	58.9	69.4	2 940
Graph-of-Thought	88.8	69.8	77.5	44.0	75.7	71.1	77.0	64.3	59.2	69.7	3 320
Static multi-agent protocols											
Majority Vote	88.2	69.0	77.0	43.2	75.0	70.3	76.4	63.5	58.4	69.0	2 050
Chain	87.0	67.8	76.5	42.5	73.5	69.4	75.8	62.8	57.7	68.1	4 420
Star	87.6	68.3	76.8	42.9	74.1	69.8	76.1	63.1	58.0	68.5	3 640
Full Debate	85.5	65.9	74.6	40.8	72.4	68.0	74.2	61.1	56.0	66.5	8 780
AutoGen SC	88.0	68.7	77.2	43.0	75.8	70.9	76.2	63.3	58.2	69.0	5 040
Adaptive multi-agent systems											
DyLAN	88.8	69.7	77.9	43.9	76.2	71.3	77.1	64.4	59.1	69.8	4 050
GPTSwarm	89.4	70.3	78.3	44.5	76.8	72.0	77.6	65.0	59.8	70.4	4 830
G-Designer	90.5	72.1	79.8	46.2	78.5	73.8	78.9	66.7	61.5	72.0	3 620
AGP	91.2	73.0	80.4	47.0	79.4	74.6	79.5	67.3	62.2	72.7	3 110
AMAS	90.0	71.5	79.3	45.7	78.0	73.2	78.4	66.1	60.9	71.5	4 280
Guided Top. Diff.	90.8	72.5	80.0	46.5	78.9	74.1	79.2	66.9	61.8	72.3	3 940
GAMMA	**92.8**	**75.4**	**82.6**	**49.8**	**81.7**	**77.2**	**81.3**	**69.7**	**64.5**	**75.0**	2 860

5 EXPERIMENT

5.1 EXPERIMENTAL SETUP

Benchmarks. We evaluate GAMMA on nine datasets covering five reasoning families: GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021) for mathematical reasoning; MMLU (Hendrycks et al., 2020) and GPQA-Diamond (Rein et al., 2023) for knowledge-intensive and scientific reasoning; HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021) for code generation; StrategyQA (Geva et al., 2021) and HotpotQA (Yang et al., 2018) for multi-hop QA; and LongBench-Reasoning (Bai et al., 2024) for long-context reasoning. This suite tests whether adaptive communication remains useful across symbolic, knowledge-intensive, verifier-based, multi-hop, and long-context settings.

Baselines. We compare against single-model reasoning methods, static multi-agent protocols, and adaptive multi-agent systems. Single-model baselines include direct inference, CoT (Wei et al., 2022), Self-Consistency (Wang et al.), Tree-of-Thought (Yao et al., 2023), and Graph-of-Thought (Besta et al., 2024). Static multi-agent baselines include majority voting, chain, star, full debate, and AutoGen-style solver–critic (Wu et al.). Adaptive baselines include DyLAN (Liu et al., 2023), GPTSwarm (Zhuge et al., 2024), G-Designer (Zhang et al.), AGP (Li et al., 2025), AMAS (Leong et al., 2025), and Guided Topology Diffusion (Jiang et al., 2025).

Base models and implementation. Unless otherwise stated, all methods use the same base LLM, temperature schedule, parser, verifier, maximum agent pool size, and prompt roles. The default base model is Qwen-3.5-9B, with $K{=}6$ agents. GAMMA uses a single multi-head graph-policy architecture within each reported setting: it predicts agent contributions and candidate-edge gains from independent traces, selects communication edges under the specified budget, and uses predicted node contributions to weight the final aggregation. This implementation follows the framework in Section 4 and is reported simply as GAMMA throughout the experiments.

5.2 OVERALL PERFORMANCE

Table 1 reports accuracy and online inference token usage across all nine datasets. GAMMA achieves 75.0% macro-averaged accuracy, improving over AGP by 2.3 percentage points while using fewer online inference tokens (2 860 vs. 3 110). Relative to the evaluated adaptive multi-agent

Table 2: Ablations of GAMMA supervision components, macro-averaged across all nine datasets.

Variant	Avg. Acc. (%) ↑	Avg. Tokens ↓	Wrong Cons. (%) ↓	Density ↓
Full GAMMA	75.0	2 860	6.1	0.23
No Myerson loss	73.2	2 980	8.0	0.25
No edge potential loss	73.5	3 760	8.5	0.37
No sparsity regularizer	74.6	4 710	6.6	0.46
Coalition value w/o I_x^+	73.8	2 900	7.1	0.24
Reward-only GNN	72.6	3 670	8.9	0.39
Similarity-only edges	72.9	3 210	8.3	0.34

Table 3: Allocation diagnostics macro-averaged across all nine datasets. Density is the fraction of possible edges selected. Correction rate is the fraction of Single errors fixed by the multi-agent system. Error propagation is the fraction of Single-correct instances flipped to incorrect. Edge ECE is a binned calibration error comparing predicted edge gains with held-out game-derived edge targets.

Method	Density ↓	Corr. Rate (%) ↑	Error Prop. (%) ↓	Wrong Cons. (%) ↓	Edge ECE ↓
Full Debate	1.00	17.8	13.1	14.2	—
G-Designer	0.38	23.9	6.5	9.0	0.145
AGP	0.31	25.7	6.0	8.2	0.131
Guided Top. Diff.	0.35	24.5	6.2	8.6	0.138
GAMMA	**0.23**	**30.8**	**4.1**	**6.1**	**0.076**

baselines, GAMMA therefore improves both average accuracy and inference-time token usage. The improvement is consistent across the evaluated task categories. Relative to AGP, GAMMA gains 1.6 points on GSM8K and 2.4 points on MATH, 2.2 points on MMLU and 2.8 points on GPQA-Diamond, and also improves both code-generation benchmarks, both multi-hop QA benchmarks, and LongBench-Reasoning. Sections 5.3–5.5 analyze how these aggregate differences relate to node-level supervision, edge-level supervision, graph sparsity, and error propagation.

Full debate uses substantially more communication yet achieves lower average accuracy than several sparser alternatives, together with a higher measured error-propagation rate in Section 5.4. Adaptive baselines already allow communication structure to vary by instance; GAMMA differs by introducing separate game-derived supervision targets for agent influence and candidate-edge value.

5.3 Ablation: What Makes GAMMA Work?

Table 2 shows that GAMMA's supervision components provide complementary signals. Removing Myerson supervision reduces accuracy from 75.0% to 73.2% and increases wrong consensus from 6.1% to 8.0%, supporting the usefulness of explicit agent-level contribution supervision for final aggregation. Removing edge-potential supervision reduces accuracy to 73.5% and increases graph density from 0.23 to 0.37, supporting the role of an explicit edge-value target in communication selection. Removing the useful uncertainty term I_x^+ reduces average accuracy to 73.8%, providing evidence that uncertainty reduction contributes information beyond task-quality supervision alone. The remaining ablations show that generic sparsification or a graph-level reward alone does not reproduce the full model's behavior. Without the sparsity regularizer, accuracy decreases only modestly from 75.0% to 74.6%, while online token usage increases from 2 860 to 4 710 and graph density doubles from 0.23 to 0.46, indicating that the sparsity term primarily controls communication usage. Reward-only and similarity-only variants also perform worse than the full model, suggesting that the explicit node/edge decomposition provides useful supervision beyond a graph-level target or semantic similarity alone.

5.4 Diagnostics of Game-Theoretic Allocation

Table 3 shows that GAMMA selects a substantially sparser graph while attaining a higher correction rate and a lower error-propagation rate than the evaluated baselines. Together with the ablation results, this pattern suggests that the observed sparsity is not explained by generic edge removal alone; the learned value signals help determine which communications are retained. The contrast with full debate

Table 4: Calibration of learned game-derived targets.

Model	Edge ECE ↓	Edge Spearman ↑	Node ECE ↓	Node Spearman ↑
Reward-only GNN	0.156	0.361	0.174	0.329
Similarity-only edges	0.149	0.388	0.161	0.347
No Myerson loss	0.098	0.531	0.152	0.366
No edge potential loss	0.143	0.397	0.103	0.518
GAMMA	**0.076**	**0.589**	**0.091**	**0.563**

Table 5: Robustness under degraded or adversarial agents. "Clean" uses the default agent pool. "One weak" replaces one agent with a weaker model. "One adversarial" and "Two adversarial" introduce agents that systematically argue for incorrect answers with high confidence. Accuracy is macro-averaged across all datasets.

Method	Clean ↑	One weak ↑	One adversarial ↑	Two adversarial ↑
Full Debate	66.5	64.3	58.1	52.4
G-Designer	72.0	70.3	67.1	63.0
AGP	72.7	71.0	68.0	64.5
GAMMA	**75.0**	**74.1**	**72.3**	**69.0**

is consistent with the same interpretation from the opposite direction. Dense communication gives every agent broad message access, while the measured error-propagation rate is higher. GAMMA instead retains edges according to predicted cost-adjusted marginal value. Together, these diagnostics are consistent with the intended role of value-guided sparse communication: retaining corrective interactions while limiting error propagation.

Table 4 evaluates predicted edge gains against held-out edge-potential targets and predicted node contributions against held-out Myerson targets. ECE measures calibration error, while Spearman measures rank correlation. GAMMA achieves the best node- and edge-level calibration and ranking performance among the evaluated variants. Removing either Myerson or edge-potential supervision degrades the corresponding prediction quality.

5.5 Robustness to Unreliable Agents

Table 5 evaluates GAMMA under heterogeneous or unreliable agent pools. Such heterogeneity can arise when agents differ in model capability, prompting, tools, or available knowledge, so equal influence is not always an appropriate default (Liu et al.). GAMMA degrades by 0.9 points with one weak agent, compared with 1.7 points for AGP and 2.2 points for Full Debate. Under two adversarial agents, GAMMA retains 69.0% accuracy, while Full Debate drops to 52.4%. The smaller degradation aligns with GAMMA's design: aggregation weights depend on predicted marginal contribution rather than confidence alone, while edge selection suppresses interactions assigned low cost-adjusted value.

6 Conclusion

We presented GAMMA, an offline-supervised framework for adaptive multi-agent message allocation in LLM reasoning. GAMMA separates two decisions that are often conflated in topology learning: how much influence each agent should receive and which communication edges should be executed. Node-level Myerson contributions and cost-aware conditional edge-value targets provide structured offline supervision, which is amortized into a lightweight online graph policy. Across nine reasoning benchmarks, GAMMA achieves higher average accuracy than the evaluated adaptive multi-agent baselines while using fewer online inference tokens. Ablations and diagnostic analyses support the benefits of separating node-level and edge-level supervision, with lower error-propagation and wrong-consensus rates.

Future work will extend GAMMA to distribution-shifted tasks, multi-round communication, tool-using agents, and settings without reliable task-level supervision. Sequential allocation and richer agent representations may further improve scalability and transfer.

AI USE STATEMENT

Generative AI tools were used only for language editing and readability. All technical content, experimental design, analysis, and conclusions were determined and verified by the authors, who take full responsibility for the manuscript.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 3119–3137, 2024.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 17682–17690, 2024.

Xiaohe Bo, Zeyu Zhang, Quanyu Dai, Xueyang Feng, Lei Wang, Rui Li, Xu Chen, and Ji-Rong Wen. Reflective multi-agent collaboration based on large language models. *Advances in Neural Information Processing Systems*, 37:138595–138631, 2024.

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. *arXiv preprint arXiv:2107.03374*, 2021.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, et al. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *The Twelfth International Conference on Learning Representations*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Forty-first international conference on machine learning*, 2024.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361, 2021.

Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *International conference on machine learning*, pp. 2242–2251. PMLR, 2019.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

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. *arXiv preprint arXiv:2103.03874*, 2021.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *The twelfth international conference on learning representations*, 2023.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations*.

Eric Hanchen Jiang, Guancheng Wan, Sophia Yin, Mengting Li, Yuchen Wu, Xiao Liang, Xinfeng Li, Yizhou Sun, Wei Wang, Kai-Wei Chang, et al. Dynamic generation of multi-llm agents communication topologies with graph diffusion models. *arXiv preprint arXiv:2510.07799*, 2025.

Hui Yi Leong, Yuheng Li, Yuqing Wu, Wenwen Ouyang, Wei Zhu, Jiechao Gao, and Wei Han. Amas: Adaptively determining communication topology for llm-based multi-agent system. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 2061–2070, 2025.

Boyi Li, Zhonghan Zhao, Der-Horng Lee, and Gaoang Wang. Adaptive graph pruning for multi-agent communication. *arXiv preprint arXiv:2506.02951*, 2025.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in neural information processing systems*, 36:51991–52008, 2023.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. Encouraging divergent thinking in large language models through multi-agent debate. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pp. 17889–17904, 2024.

Shixuan Liu, Yue He, Haotian Wang, Wenjing Yang, Yunfei Wang, Peng Cui, and Zhong Liu. Environment inference for learning generalizable dynamical system. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Shixuan Liu, Changjun Fan, Kewei Cheng, Yunfei Wang, Peng Cui, Yizhou Sun, and Zhong Liu. Inductive meta-path learning for schema-complex heterogeneous information networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):10196–10209, 2024.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. Dynamic llm-agent network: An llm-agent collaboration framework with agent team optimization. *arXiv preprint arXiv:2310.02170*, 2023.

Roger B Myerson. Graphs and cooperation in games. *Mathematics of operations research*, 2(3): 225–229, 1977.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, et al. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 15174–15186, 2024.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

Lloyd S Shapley et al. A value for n-person games. 1953.

Xu Shen, Yixin Liu, Yiwei Dai, Yili Wang, Rui Miao, Yue Tan, Shirui Pan, and Xin Wang. Understanding the information propagation effects of communication topologies in llm-based multi-agent systems. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 12358–12372, 2025.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345, 2024.

Mengdie Flora Wang, Haochen Xie, Guanghui Wang, Aijing Gao, Guang Yang, Ziyuan Li, Qucy Wei Qiu, Fangwei Han, Hengzhi Qiu, Yajing Huang, et al. From debate to decision: Conformal social choice for safe multi-agent deliberation. *arXiv preprint arXiv:2604.07667*, 2026.

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

Zhexuan Wang, Yutong Wang, Xuebo Liu, Liang Ding, Miao Zhang, Jie Liu, and Min Zhang. Agentdropout: Dynamic agent elimination for token-efficient and high-performance llm-based multi-agent collaboration. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 24013–24035, 2025.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First conference on language modeling*.

Yizhe Xie, Congcong Zhu, Xinyue Zhang, Tianqing Zhu, Dayong Ye, Minfeng Qi, Huajie Chen, and Wanlei Zhou. From spark to fire: Modeling and mitigating error cascades in llm-based multi-agent collaboration. *arXiv preprint arXiv:2603.04474*, 2026.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2369–2380, 2018.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

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 *ICLR 2025 Workshop on Foundation Models in the Wild*.

Heng Zhang, Yuling Shi, Xiaodong Gu, Zijian Zhang, Haochen You, Lubin Gan, Yilei Yuan, and Jin Huang. Hyperagent: Leveraging hypergraphs for topology optimization in multi-agent communication. *arXiv preprint arXiv:2510.10611*, 2025.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. Gptswarm: Language agents as optimizable graphs. In *Forty-first International Conference on Machine Learning*, 2024.

A Broader Impact and Limitations

Broader impact. GAMMA improves the reliability and online communication efficiency of multi-agent LLM reasoning in the evaluated settings, which may benefit applications in education, scientific research, and code generation. By reducing selected communication edges, it can reduce communication-related inference-token usage. However, more reliable multi-agent systems could be misused for generating more convincing disinformation or automating tasks that displace human labor. We encourage responsible deployment with appropriate safeguards.

Limitations. (1) GAMMA requires an offline label-generation stage involving additional multi-agent rollouts, game-value estimation, and graph-policy training. Accordingly, the token comparisons in the main experiments measure online inference usage and should not be interpreted as a complete accounting of end-to-end training and deployment cost. (2) The offline value construction assumes access to labeled training data or a sufficiently informative task-specific verifier. The quality of the

resulting Myerson and edge-value targets therefore depends on the quality and alignment of this supervision. (3) The game-derived labels are surrogate credit-assignment targets defined relative to our coalition value, graph-sampling distribution, and Monte Carlo approximation. They should not be interpreted as task-independent ground-truth causal effects of individual agents or messages. (4) Our experiments evaluate the policy within the benchmark settings described in Section 5.1. They do not establish zero-shot generalization to substantially different task distributions, agent pools, or real-world long-horizon workflows. (5) The current communication protocol is single-round. Sequential interaction would make edge utility explicitly dependent on previously executed messages and would require an extension of the current allocation model. (6) Myerson contribution estimation becomes increasingly expensive as the agent pool grows; although Monte Carlo sampling avoids exact enumeration, substantially larger pools may require more structure-aware estimators. (7) The current policy uses frozen sentence embeddings and a fixed set of agent roles. Richer reasoning representations and jointly learned role design may improve transfer but would introduce additional training complexity.

B Pareto Frontier Visualization

Figure 3 plots each method as a point in the accuracy–online-token space. The Pareto frontier is computed over the reported points. GAMMA lies above and to the left of the evaluated adaptive topology-learning baselines, while lower-cost methods such as majority voting occupy a different region of the accuracy–token plane.

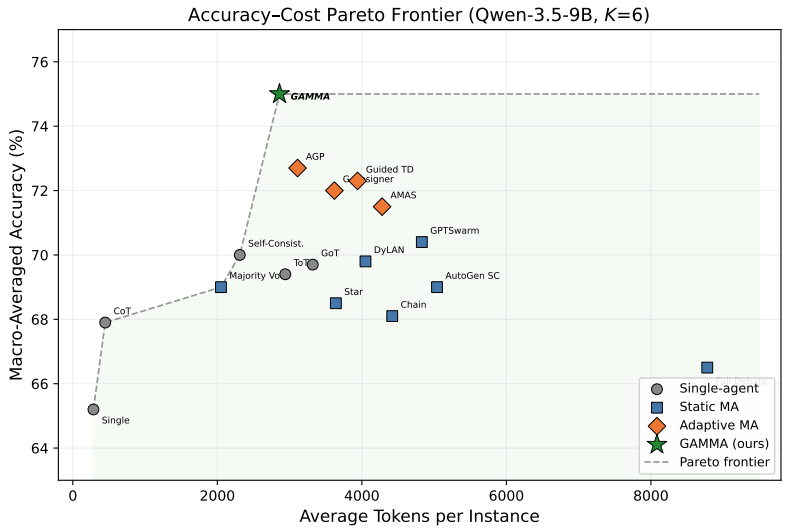


Figure 3: Accuracy versus online inference-token usage across all methods. Each point represents one method; accuracy and token usage are macro-averaged over all nine datasets. GAMMA lies above and to the left of the evaluated adaptive topology-learning baselines.

C Implementation Details

Implemented objective. The main implementation follows Eq. equation 10. The default training loss contains node Myerson regression, graph-conditional edge-value regression, and sparsity regularization:

$$\mathcal{L}_{\text{main}} = \lambda_\psi \frac{1}{|\mathcal{N}|} \sum_i \|\hat{\psi}_i - \widetilde{\psi}_i^G\|_2^2 + \lambda_e \frac{1}{|\mathcal{E}_{\text{cand}}|} \sum_{e_{ij}} \|\widehat{\Delta\Phi}_{ij} - \Delta\Phi_x(e_{ij} \mid G)\|_2^2 + \lambda_{sp} \frac{1}{|\mathcal{E}_{\text{cand}}|} \sum_{e_{ij}} p_{ij}. \tag{13}$$

Graph potential is used only as an offline intermediate for constructing graph-conditional edge targets. It is not used as a separate prediction loss.

Hyperparameters. The default loss weights are $\lambda_\psi = 1.0$, $\lambda_e = 1.0$, and $\lambda_{sp} = 0.01$. The coalition value weight is $\alpha = 0.3$. The graph-level communication cost weight is $\lambda_c = 0.1$. Monte Carlo Myerson estimation uses $M = 100$ permutations. The edge-value threshold is $\tau_\Phi = 0.0$, so selected edges must have positive predicted cost-adjusted gain before budget constraints are applied. Hard node pruning is not part of the default inference procedure; node scores are used for contribution-weighted aggregation.

Training details. Offline label generation uses eight graph configurations per task instance: an empty graph, a chain, a star, a fully connected graph, two independently sampled sparse graphs, and two value-guided sparse graphs produced by preliminary graph policies. The value-guided graphs are used only to diversify the offline rollout distribution.

Agent prompt design. Each of the K=6 agents receives a shared task instruction appended with an agent-specific role suffix (e.g., "`You are a careful step-by-step reasoner`", "`You are a devil's-advocate critic`"). Roles are drawn from a pool of 12 templates; the default selection assigns two step-by-step solvers, one planner, one critic, one verifier, and one diversity-seeker. Temperature is set to 0.7 for all agents except the verifier, which uses 0.3 for more deterministic judgments.

Communication protocol. When an edge e_{ij} is selected, agent j receives a structured message containing agent i's parsed answer, confidence, and a rationale field capped at 256 tokens. All selected messages are constructed from the independent pre-communication traces, and each recipient updates once after receiving its selected incoming messages. Thus, the current implementation uses a single parallel communication round rather than sequential edge-by-edge graph construction or multi-round message propagation.

D GNN Architecture Details

The graph policy f_θ is a 3-layer Graph Attention Network (GAT) with the following specifications.

Agent features (dimension 768). For each agent i we concatenate: (i) a 512-dimensional sentence embedding of the agent's rationale r_i obtained from a frozen Sentence-T5-Base encoder, (ii) a 128-dimensional embedding of the parsed answer y_i (hash-bucketed for discrete answers; a small MLP projection for numerical answers), (iii) a 64-dimensional vector of scalar features including self-reported confidence c_i, verifier score q_i when available, token count τ_i, answer-distribution entropy across all agents, and binary indicators for answer agreement with the current plurality, and (iv) a 64-dimensional learned role embedding. The sentence encoder is frozen and serves only as a fixed textual representation; task-specific credit prediction is learned by the GNN and its prediction heads.

Edge features (dimension 128). For each candidate edge e_{ij}, we form a 64-dimensional MLP projection of the scalar pairwise features—cosine similarity between rationale embeddings, binary answer agreement, absolute confidence difference $|c_i - c_j|$, and estimated message cost—and concatenate it with a 64-dimensional bilinear interaction $\mathbf{W}[\mathbf{h}_i \odot \mathbf{h}_j]$.

Task features (dimension 128). A frozen Sentence-T5-Base embedding of the task prompt, projected to 128 dimensions. This vector is concatenated to every 768-dimensional agent feature before the first GAT layer, giving an 896-dimensional input representation.

Prediction heads. After 3 GAT layers (hidden dimension 256, 4 attention heads, ELU activation, dropout 0.1), the model produces:

- **Node head**: a 2-layer MLP $\mathbb{R}^{256} \to \mathbb{R}^{128} \to \mathbb{R}$ predicting $\hat{\psi}_i$.
- **Edge head**: a 2-layer MLP operating on $[\mathbf{h}_i \| \mathbf{h}_j \| \mathbf{e}_{ij}]$ predicting $\widehat{\Delta\Phi}_{ij}$; the soft selection score is derived as $p_{ij} = \sigma(\widehat{\Delta\Phi}_{ij})$.

Total trainable parameters: approximately 2.4 M.

E SENSITIVITY ANALYSIS

E.1 SENSITIVITY TO NUMBER OF AGENTS K

Table 6: Effect of agent pool size K on GAMMA, macro-averaged across all nine datasets.

K	Avg. Acc. (%)	Avg. Tokens	Density	Corr. Rate (%)	Error Prop. (%)
3	71.8	1 620	0.31	24.2	4.8
4	73.1	2 080	0.27	27.1	4.5
6	75.0	2 860	0.23	30.8	4.1
8	75.6	3 540	0.19	31.9	3.9
10	75.8	4 280	0.16	32.3	3.8

Table 6 shows that accuracy improves with K but saturates around $K{=}8$. GAMMA reduces the observed selected-edge density as K increases, mitigating the communication growth of dense protocols over the evaluated range. The accompanying increase in correction rate is consistent with the larger agent pool providing additional opportunities for complementary hypotheses. We use $K{=}6$ as the default operating point in the main experiments.

E.2 SENSITIVITY TO COST WEIGHT λ_c

Table 7: Effect of the cost weight λ_c on GAMMA. Setting $\lambda_c{=}0$ removes the communication-cost term from the offline edge target; larger values place greater penalty on communication.

λ_c	Avg. Acc. (%)	Avg. Tokens	Density
0.00	74.5	4 920	0.48
0.01	74.8	3 870	0.35
0.05	74.9	3 210	0.28
0.10	75.0	2 860	0.23
0.20	74.6	2 340	0.17
0.50	73.4	1 780	0.11
1.00	71.2	1 320	0.06

Within the evaluated settings, $\lambda_c{=}0.1$ provides a favorable balance between average accuracy and online token usage. Setting $\lambda_c{=}0$ yields similar accuracy at substantially higher token usage, while $\lambda_c{\geq}0.5$ is associated with lower density and lower accuracy. The smooth change across the evaluated range suggests that the method is not highly sensitive to small perturbations around the default value.

E.3 SENSITIVITY TO SPARSITY WEIGHT λ_{sp}

Table 8: Effect of the sparsity weight λ_{sp}.

λ_{sp}	Avg. Acc. (%)	Avg. Tokens	Density
0.000	74.6	4 710	0.46
0.005	74.9	3 440	0.30
0.010	75.0	2 860	0.23
0.020	74.7	2 410	0.18
0.050	73.8	1 890	0.12
0.100	72.1	1 480	0.07

We use $\lambda_{sp} = 0.01$ as the default in the reported experiments. The sparsity regularizer interacts with λ_c, but the two act at different stages: λ_c shapes the offline cost-adjusted edge target, whereas λ_{sp} directly discourages large soft edge-selection scores in the learned policy.

Table 9: Effect of the number of Monte Carlo permutations M used for Myerson estimation during offline label generation. Online inference is unaffected.

M	Avg. Acc. (%)	Node Spearman	Label Time (rel.)
10	74.0	0.498	0.12×
25	74.4	0.531	0.27×
50	74.8	0.552	0.52×
100	75.0	0.563	1.00×
200	75.1	0.568	1.98×
500	75.1	0.571	4.93×

E.4 Sensitivity to Monte Carlo Permutations M

Within the evaluated range, M=100 provides a favorable trade-off between label stability and computation. Beyond M=100, the reported node-rank correlation and downstream accuracy change only marginally.

F Failure Prediction from Amortized Game Signals

Table 10: Failure prediction performance using inference-time signals. AUC and F1 are computed for the binary task of predicting whether the system's final answer is incorrect.

Signal	AUC	F1	Precision	Recall
Vote margin	0.621	0.337	0.284	0.415
Mean confidence	0.598	0.312	0.261	0.389
Answer entropy	0.643	0.358	0.302	0.441
Mean selected edge value	0.681	0.394	0.343	0.462
Node-weight concentration	0.662	0.376	0.319	0.458
Potential margin + entropy	**0.731**	**0.438**	**0.389**	**0.501**

Although GAMMA is trained for message allocation, its predicted game quantities also provide reliability signals. Vote margin and mean confidence are weak indicators because they treat agreement as correctness. This is precisely the assumption that often fails in multi-agent systems: several agents can agree because they share the same flawed rationale.

In these diagnostics, game-derived signals are more informative than vote margin or mean confidence for predicting final-answer failure. Low predicted edge value, unstable contribution concentration, and high answer entropy are associated with a higher error rate. These observations suggest that the learned allocation signals could also be explored as triggers for downstream verification or escalation; evaluating such escalation policies is left to future work.

G Error Taxonomy and Qualitative Case Studies

To understand when GAMMA succeeds and fails, we manually inspect 100 randomly selected instances from MATH and GPQA-Diamond (50 from each) and classify the outcomes into six categories.

Case study 1: Successful correction (MATH). A competition algebra problem requires factoring $x^4 - 6x^2 + 1$. Four agents attempt direct expansion or substitution; two arrive at the correct factored form. One agent (Agent 3) initially produces the wrong sign in an intermediate step but reaches a partially correct structure. Agent 5, a critic-role agent, identifies the sign error. GAMMA assigns Agent 5 a high Myerson contribution (0.38) despite low self-reported confidence (0.45), and the edge $e_{5\to3}$ receives the highest predicted edge value (+0.21). After communication, Agent 3 corrects its sign and the weighted vote converges to the correct answer.

Table 11: Error taxonomy over 100 inspected instances. Each instance is classified by whether GAMMA answered correctly and the mechanistic cause.

Category	Count	Description
Correct via pruning	34	All agents initially correct; GAMMA correctly prunes redundant edges, saving tokens.
Correct via correction	22	At least one agent initially wrong; GAMMA selects the corrective edge and upweights the contributor.
Correct despite noise	11	Communication introduces minor perturbation but contribution-weighted voting still selects the right answer.
Wrong: all agents wrong	21	No agent finds the correct answer independently; no communication can help.
Wrong: correct agent down-weighted	8	One agent has the correct answer but receives low predicted Myerson contribution, and its edge is pruned.
Wrong: error propagation	4	A wrong agent's message contaminates a previously correct agent through a selected edge.

Case study 2: Failure mode—correct agent pruned (GPQA). A graduate-level chemistry question about reaction stereoselectivity is answered correctly by only Agent 2. Agent 2's rationale is short and uses terminology that differs from the majority cluster, and its rationale embedding consequently has low similarity to those of the other agents. The GNN assigns Agent 2 a moderate Myerson score (0.12), while the edge connecting Agent 2 to the majority cluster receives a negative predicted potential (-0.04) and is pruned; the majority's wrong answer then prevails. This example illustrates a failure associated with an underrepresented reasoning pattern and motivates richer semantic representations, without by itself establishing which individual feature caused the prediction error.

Case study 3: No agent correct (MATH Level 5). On a difficult number-theory problem, all six agents produce incorrect answers through distinct but equally flawed reasoning paths. GAMMA assigns low predicted edge values on average (mean: -0.08) and uses minimal communication (density: 0.07), limiting online token usage. While the final answer is still wrong, the sparse graph avoids the error-amplification observed in full debate, where agents converge on one particular wrong answer with high confidence.

H Algorithms

This appendix provides the procedural details of GAMMA. The purpose of these algorithms is to make explicit how the game-theoretic quantities introduced in Section 4 are instantiated in practice and how they are separated between an offline supervision stage and an online inference stage. GAMMA does not compute cooperative-game values at test time. Instead, it first uses offline rollouts to estimate node-level and edge-level credit signals, and then distills these signals into a lightweight graph policy that can be executed efficiently during inference.

We provide two algorithms. Algorithm 1 describes the offline game-supervised label generation and policy training procedure. This stage is allowed to use gold labels or strong verifier feedback, multiple sampled communication graphs, and Monte Carlo estimates of Myerson-style marginal contributions. Its output is a supervised training set containing independent traces, graph-conditioned outcomes, node contribution targets, and graph-conditional edge-value targets. Algorithm 2 describes the online inference procedure. At test time, GAMMA observes only the new question and the independently generated agent traces, predicts amortized estimates of the offline game-derived targets with the trained policy f_θ, selects a sparse communication graph, and aggregates the final answer using predicted contribution weights.

H.1 Offline Game-Supervised Label Generation and Policy Training

Algorithm 1 outlines the offline phase. For each training task x, GAMMA first asks all agents to solve the problem independently. These independent traces provide the pre-communication evidence from which node, edge, and task features will later be constructed. Importantly, the independent

traces are collected before any inter-agent communication occurs, so that the learned policy can be applied at inference time without requiring expensive exploratory communication.

GAMMA then samples a diverse set of candidate communication graphs from the graph sampler Γ. These graphs include sparse, dense, random, and value-guided structures, allowing the offline procedure to observe how different communication patterns affect answer quality, verifier scores, and token cost. For each sampled graph G, the agents communicate according to the edges in G, produce graph-conditioned states, and return a final aggregated answer. The resulting rollouts provide the empirical outcomes from which the predefined node- and edge-level credit targets are estimated.

The next steps convert these rollouts into game-supervised labels. For Myerson estimation, predecessor coalitions are induced directly by the M sampled agent permutations in Eq. equation 7; no exhaustive coalition enumeration is required.

Given the graph-restricted value function, GAMMA estimates node-level Myerson contribution targets $\{\widetilde{\psi}_i^G\}_{i=1}^K$ using Eq. equation 7. GAMMA also estimates graph-conditional edge targets $\{\Delta\Phi_x(e_{ij} \mid G)\}_{e_{ij}\in\mathcal{E}_{\text{cand}}}$ using Eq. equation 9. These quantities are defined relative to the chosen value construction and the sampled graph context. Because the online policy observes only independent pre-communication traces, repeated supervision across graphs sampled from Γ amortizes these context-dependent targets into predictions that approximate their conditional mean under the offline graph-sampling distribution.

After all training tasks and sampled graphs have been processed, the saved traces and labels are used to train the graph policy f_θ with the compact objective in Eq. equation 10. The policy learns node contribution scores and edge-gain predictions; soft edge-selection scores are derived from the predicted gains. This training procedure amortizes the expensive offline game computation into a deployable inference-time model.

Algorithm 1 Offline game-supervised label generation and policy training for GAMMA

Require: Training tasks $\mathcal{D}_{\text{train}}$, agent pool $\mathcal{A}$, graph sampler Γ

for task $x \in \mathcal{D}_{\text{train}}$ **do**
  Collect independent agent traces $\{s_i\}_{i=1}^K$
  for candidate graph $G \sim \Gamma(x)$ **do**
    Run multi-agent communication under G
    Store post-communication states, final answer, verifier score, and token cost
    Sample M agent permutations and evaluate the coalition values required by their predecessor sets using Eq. equation 2
    Convert coalition values into graph-restricted values $\widehat{v}_x^G(S)$ using Eq. equation 5
    Estimate node targets $\{\widetilde{\psi}_i^G\}_{i=1}^K$ using Eq. equation 7
    Estimate edge targets $\{\Delta\Phi_x(e_{ij} \mid G)\}_{e_{ij}\in\mathcal{E}_{\text{cand}}}$ using Eq. equation 9
    Save traces, node labels, edge labels, and communication costs
  end for
end for
Train graph policy f_θ with Eq. equation 10

H.2 Amortized Online Inference

Algorithm 2 summarizes the online phase. The key difference from the offline phase is that GAMMA no longer enumerates coalitions, samples multiple communication graphs, or computes Myerson values directly. All expensive game-theoretic computation has already been distilled into the trained graph policy f_θ. Therefore, online inference requires only one independent trace from each agent, one policy forward pass, one sparse communication round, and one contribution-weighted aggregation step.

Given a new question x', GAMMA first generates independent traces $\{s_i\}_{i=1}^K$. These traces contain each agent's initial answer, rationale, confidence or inference-available verifier signals when available, and token cost. From these traces, GAMMA constructs node features for individual agents, edge features for candidate pairwise interactions, and task features for the input question. The trained

policy f_θ predicts the amortized node contribution score $\hat{\psi}_i$ and edge gain $\widehat{\Delta\Phi}_{ij}$; the soft selection score p_{ij} is derived from the predicted edge gain.

The selected communication graph is formed by retaining edges with positive predicted cost-adjusted gain according to Eq. equation 11, subject to the communication budget. GAMMA then executes one sparse communication round along the selected edge set $\mathcal{E}^*$ rather than using dense debate or a fixed topology.

Finally, GAMMA aggregates the post-communication answers using contribution weights computed from Eq. equation 12. Agents with positive predicted contribution receive greater influence, while agents with negative or negligible predicted contribution are down-weighted. For discrete-answer tasks, this corresponds to contribution-weighted voting; for numerical or structured outputs, equivalent answers are normalized or verified before aggregation. The algorithm returns the final answer $\hat{y}$ together with the selected graph and the predicted node and edge scores, which can also be used for analysis or failure prediction.

Algorithm 2 Online game-supervised inference with GAMMA

Require: New question x', agents $\mathcal{A}$, trained policy f_θ

1: Generate independent traces $\{s_i\}_{i=1}^{K}$
2: Build node, edge, and task features from independent traces
3: Predict node contributions $\hat{\psi}_i$ and edge gains $\widehat{\Delta\Phi}_{ij}$; derive $p_{ij} = \sigma(\widehat{\Delta\Phi}_{ij})$
4: Select sparse high-value edges $\mathcal{E}^*$ using Eq. equation 11
5: Run multi-agent communication only along $\mathcal{E}^*$
6: Aggregate final answers using contribution weights from Eq. equation 12
7: Return final answer $\hat{y}$, selected graph G^*, node scores, and edge scores