BEYOND SOLO AND CONSISTENCY: VINDICATING MULTI-AGENT DEBATE VIA CONDITIONAL PROGRESSIVE PRUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large Language Model (LLM) based Multi-Agent Debate (MAD) is one of the most effective test time scaling techniques. Through multi-round communication, agents complement each other in knowledge and reasoning and solve tasks that no single member can solve. However, existing MAD frameworks fail to beat strong Single Agent and Consistency-based baselines under the same strict cost limit, which shakes the foundation of the MAD field. We propose Conditional Progressive Pruning (CPP), a lightweight pruning framework that fully exploits multi-round MAD. CPP outperforms all existing MAD frameworks on multiple dominated benchmarks. It is also the first to fully outperform consistency methods. Our code, detailed Agent interaction records will be released soon.

1 INTRODUCTION

In today's AI field, the trend from individual intelligence to collective intelligence is becoming clearer (Du et al., 2023), making Multi-Agent Collaboration one of the most popular research directions. Previous work has shown that collaboration is effective if and only if external information exists. Therefore we focus on the setting of heterogeneous LLM teams. The first type of collaboration work assigns different roles and duties to multiple agents (Liu et al., 2023). These works essentially study task planning and distribution. Each agent still solves a single-step subtask alone. The second type of collaboration work can be called Multi-Agent Debate (MAD). In MAD, multiple agents are asked to solve the same problem together through multi-round communication (Li et al., 2026b). Different LLMs complement each other in knowledge and reasoning. Thanks to this, a correct solution can emerge during communication even if no team member has a fully correct view at the start (Choi et al., 2026). This means MAD cannot simply be seen as a process where strong agents persuade weak ones. In fact, the advantage of multi-round communication is not stable (Huang et al., 2024). Under the same computational cost, existing MAD frameworks can hardly fully outperform strong consistency-based baselines. Consistency methods first let different LLMs solve the problem independently. They then use majority voting to get the answer. As the number of agents increases, these methods show a stable trend of performance growth (Li et al., 2024). However, consistency methods are only independent repeated sampling. They are a communication-free, one-dimensional form of cooperation and their performance will eventually converge (Zhang et al., 2023). In contrast, agents in MAD can essentially improve reasoning through communication. So we believe the potential of MAD has not been fully exploited yet. That is why the value of the MAD field is still debated. Existing MAD frameworks generally have the following drawbacks:

1) Communication cost is counted only by the number of tokens, without normalizing the cost across different LLMs, so the cost is optimistically estimated.
2) Did not make full use of the intermediate results of multi-round discussion for pruning, and they do not properly model the behavior of different agents to enable personalized information passing.
3) They do not design reinforcement learning algorithms specifically for the debate setting, which leaves room for optimization.

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.

To address the above challenges, we propose Conditional Progressive Pruning (CPP), a lightweight pruning framework. Previous MAD methods make decisions only based on the interactions between LLM profiles (Chai et al., 2024). Unlike them, CPP is the first to model the conditional probability between earlier and later actions in an autoregressive way. It also performs progressive pruning in the inference pipeline directly based on the results between rounds of MAD. This allows CPP to make full use of all the information produced in MAD for decision making, and thus shows the advantage of communication-based collaboration. The only work similar to ours is AnyMAC (Wang et al., 2025a), which also successfully uses intermediate results to improve MAD performance. However, it does not model the relations between actions within and across rounds. Overall, our contributions can be summarized in the following three points:

- To the best of our knowledge, we are the first to propose a framework that autoregressively prunes MAD based on intermediate results. It is also the first to fully outperform consistency methods and single agents under strictly the same computational cost. This ends the debate on whether MAD is valuable and vindicates the significance of the MAD field.
- We design a reinforcement learning algorithm for multi-round MAD. The proposed tailored reward and Agent Behavior Embedding can generally improve the convergence performance of MAD.
- Extensive experiments on MATH-Hard, GPQA and MMLU-Pro show that our CPP becomes the SoTA MAD framework, with consistent advantages in robustness, efficiency and scalability. We will also release the source code and detailed agent debate logs.

2 Related Work

2.1 Heuristic Multi-Agent Debate

Homogeneous and Heterogeneous Consistency Methods improve performance as the number of agents scales, through independent sampling and majority voting (Wang et al., 2022). They are a fully communication-free form of cooperation. Methods such as IoE (Madaan et al., 2023), PHP (Zheng et al., 2023) and Reflexion (Shinn et al., 2023) let an agent discuss with itself over multiple rounds to reach a solution, but they are very unstable without external feedback (Choi et al., 2026). (Du et al., 2023) first proposes a static MAD framework in which agents are fully connected. Methods such as LLM-Intervention and S^2MAD (Zeng et al., 2025) then try to prune dynamically based on the quality and diversity of debate responses. Some works also let LLMs select information by themselves in a zero-shot way (Kim et al., 2024), but this greatly increases the cost.

2.2 Learning-based Multi-Agent Debate

GPTSwarm (Zhuge et al., 2024) is the first to introduce reinforcement learning for dynamic pruning in MAD. It formulates each communication edge as a single parameter, and works such as AgentPrune (Zhang et al., 2025) and AgentDropout (Wang et al., 2025b) are essentially the same. G-Designer (Zhang et al., 2024) is the first to introduce an extra lightweight pruning model to optimize the flow, but its decisions rely only on the interaction between the profiles of two agents. MASGPT (Ye et al., 2025) is the first to generate MAD topologies in an autoregressive and generative way under supervised learning. ARG-Designer (Li et al., 2026a) later realizes this paradigm better through reinforcement learning. However, both of them rely only on agent embedding information. AnyMAC (Wang et al., 2025a) innovatively filters the results between rounds directly to achieve independent pruning. However, it does not consider the conditional probability between actions, and it does not explore reinforcement learning algorithms designed for debate. MasHost (Yang et al., 2025b) designs a reward for the debate setting, which gives rewards based on the change of the MAD state before and after an action. However, its numerical design does not fit the multi-round MAD setting. For agent embedding modeling, Symbolic-MoE (Chen et al., 2025) builds on text-introduction based profile initialization and adds statistical information. This is the simplest form of agent behavior modeling.

3 CPP: CONDITIONAL PROGRESSIVE PRUNING

3.1 INFERENCE PIPELINE

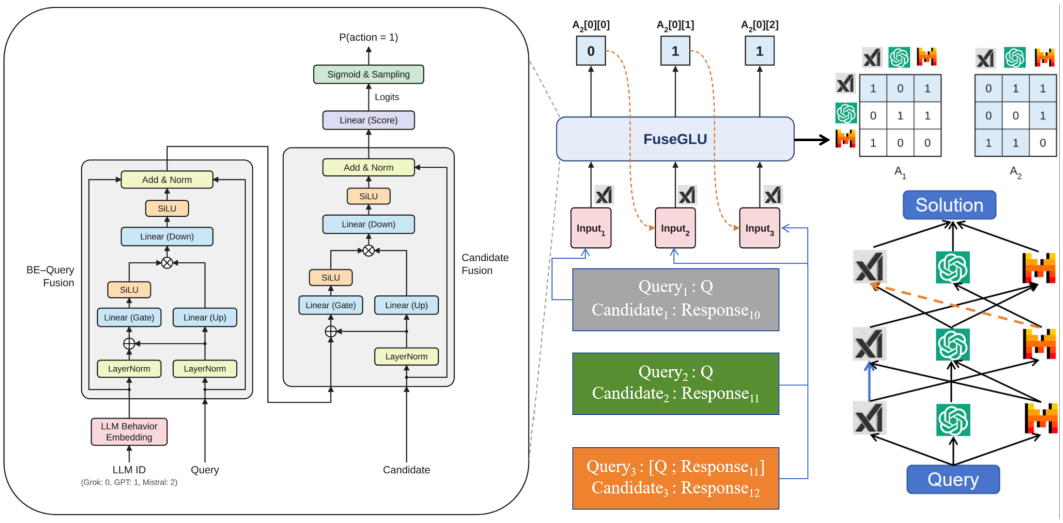


Figure 1: Overview of our CPP pipeline. FuseGLU takes the LLM behavior embedding, the query and the candidate response as input, and autoregressively decides each action by combining the available information with the candidate conditioned on previous actions.

3.1.1 AGENT BEHAVIOR MODELING

Previous works usually initialize agent embeddings with text-introduction profiles. In contrast, our behavior embedding is obtained by directly encoding the responses of each agent on specific questions from the validation set, and it is then trained together with CPP. This initialization directly models the concrete behavior of each agent, so it carries richer information than a general profile description.

3.1.2 CONDITIONAL PRUNING: FUSEGLU

Existing pruning models usually take the query as the main input and apply simple projections to it. In this case, the query is transformed with a low degree of nonlinearity, and the agent identity and the candidate response only interact with it in a shallow way. To address this, we propose FuseGLU, a variant of the Gated Linear Unit (GLU). Instead of transforming the query alone, FuseGLU fuses the agent behavior embedding and the candidate response into the gate and up projections (Yang et al., 2025a). This lets the extra information directly control how the query is transformed and increases the nonlinearity of the model. FuseGLU consists of two fusion blocks with the same form. For a main input $\mathbf{x}$ and a fused input $\mathbf{y}$, both after layer normalization, a fusion block is defined as

$$\mathrm{Fuse}(\mathbf{x},\mathbf{y}) = \mathrm{Norm}\Big(\mathbf{x} + \mathrm{SiLU}\big(W_d\big[\mathrm{SiLU}\big(W_g(\mathbf{x}+\mathbf{y})\big) \odot W_u\,\mathbf{y}\big]\big)\Big), \tag{1}$$

where W_g, W_u and W_d are the gate, up and down projections, and $\odot$ is element-wise multiplication. The first block fuses the behavior embedding $\mathbf{e}_i$ of the receiving agent a_i with the query, and the second block fuses the result with the candidate response. The output is mapped to a score and gives the probability of keeping the edge from a_j to a_i in round t,

$$p_\theta\big(a^t_{i,j}=1\big) = \sigma\Big(\mathbf{w}_s^\top\,\mathrm{Fuse}\big(\mathrm{Fuse}(\mathbf{e}_i,\mathbf{q}^t_{i,j}),\,\mathbf{r}^{t-1}_j\big)\Big), \tag{2}$$

where $\mathbf{r}^{t-1}_j$ is the response of a_j in the previous round and $\mathbf{q}^t_{i,j}$ is the current query.

The key of CPP is that the query is not fixed. Each agent first decides whether to keep its own previous response, and then decides on the other agents in a fixed order. Every response that has

already been selected is concatenated to the original question, and the result becomes the query $\mathbf{q}_{i,j}^t$ for the next decision. In this way, each action is conditioned on the earlier actions of the same agent in the same round. Since the responses of round t also depend on the edges chosen in round $t-1$, actions are linked across rounds as well. The whole pruning policy is therefore modeled autoregressively as

$$\pi_\theta(\mathcal{G}) = \prod_{t=1}^{T} \prod_{i=1}^{N} \prod_{k=1}^{N} p_\theta\big(a_{i,o_i(k)}^t \,\big|\, a_{i,o_i(<k)}^t,\, \mathcal{H}^{t-1}\big), \tag{3}$$

where $\mathcal{G}$ is the debate graph over all rounds, $o_i(k)$ is the k-th agent in the decision order of a_i, and $\mathcal{H}^{t-1}$ is the debate history up to round $t-1$. This makes full use of the intermediate results of MAD when making each pruning decision.

3.2 Progressive Reinforcement Learning for MAD

3.2.1 Debate-Tailored Reward Design

We design a reward tailored for multi-round debate, as shown in Table 1. The sign pair indicates whether the answer of the current agent is correct before and after an action, and the action is 1 if the edge is kept and 0 if it is pruned. The scalar a controls the scale of the reward. We use a larger a in later rounds, since they are closer to the final result. When the answer is already correct $(++)$, further communication is not necessary. So pruning the edge is rewarded with a, while keeping it is rewarded with $k \cdot a$, where $k \in [-1, 1]$ is a coefficient that can make this reward either positive or negative. An action that turns a correct answer into a wrong one $(+-)$ receives $-2a$, and an action that turns a wrong answer into a correct one $(-+)$ receives $2a$. When the answer stays wrong $(--)$, both actions receive $-a/(N-1)$, where N is the number of agents in one round. This small penalty encourages agents to act in order to move from wrong to correct. Even if the answer stays wrong in all earlier actions of a round and only becomes correct at the last action, the accumulated reward of these actions is still positive.

Table 1: Multi-Round Debate Reward

Action \ Sign	+ +	+ -	- +	- -
1	$k \cdot a$	$-2a$	$2a$	$-a/(N-1)$
0	a	–	–	$-a/(N-1)$

3.2.2 Inter-Round Reward Accumulation

An action affects not only its own result but also the actions that follow it. Take the bold blue edge in the first round of Figure 1 as an example. This edge decides whether Grok keeps its own previous response. Its result is concatenated into the query of Grok's later decisions in the same round, so it affects Grok's choices on GPT and Mistral. It also changes Grok's response in this round, which becomes a candidate for the agents in the next round. The actions affected by this edge are marked with light blue backgrounds in A_1 and A_2. Therefore, the return of an action accumulates the rewards of all its subsequent related actions. We use two discount factors for this accumulation. β discounts rewards across rounds, and γ discounts rewards within a round. The subsequent actions of each action are listed in Appendix A. We then optimize the pruning policy with on-policy policy gradient,

$$\nabla_\theta J(\theta) = \mathbb{E}_{\mathcal{G}\sim\pi_\theta}\Big[\sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{k=1}^{N} G_{i,o_i(k)}^t \,\nabla_\theta \log p_\theta\big(a_{i,o_i(k)}^t \,\big|\, a_{i,o_i(<k)}^t,\, \mathcal{H}^{t-1}\big)\Big], \tag{4}$$

where $G_{i,o_i(k)}^t$ is the accumulated return of action $a_{i,o_i(k)}^t$ discounted by β and γ.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

We conduct experiments on benchmarks such as MATH-Hard (Hendrycks et al., 2021), MMLU-Pro (Wang et al., 2024) and GPQA-Diamond (Rein et al., 2023), which test the reasoning ability, the factuality ability and the combination of both in MAD, respectively. For MATH-Hard, we sample 150/100/100 problems of the highest difficulty level (Level 5) from MATH as the training/validation/test sets. The corresponding sample sizes are 150/100/100 for MMLU-Pro and 98/50/100 for GPQA. All main experiments are done with Grok-4.1-non-reasoning, GPT-4.1-nano and Mistral-small-3.2. Due to the deprecation of Grok 4.1, we switch to Grok-4.3 for the robustness experiments, the ablation studies on the reward and the pipeline, and the efficiency experiments. Our pruning model uses the proposed FuseGLU architecture, and all experiments are trained with on-policy reinforcement learning. The main hyperparameters we tune include the learning rate, batch size, entropy coefficient and entropy decay rate. More details will be provided in our code, which will be open-sourced soon. More detailed information about all baselines can be found in Section 2

4.2 MAIN RESULTS

4.2.1 BEYOND EXISTING MADS

Table 2 compares CPP with existing MAD methods, where the cost is normalized with respect to Full-Debate. CPP achieves the highest accuracy on all three benchmarks, reaching 68.17% on MATH-Hard, 68.67% on GPQA-Diamond and 66.33% on MMLU-Pro. At the same time, its cost stays below that of Full-Debate on every benchmark. The strongest baselines, ARG-Designer and AnyMAC, also use learned pruning, but CPP still outperforms both of them on all benchmarks, with the largest gain on MMLU-Pro. These results show that CPP outperforms all existing MAD frameworks.

Table 2: CPP Main Results: Comparison with MADs

Method	MATH-Hard		GPQA-Diamond		MMLU-Pro	
	Acc (%)	Cost	Acc (%)	Cost	Acc (%)	Cost
Grok	48.50 ± 1.38	11.17%	57.33 ± 4.27	13.50%	57.00 ± 2.90	12.05%
GPT	43.83 ± 4.79	7.56%	38.67 ± 3.78	6.78%	42.17 ± 3.43	6.49%
Mistral	46.17 ± 3.97	8.15%	45.33 ± 3.44	4.46%	57.33 ± 1.51	5.22%
Vote	55.00 ± 3.63	26.88%	58.67 ± 3.01	24.74%	60.67 ± 3.61	23.75%
IoE-Grok	56.83 ± 3.06	28.24%	65.83 ± 2.32	32.79%	58.83 ± 3.60	32.84%
IoE-GPT	48.50 ± 4.85	17.12%	45.33 ± 5.47	15.48%	47.67 ± 1.63	15.30%
IoE-Mistral	47.33 ± 3.44	20.83%	45.83 ± 3.76	11.21%	53.67 ± 2.66	11.84%
IoE-Vote	60.33 ± 3.72	66.19%	62.50 ± 1.97	59.48%	61.17 ± 3.31	59.98%
php-Grok	53.83 ± 1.94	27.68%	61.00 ± 2.61	28.85%	57.00 ± 2.00	26.24%
php-GPT	51.00 ± 3.46	17.83%	42.50 ± 4.04	14.62%	44.50 ± 3.08	14.17%
php-Mistral	50.83 ± 2.99	22.50%	40.50 ± 1.87	9.24%	57.83 ± 1.17	10.83%
php-Vote	59.33 ± 2.73	68.00%	60.17 ± 4.02	52.70%	60.50 ± 1.38	51.23%
Reflexion-Grok	54.17 ± 2.71	36.81%	66.83 ± 2.23	53.31%	62.17 ± 2.64	52.63%
Reflexion-GPT	47.17 ± 2.23	20.53%	49.33 ± 3.27	30.26%	47.83 ± 3.13	26.73%
Reflexion-Mistral	51.17 ± 1.33	16.95%	46.50 ± 3.62	20.35%	56.83 ± 1.47	22.13%
Reflexion-Vote	60.83 ± 3.66	74.29%	63.83 ± 3.19	103.92%	63.17 ± 2.40	101.48%
Full-Debate	65.67 ± 3.27	**100.00%**	67.50 ± 2.88	**100.00%**	63.67 ± 1.86	**100.00%**
Random	64.50 ± 5.05	91.39%	65.33 ± 4.08	91.90%	64.17 ± 4.26	90.01%
LLM-Self	64.00 ± 2.10	119.32%	66.50 ± 4.72	138.00%	61.83 ± 4.36	141.02%
Intervention	63.67 ± 2.66	91.97%	61.17 ± 3.97	94.28%	60.50 ± 2.88	90.89%
S^2-MAD	60.00 ± 2.53	96.72%	62.17 ± 4.62	93.89%	61.50 ± 2.07	92.30%
IoE-Debate	64.17 ± 3.87	92.01%	68.17 ± 2.79	92.24%	62.67 ± 1.51	93.97%
Debate-IoE	65.67 ± 2.58	94.24%	67.00 ± 3.63	90.16%	60.83 ± 4.36	93.37%
Zero	57.33 ± 2.73	84.93%	58.67 ± 3.27	81.50%	63.50 ± 2.88	79.37%
ARG-Designer	68.00 ± 4.38	99.58%	67.67 ± 3.27	97.09%	64.00 ± 4.38	86.90%
AnyMac	67.83 ± 2.32	95.94%	65.67 ± 2.80	94.40%	63.00 ± 2.53	90.27%
CPP	**68.17 ± 2.14**	98.61%	**68.67 ± 1.51**	97.63%	**66.33 ± 3.33**	99.43%

Beyond the top-line comparison, the table reveals several consistent patterns. Majority voting (Vote) already provides a sizable boost over any single model (e.g., 55.00% vs. at most 48.50% on MATH-Hard), but the self-refinement family (IoE, php, Reflexion) achieves further gains only at a steep cost premium—Reflexion-Vote, for instance, exceeds Full-Debate's own cost on both MATH-Hard (74.29%) and GPQA-Diamond (103.92%) while still trailing CPP in accuracy by 4–8 points. Among the fixed-topology MAD baselines, cost varies substantially (85%–141% across benchmarks) without a corresponding accuracy advantage: LLM-Self incurs the highest cost on two benchmarks yet remains below CPP in accuracy, and S^2-MAD combines moderate cost with the lowest accuracy in its group, indicating that neither self-determined nor heuristic pruning reliably identifies efficient communication structures.

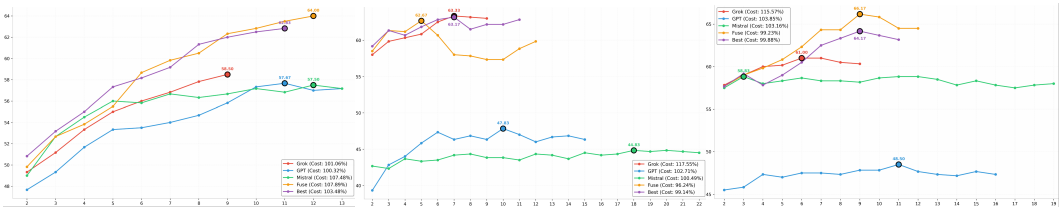
Comparing the two learned-pruning baselines with CPP further highlights the value of the proposed design. ARG-Designer is competitive on MATH-Hard (68.00%) but its accuracy drops on MMLU-Pro (64.00%) alongside a markedly lower cost (86.90%), suggesting over-aggressive pruning on that benchmark; AnyMAC is more cost-stable (90–96%) but consistently falls 0.3–3.3 points behind CPP in accuracy. CPP, in contrast, maintains both the highest accuracy and a cost consistently below Full-Debate (93.79–99.43%) across all three benchmarks, indicating that its pruning decisions generalize better across task distributions rather than overfitting to a specific benchmark's structure.

4.2.2 Beyond Extended Consistency

Table 3 compares MAD methods with consistency methods under the same computational cost. Rows named after a single model use homogeneous consistency, and all other rows use heterogeneous consistency. Our debate has 3 agents and 3 rounds, which corresponds to 9 samples. However, communication brings extra cost, so giving consistency methods only 9 samples would not be a fair comparison. We therefore extend the number of samples of consistency methods beyond 9 to match the cost of MAD. Consistency-Best extends the samples using only Grok, the strongest model, while Consistency-Fuse extends the samples of all models in equal proportion.

The last three rows with the prefix E denote the Extension setting. In the main experiments, each round contains one Grok, one GPT and one Mistral agent. In the Extension setting, we change the arrangement so that the first round uses only Mistral, the second round uses only GPT and the third round uses only Grok. MAD methods still beat consistency methods under this new arrangement, which shows that MAD has a large room for further optimization.

Most importantly, CPP is the only method that outperforms all consistency methods on every benchmark. On MMLU-Pro, CPP in Table 2 is the only MAD method that beats Consistency-Fuse. This remains true in the Extension setting, where E_CPP is still the only method that outperforms all consistency methods on MMLU-Pro.



(a) MATH (b) GPQA (c) MMLU

Figure 2: Detailed Consistency Results. Full results can be found in Appendix D

4.2.3 Beyond Scaled Chain of Thought

We also compare CPP with single agents that use more CoTs under a similar cost, as shown in Table 4. Adding more CoTs does not bring steady gains to single agents, and the best result among all settings is only 51.00% from Mistral with 9 CoTs. In contrast, CPP reaches 68.17% on MATH-Hard with a cost of 98.61%, which clearly outperforms single agents under the same computational cost.

Table 3: All Consistency Mode and MAD Struture Extension Results

Method	MATH-Hard		GPQA-Diamond		MMLU-Pro	
	Acc (%)	Cost	Acc (%)	Cost	Acc (%)	Cost
Consistency-Grok	58.50 ± 1.97	101.06%	63.33 ± 1.63	91.43%	61.00 ± 1.55	77.05%
Consistency-GPT	57.67 ± 1.03	84.88%	47.83 ± 1.60	68.47%	48.50 ± 1.05	71.40%
Consistency-Mistral	57.50 ± 1.97	99.21%	45.33 ± 3.44	4.46%	58.83 ± 1.72	16.29%
Consistency-Fuse	64.00 ± 2.45	107.89%	62.67 ± 2.16	56.88%	66.17 ± 2.32	78.94%
Consistency-Best	62.83 ± 2.14	103.48%	63.17 ± 2.48	74.85%	64.17 ± 3.06	86.30%
E_Full-Debate	**69.50 ± 2.59**	103.95%	66.33 ± 3.14	100.52%	66.00 ± 2.68	103.83%
E_ARG-Designer	68.17 ± 1.17	103.15%	64.00 ± 1.67	95.45%	62.83 ± 2.14	85.21%
E_CPP	67.17 ± 2.04	101.23%	**68.50 ± 1.38**	97.13%	**68.00 ± 2.28**	99.12%

Table 4: **Accuracy** (%) and **Cost** of single models across different numbers of CoTs on MATH-Hard

Model \ CoTs	0	6	9	15	21	27	36	48	60
Grok	48.17±2.93 5.65%	46.67±3.08 17.44%	46.50±2.59 24.68%	44.50±3.08 40.78%	44.33±2.66 54.26%	45.00±3.03 67.35%	44.67±1.86 86.47%	46.83±2.48 110.28%	–
GPT	28.17±2.14 3.69%	**46.83±2.64** 11.37%	41.83±4.36 14.92%	42.83±3.54 22.91%	42.83±2.64 29.53%	40.50±3.15 36.56%	41.83±4.12 46.74%	42.67±3.01 59.20%	41.33±4.72 76.52%
Mistral	18.17±4.58 2.41%	44.17±3.13 11.20%	**51.00±3.41** 16.58%	47.00±5.66 24.79%	49.67±2.25 33.53%	48.50±3.27 41.21%	47.50±3.02 51.00%	49.67±1.63 65.37%	47.67±2.94 81.50%

4.3 Ablation Study and Analysis

4.3.1 Adversarial Robustness

Table 5 reports results after Grok is updated to Grok-4.3, under both the vanilla and the adversarial settings. In the vanilla setting, CPP achieves the best accuracy of 65.17% with a lower cost than Full-Debate and ARG-Designer. In the adversarial setting, single agents and consistency methods drop sharply to around 20% or lower, while debate methods remain much more robust. Since CPP makes decisions based on intermediate results, it can be directly trained in the adversarial setting. In contrast, methods such as ARG-Designer only rely on agent profiles and cannot perform adversarial training. After adversarial training, Adv-CPP keeps the same accuracy of 65.17% in the vanilla setting, with a lower cost and a smaller variance. In the adversarial setting, its accuracy increases from 57.33% to 58.00%, which is the best among all methods.

Table 5: Adversarial Robustness

Method	Vanilla				Adversarial			
	Ref	Acc (%)	Cost	Edge	Ref	Acc (%)	Cost	Edge
Grok	–	41.17 ± 3.60	13.24%	–	–	9.83 ± 2.48	9.78%	–
GPT	–	43.83 ± 4.79	3.95%	–	–	4.67 ± 2.34	1.01%	–
Mistral	–	46.17 ± 3.97	4.26%	–	–	12.67 ± 2.07	1.07%	–
Vote	–	51.17 ± 3.31	21.45%	–	–	10.17 ± 1.72	11.86%	–
IoE-Vote	–	52.83 ± 3.97	51.66%	–	–	47.17 ± 4.26	52.34%	–
Consistency-Mistral	–	59.67 ± 2.25	94.62%	–	–	20.83 ± 1.47	16.41%	–
Consistency-Fuse	–	62.50 ± 0.84	94.94%	–	–	18.00 ± 1.26	54.67%	–
Consistency-Best	–	62.17 ± 1.72	94.81%	–	–	18.17 ± 1.60	37.76%	–
Zero*	0, 0, 0	52.50 ± 1.97	68.23%	0.00	0.5, 0.5, 0.5	***61.17 ± 2.93***	77.67%	9.00
Full-Debate	1, 1, 1	62.17 ± 1.94	**100.00%**	18.00	1, 1, 1	54.50 ± 1.64	78.28%	18.00
Debate-IoE	0.67, 0.67, 0.67	61.00 ± 2.53	87.39%	12.00	0.67, 0.67, 0.67	53.67 ± 2.16	62.67%	12.00
IoE-Debate	0.67, 0.67, 0.67	59.67 ± 3.14	79.36%	12.00	0.67, 0.67, 0.67	57.67 ± 2.58	72.64%	12.00
ARG-Designer	0.99, 0.99, 0.99	64.67 ± 4.55	101.55%	17.79	0.99, 0.99, 0.99	56.83 ± 2.32	78.78%	17.80
CPP	0.91, 0.87, 0.86	**65.17 ± 3.37**	93.79%	15.83	0.9, 0.86, 0.86	57.33 ± 2.66	77.49%	15.73
Adv-CPP	0.9, 0.86, 0.86	**65.17 ± 1.47**	91.50%	15.70	0.89, 0.85, 0.85	58.00 ± 2.97	78.11%	15.55

4.3.2 Reward Ablation with Efficiency Analysis

Reward ablation. The right part of Table 6 studies the design of the reward. R_+a is the setting used in our main experiments, where $k = 1$ is kept during the whole training. We first change the value of k from the start of training. As k decreases from 1 to 0.5, 0 and -1, the number of

kept edges drops from 15.83 to 12.70, 9.17 and 7.18, and the cost drops from 93.79% to 86.91%, 81.20% and 77.14%. This shows that k directly controls the sparsity of the debate graph. However, the accuracy also drops to 62.33%, 59.50% and 60.00%, which means that pruning too much from the start removes useful communication. We then remove the scale differences in the reward step by step. In w/o Round-Var., all rounds use the same scale a, while our main setting uses $a = 1$ in the first round and $a = 1.5$ in the second round. The accuracy drops from 65.17% to 64.67%. Vanilla ± 1 further removes the scale differences within a round and only gives rewards of $+1$ or -1, which lowers the accuracy to 64.33%. The steady drop shows that both the larger weight on later rounds and the different reward values within a round contribute to the performance. Giving later rounds a larger weight helps the model focus on actions that are closer to the final result, and the different reward values within a round help the model tell apart actions that fix a wrong answer, break a correct answer or bring no change. We also compare our reward with other ways to control sparsity. Edge-Reg adds an explicit regularization on edge logits to encourage sparsity. It reaches 63.33% with a cost of 90.56%, which is lower than CPP in accuracy. In contrast, our reward controls sparsity directly through k without any extra regularization term. For example, Second_0 reaches a higher accuracy of 64.00% with a lower cost of 89.00% than Edge-Reg. This is an advantage of our reward design, since sparsity and accuracy are balanced inside one reward instead of being handled by two separate terms. The reward of MasHost, which is also designed for debate, only reaches 61.67%, which is 3.5 points lower than ours. This shows that a reward tailored for multi-round debate is necessary. Finally, we study two-stage training. The Second rows first train with $k = 1$ and then change k to 0.5, 0 or -1 in the second stage. Compared with setting the same k from the start, two-stage training keeps much more accuracy. With $k = 0.5$, the accuracy rises from 62.33% to 62.83%. With $k = 0$, it rises from 59.50% to 64.00%, and with $k = -1$, it rises from 60.00% to 63.00%. At the same time, the number of edges stays close to that of CPP, while the cost drops to around 89% to 91%. This suggests that the first stage learns which edges are useful, and the second stage then removes costly edges that bring little benefit. Overall, these results show that our reward design is effective, and that the coefficient k offers a simple and flexible way to trade accuracy for lower cost.

Table 6: Ablation study on Pipeline and Reward design.

	Pipeline					**Reward**			
Ablation	Ref	Acc (%)	Cost	Edge	**Ablation**	Ref	Acc (%)	Cost	Edge
10-Sample	0.91, 0.86, 0.86	**66.00 ± 2.37**	96.73%	15.77	R_ + a	0.91, 0.87, 0.86	**65.17 ± 3.37**	93.79%	15.83
30-Sample	0.95, 0.92, 0.90	64.67 ± 3.08	97.34%	16.61	R_0.5a	0.73, 0.7, 0.69	62.33 ± 2.25	86.91%	12.70
90-Sample	0.92, 0.88, 0.87	64.00 ± 1.10	96.27%	16.08	R_0	0.53, 0.5, 0.5	59.50 ± 2.66	81.20%	9.17
150-Sample	0.87, 0.85, 0.84	62.33 ± 3.50	94.24%	15.34	R_ - a	0.39, 0.39, 0.41	60.00 ± 2.10	77.14%	7.18
w/o Training	0.95, 0.92, 0.91	63.33 ± 2.94	96.66%	16.73	w/o Round-Var.	0.83, 0.8, 0.79	64.67 ± 1.86	92.94%	14.50
Random	0.92, 0.87, 0.87	63.83 ± 2.79	95.20%	15.97	Vanilla ±1	0.91, 0.87, 0.86	64.33 ± 2.80	92.08%	15.84
Text-Profile	0.76, 0.74, 0.73	61.67 ± 2.66	87.60%	13.36	Edge-Reg	0.8, 0.76, 0.76	63.33 ± 3.88	90.56%	13.92
Symbolic-MoE	0.86, 0.84, 0.81	60.00 ± 3.46	94.16%	15.07	MasHost	0.85, 0.8, 0.82	61.67 ± 3.56	89.76%	14.85
Query-Linear	0.91, 0.87, 0.86	63.67 ± 1.75	92.23%	15.85	Second_0.5a	0.89, 0.85, 0.85	62.83 ± 2.40	91.31%	15.57
BE-Dominated	0.99, 0.99, 0.98	64.17 ± 2.79	99.89%	17.78	Second_0	0.9, 0.86, 0.86	64.00 ± 3.69	89.00%	15.77
w/o BE	0.28, 0.31, 0.34	57.83 ± 4.26	80.87%	5.53	Second_- a	0.87, 0.83, 0.82	63.00 ± 2.61	91.47%	15.12

Table 7: **Accuracy (%, mean ± std)** on the left and **Cost** on the right. For the first 3 lines, the columns denote the token limit configurations.

Method	q5/ q5/ q5		q75/ q75/ q75		q80/ q90/ q90		q85/ q95/ q95	
Full-Debate	52.83 ± 3.31	68.52%	58.83 ± 2.04	85.87%	60.00 ± 2.00	88.51%	60.67 ± 2.94	90.30%
ARG-Designer	52.67 ± 2.73	68.24%	61.83 ± 4.12	85.26%	59.00 ± 2.37	87.58%	60.33 ± 3.14	89.92%
Adv_CPP	**53.83 ± 2.04**	**66.26%**	57.00 ± 3.52	80.77%	**62.17 ± 3.54**	**83.27%**	**63.00 ± 3.16**	**84.48%**
IoE-Vote*				52.83 ± 3.97	78.82%			
Debate-IoE				61.00 ± 2.53	87.39%			
Consistency-Fuse-12				62.00 ± 1.67	68.46%			
Consistency-Fuse-13				**62.00 ± 1.67**	**81.70%**			
Consistency-Fuse-14				62.50 ± 0.84	94.94%			

Efficiency. Furthermore, we find that pruning alone does not reduce the cost very much. Since output tokens are more expensive than input tokens, we further reduce the cost by limiting the number of output tokens. We first collect the statistics of output tokens of each LLM on the validation set. We then truncate the outputs only in the first and second rounds, which means that the agents

only say part of what they would normally say. In the last round, the agents give complete outputs to obtain the final answer. Each configuration in Table 7 sets the token limits to the corresponding quantiles of the output token statistics. The bold results in the table are Pareto-optimal. Adv-CPP produces several Pareto-optimal results, and it is the only debate method that reaches the Pareto frontier. Under every token limit configuration, each result of Full-Debate and ARG-Designer is dominated by a result of Adv-CPP, which achieves a higher accuracy with a lower cost. For example, under q85/q95/q95, Adv-CPP reaches 63.00% with a cost of 84.48%, while Full-Debate and ARG-Designer only reach 60.67% and 60.33% with costs above 89%. Adv-CPP also compares well with consistency methods. With the q85/q95/q95 configuration, it outperforms Consistency-Fuse-14 by 0.5 points while reducing the cost by more than 10 points. Under the loosest limit, Adv-CPP still keeps the lowest cost of 66.26% among all debate methods. These results show that CPP can be combined with output truncation to further reduce the cost, and that its pruning decisions remain effective when the agents only give partial responses in the early rounds. All detailed output tokens statistics can be found in Appendix C.

4.3.3 Pipeline Ablation

Pipeline ablation. The left part of Table 6 studies the design of the pipeline. We first vary the number of validation samples used to initialize the behavior embedding, while our main experiments use 60 samples. Accuracy does not increase with more samples. Using only 10 samples already reaches 66.00%, while using 30, 90 and 150 samples gives 64.67%, 64.00% and 62.33%, respectively. This shows that a small number of responses is enough to capture the behavior of each agent, so our initialization is cheap to obtain. A possible reason for the drop is that too many responses make the embedding more general and weaken the specific behavior of each agent. We then study the effect of training and initialization. Without training, CPP only reaches 63.33%, which is close to Random with 63.83%. This shows that the gain of CPP mainly comes from reinforcement learning rather than from the initialization alone. When the behavior embedding is initialized with text-introduction profiles, the accuracy drops to 61.67% and the model keeps fewer edges. The initialization used in Symbolic-MoE, which adds statistical information on top of profiles, performs even worse with 60.00%. Both are clearly lower than our response-based initialization, which confirms that encoding the actual responses of agents provides richer information about their behavior than a general description. Next, we study the structure of FuseGLU. Query-Linear removes the fusion inside the GLU, which lowers the nonlinearity applied to the query. It keeps almost the same number of edges as CPP, with 15.85 compared to 15.83, but its accuracy drops to 63.67%. This shows that the gain of FuseGLU comes from better decisions on which edges to keep, not from keeping more edges. BE-Dominated swaps the positions of the behavior embedding and the query, so that the behavior embedding leads the computation. In this case, the model keeps almost all edges, with reference rates close to 0.99 and a cost of 99.89%, while its accuracy drops to 64.17%. The model tends to decide based on agent identity and loses its ability to prune according to the content of the debate. Removing the behavior embedding causes the largest drop among all variants. The model keeps only 5.53 edges and its accuracy falls to 57.83%. Without knowing which agent is making the decision, the model cannot judge when communication is useful and therefore prunes too much. Taken together, these results show that the behavior embedding is necessary, and that it works best as extra information fused into the gate and up projections rather than as the main input. They also show that the full design of CPP, including response-based initialization, reinforcement learning and the fusion structure of FuseGLU, is needed to reach the best balance between accuracy and cost.

5 Conclusion

We propose the Conditional Progressive Pruning framework. It is the first to prune MAD by efficiently using intermediate information between rounds and modeling the conditions across actions. This fully releases the potential of communication-based collaboration. CPP is also the first SoTA MAD framework that fully outperforms Single Agent and Heterogeneous Consistency methods under the same computational cost. Overall, we are the first to fully verify the advantage of MAD, which ends the debate on whether MAD is valuable. We also analyze the promising room for optimizing MAD and lay an empirical foundation for future multi-agent work.

AI USE STATEMENT

When writing the code, we did not use AI to generate any new code. However, we did use AI to check our handwritten code and asked it for optimization suggestions. The final version of the code contains no part directly generated by AI. When writing the paper, most of our tables and figures were generated with the help of AI based on experimental results obtained by humans, i.e., we used AI to generate tables and figures. We did not use AI to generate any raw data or to explore scientific ideas. We also used AI to polish the English of our draft and to give writing suggestions. Finally, we did not use AI at all to make mathematical claims, provide key elements for proving mathematical claims, help write proofs, propose or refine hypotheses, design research methods or experiments or give feedback on them, or implement methods.

REFERENCES

Ziwei Chai, Guoyin Wang, Jing Su, Tianjie Zhang, Xuanwen Huang, Xuwu Wang, Jingjing Xu, Jianbo Yuan, Hongxia Yang, Fei Wu, et al. An expert is worth one token: Synergizing multiple expert llms as generalist via expert token routing. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11385–11396, 2024.

Justin Chen, Sukwon Yun, Elias Stengel-Eskin, Tianlong Chen, and Mohit Bansal. Symbolic mixture-of-experts: Adaptive skill-based routing for heterogeneous reasoning. 2025.

Hyeong Kyu Choi, Jerry Zhu, and Sharon Li. Debate or vote: Which yields better decisions in multi-agent large language models? *Advances in Neural Information Processing Systems*, 38: 101732–101764, 2026.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. *arXiv preprint arXiv:2305.14325*, 2023.

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.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *International conference on learning representations*, volume 2024, pp. 32808–32824, 2024.

Yubin Kim, Chanwoo Park, Hyewon Jeong, Yik S Chan, Xuhai Xu, Daniel McDuff, Hyeonhoon Lee, Marzyeh Ghassemi, Cynthia Breazeal, and Hae W Park. Mdagents: An adaptive collaboration of llms for medical decision-making. *Advances in Neural Information Processing Systems*, 37:79410–79452, 2024.

Junyou Li, Qin Zhang, Yangbin Yu, Qiang Fu, and Deheng Ye. More agents is all you need. *arXiv preprint arXiv:2402.05120*, 2024.

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.

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.

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

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

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.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Song Wang, Zhen Tan, Zihan Chen, Shuang Zhou, Tianlong Chen, and Jundong Li. Anymac: Cascading flexible multi-agent collaboration via next-agent prediction. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 11566–11578, 2025a.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290, 2024.

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, 2025b.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Kuo Yang, Xingjie Yang, Linhui Yu, Qing Xu, Yan Fang, Xu Wang, Zhengyang Zhou, and Yang Wang. Mashost builds it all: autonomous multi-agent system directed by reinforcement learning. *arXiv preprint arXiv:2506.08507*, 2025b.

Rui Ye, Shuo Tang, Rui Ge, Yaxin Du, Zhenfei Yin, Siheng Chen, and Jing Shao. Mas-gpt: Training llms to build llm-based multi-agent systems. *arXiv preprint arXiv:2503.03686*, 2025.

Yuting Zeng, Weizhe Huang, Lei Jiang, Tongxuan Liu, Xitai Jin, Chen Tianying Tiana, Jing Li, and Xiaohua Xu. S2-mad: Breaking the token barrier to enhance multi-agent debate efficiency. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 9393–9408, 2025.

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. *arXiv preprint arXiv:2410.11782*, 2024.

Guibin Zhang, Yanwei Yue, Zhixun Li, Sukwon Yun, Guancheng Wan, Kun Wang, Dawei Cheng, Jeffrey Yu, and Tianlong Chen. Cut the crap: An economical communication pipeline for llm-based multi-agent systems. In *International Conference on Learning Representations*, volume 2025, pp. 75389–75428, 2025.

Jintian Zhang, Xin Xu, Ningyu Zhang, Ruibo Liu, Bryan Hooi, and Shumin Deng. Exploring collaboration mechanisms for llm agents: A social psychology view. *arXiv preprint arXiv:2310.02124*, 2023.

Chuanyang Zheng, Zhengying Liu, Enze Xie, Zhenguo Li, and Yu Li. Progressive-hint prompting improves reasoning in large language models. *arXiv preprint arXiv:2304.09797*, 2023.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. Language agents as optimizable graphs. *arXiv preprint arXiv:2402.16823*, 2024.

A Action-Wise Influence Dependency

Table 8: Dependency structure of A_1 and A_2 under 3×3 MAD Architecture.

Action	**Dependency**			
$A_1[0][0]$	$(A_1[0][0], 1)$	$(A_1[0][1], \gamma)$	$(A_1[0][2], \gamma^2)$	$(A_2[0][0], \beta)$
	$(A_2[0][1], \beta\gamma)$	$(A_2[0][2], \beta\gamma^2)$	$(A_2[1][0], \beta\gamma)$	$(A_2[1][2], \beta\gamma^2)$
	$(A_2[2][0], \beta\gamma)$	$(A_2[2][1], \beta\gamma^2)$		
$A_1[0][1]$	$(A_1[0][1], 1)$	$(A_1[0][2], \gamma)$	$(A_2[0][0], \beta)$	$(A_2[0][1], \beta\gamma)$
	$(A_2[0][2], \beta\gamma^2)$	$(A_2[1][0], \beta\gamma)$	$(A_2[1][2], \beta\gamma^2)$	$(A_2[2][0], \beta\gamma)$
	$(A_2[2][1], \beta\gamma^2)$			
$A_1[0][2]$	$(A_1[0][2], 1)$	$(A_2[0][0], \beta)$	$(A_2[0][1], \beta\gamma)$	$(A_2[0][2], \beta\gamma^2)$
	$(A_2[1][0], \beta\gamma)$	$(A_2[1][2], \beta\gamma^2)$	$(A_2[2][0], \beta\gamma)$	$(A_2[2][1], \beta\gamma^2)$
$A_1[1][0]$	$(A_1[1][0], 1)$	$(A_1[1][2], \gamma)$	$(A_2[0][1], \beta\gamma)$	$(A_2[0][2], \beta\gamma^2)$
	$(A_2[1][1], \beta)$	$(A_2[1][0], \beta\gamma)$	$(A_2[1][2], \beta\gamma^2)$	$(A_2[2][1], \beta\gamma^2)$
$A_1[1][1]$	$(A_1[1][1], 1)$	$(A_1[1][0], \gamma)$	$(A_1[1][2], \gamma^2)$	$(A_2[0][1], \beta\gamma)$
	$(A_2[0][2], \beta\gamma^2)$	$(A_2[1][1], \beta)$	$(A_2[1][0], \beta\gamma)$	$(A_2[1][2], \beta\gamma^2)$
	$(A_2[2][1], \beta\gamma^2)$			
$A_1[1][2]$	$(A_1[1][2], 1)$	$(A_2[0][1], \beta\gamma)$	$(A_2[0][2], \beta\gamma^2)$	$(A_2[1][1], \beta)$
	$(A_2[1][0], \beta\gamma)$	$(A_2[1][2], \beta\gamma^2)$	$(A_2[2][1], \beta\gamma^2)$	
$A_1[2][0]$	$(A_1[2][0], 1)$	$(A_1[2][1], \gamma)$	$(A_2[0][2], \beta\gamma^2)$	$(A_2[1][2], \beta\gamma^2)$
	$(A_2[2][2], \beta)$	$(A_2[2][0], \beta\gamma)$	$(A_2[2][1], \beta\gamma^2)$	
$A_1[2][1]$	$(A_1[2][1], 1)$	$(A_2[0][2], \beta\gamma^2)$	$(A_2[1][2], \beta\gamma^2)$	$(A_2[2][2], \beta)$
	$(A_2[2][0], \beta\gamma)$	$(A_2[2][1], \beta\gamma^2)$		
$A_1[2][2]$	$(A_1[2][2], 1)$	$(A_1[2][0], \gamma)$	$(A_1[2][1], \gamma^2)$	$(A_2[0][2], \beta\gamma^2)$
	$(A_2[1][2], \beta\gamma^2)$	$(A_2[2][2], \beta)$	$(A_2[2][0], \beta\gamma)$	$(A_2[2][1], \beta\gamma^2)$
$A_2[0][0]$	$(A_2[0][0], 1)$	$(A_2[0][1], \gamma)$	$(A_2[0][2], \gamma^2)$	
$A_2[0][1]$	$(A_2[0][1], 1)$	$(A_2[0][2], \gamma)$		
$A_2[0][2]$	$(A_2[0][2], 1)$			
$A_2[1][0]$	$(A_2[1][0], 1)$	$(A_2[1][2], \gamma)$		
$A_2[1][1]$	$(A_2[1][1], 1)$	$(A_2[1][0], \gamma)$	$(A_2[1][2], \gamma^2)$	
$A_2[1][2]$	$(A_2[1][2], 1)$			
$A_2[2][0]$	$(A_2[2][0], 1)$	$(A_2[2][1], \gamma)$		
$A_2[2][1]$	$(A_2[2][1], 1)$			
$A_2[2][2]$	$(A_2[2][2], 1)$	$(A_2[2][0], \gamma)$	$(A_2[2][1], \gamma^2)$	

B Edge Statistics of Table 2

C Output Token Statistics on MATH-Hard

D Detailed Consistency Results

Table 9: Comparison of **Referenced Ratio** (Grok, GPT, Mistral) and **Edge Number** across MATH-Hard, GPQA-Diamond and MMLU-Pro.

Method	MATH-Hard		GPQA-Diamond		MMLU-Pro	
	Ref	Edge	Ref	Edge	Ref	Edge
Full-Debate	1, 1, 1	18.00	1, 1, 1	18.00	1, 1, 1	18.00
Random	0.5, 0.51, 0.5	9.02	0.5, 0.5, 0.5	9.02	0.5, 0.5, 0.5	9.02
LLM-Self	1.67, 1.58, 1.62	29.24	1.76, 1.47, 1.51	28.48	1.77, 1.58, 1.61	29.80
Intervention	0.4, 0.77, 0.83	12.00	0.53, 0.75, 0.69	12.00	0.56, 0.72, 0.72	12.00
S^2-MAD	0.57, 0.58, 0.57	10.31	0.52, 0.56, 0.55	9.77	0.45, 0.47, 0.46	8.24
IoE-Debate	0.67, 0.67, 0.67	12.00	0.67, 0.67, 0.67	12.00	0.67, 0.67, 0.67	12.00
Debate-IoE	0.67, 0.67, 0.67	12.00	0.67, 0.67, 0.67	12.00	0.67, 0.67, 0.67	12.00
Zero	0, 0, 0	0.00	0, 0, 0	0.00	0, 0, 0	0.00
ARG-Designer	0.97, 0.95, 0.96	17.32	0.68, 0.66, 0.69	12.15	0.28, 0.22, 0.23	4.43
AnyMac	0.8, 0.8, 0.79	14.28	0.7, 0.67, 0.68	12.22	0.6, 0.58, 0.57	10.55
CPP	0.85, 0.84, 0.84	15.24	0.82, 0.78, 0.8	14.31	0.98, 0.97, 0.97	17.52

Table 10: Output Token limit configurations at different quantiles on MATH-Hard Validation

Threshold	Grok	GPT	Mistral
q5	65	132	139
q75	157	318	432
q80	166	353	478
q85	177	393	527
q90	196	458	623
q95	233	621	872
q100	6000	6000	6000

Table 11: Consistency Accuracy (%) of Grok across different sample sizes n on MATH-Hard-I.

n	2	3	4	5	6	7	8	9	**Cost**
Grok	49.33 ±2.34	51.17 ±2.40	53.33 ±3.83	55.00 ±3.10	56.00 ±0.63	56.83 ±1.94	57.83 ±1.47	58.50 ±1.97	101.06%

Table 12: Consistency Accuracy (%) of GPT across different sample sizes n on MATH-Hard-I.

n	2	3	4	5	6	7	8	9	10	11	12	13	**Cost**
GPT	47.67 ±2.25	49.33 ±2.42	51.67 ±1.51	53.33 ±2.07	53.50 ±2.43	54.00 ±1.79	54.67 ±1.03	55.83 ±0.98	57.33 ±2.34	57.67 ±1.03	57.00 ±1.55	57.17 ±2.04	100.32%

Table 13: Consistency Accuracy (%) of Mistral across different sample sizes n on MATH-Hard-I.

n	2	3	4	5	6	7	8	9	10	11	12	13	**Cost**
Mistral	49.00 ±4.24	52.67 ±2.16	54.50 ±3.78	56.00 ±2.83	55.83 ±2.32	56.67 ±2.25	56.33 ±2.42	56.67 ±1.97	57.17 ±1.83	56.83 ±1.72	57.50 ±1.97	57.17 ±1.94	107.48%

Table 14: Consistency Accuracy (%) of Fuse-Extension across different sample sizes n on MATH-Hard-I.

n	2	3	4	5	6	7	8	9	10	11	12	**Cost**
Fuse-Extension	49.83 ±1.83	52.67 ±1.21	53.83 ±2.64	55.50 ±2.43	58.67 ±1.03	59.83 ±1.17	60.50 ±2.59	62.33 ±1.63	62.83 ±2.14	63.50 ±3.27	64.00 ±2.45	107.89%

Table 15: Consistency Accuracy (%) of Best-Extension across different sample sizes n on MATH-Hard-I.

n	2	3	4	5	6	7	8	9	10	11	**Cost**
Best-Extension	50.83 ±2.32	53.17 ±1.60	55.00 ±2.10	57.33 ±2.66	58.17 ±2.71	59.17 ±2.64	61.33 ±1.51	62.00 ±2.10	62.50 ±1.52	62.83 ±2.14	103.48%

Table 16: Consistency Accuracy (%) of Mistral across different sample sizes n on MATH-Hard-II.

n	2	3	4	5	6	7	8	9	10	11	12
Mistral	48.50 ±4.85	51.83 ±3.37	54.33 ±3.14	56.33 ±2.80	55.50 ±1.97	56.50 ±2.07	56.67 ±2.42	57.00 ±2.00	57.00 ±2.45	56.83 ±1.17	58.00 ±1.79

n	13	14	15	16	17	18	19	20	21	22	**Cost**
Mistral	57.00 ±2.10	56.67 ±2.25	58.17 ±3.49	58.83 ±3.66	59.00 ±3.10	59.00 ±3.22	58.17 ±2.32	58.67 ±2.42	59.50 ±2.51	59.67 ±2.25	94.62%

Table 17: Consistency Accuracy (%) of Fuse-Extension across different sample sizes n on MATH-Hard-II.

n	2	3	4	5	6	7	8	9	10	11	12	13	14	**Cost**
Fuse-Extension	48.00 ±3.35	51.17 ±2.14	53.33 ±2.73	53.33 ±2.58	56.50 ±2.26	58.33 ±2.34	59.17 ±0.98	60.00 ±1.10	61.83 ±1.47	62.17 ±1.17	62.00 ±1.67	62.00 ±1.67	62.50 ±0.84	94.94%

Table 18: Consistency Accuracy (%) of Best-Extension across different sample sizes n on MATH-Hard-II.

n	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	**Cost**
Best-Extension	48.00 ±3.90	51.00 ±2.37	53.33 ±3.98	53.17 ±2.99	55.83 ±2.14	56.50 ±2.66	58.17 ±2.79	57.33 ±3.44	58.50 ±1.97	60.00 ±1.10	60.50 ±1.64	60.33 ±1.37	60.83 ±1.60	61.50 ±1.52	62.17 ±1.72	94.81%

Table 19: Adversarial Consistency Accuracy (%) of Mistral across different sample sizes n on MATH-Hard-II.

n	2	3	4	5	6	7	8	9	10	11	12
Mistral	11.67 ±4.03	10.50 ±1.52	13.50 ±2.74	15.17 ±1.60	15.67 ±3.14	15.50 ±1.87	17.00 ±2.28	16.67 ±2.73	17.83 ±1.83	18.00 ±2.76	19.17 ±1.17

n	13	14	15	16	17	18	19	20	21	22	**Cost**
Mistral	19.67 ±1.21	20.83 ±1.47	20.50 ±1.52	20.50 ±3.08	19.50 ±2.07	19.50 ±2.66	19.17 ±2.04	19.00 ±1.41	20.50 ±2.35	19.83 ±1.83	25.79%

Table 20: Adversarial Consistency Accuracy (%) of Fuse-Extension across different sample sizes n on MATH-Hard-II.

n	2	3	4	5	6	7	8	9	10	11	12	13	14	**Cost**
Fuse-Extension	12.67 ±2.42	12.50 ±2.59	12.17 ±2.79	13.83 ±2.64	14.00 ±2.68	14.83 ±2.64	14.33 ±2.34	14.17 ±2.48	14.00 ±1.41	14.67 ±2.50	16.17 ±1.72	16.33 ±2.80	18.00 ±1.26	54.67%

Table 21: Adversarial Consistency Accuracy (%) of Best-Extension across different sample sizes n on MATH-Hard-II.

n	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	**Cost**
Best-Extension	12.83 ±2.56	12.83 ±1.47	13.17 ±1.94	15.50 ±3.62	14.17 ±3.54	15.00 ±2.45	15.83 ±2.32	16.17 ±2.14	16.50 ±1.87	16.50 ±1.22	16.83 ±0.98	17.33 ±1.75	16.83 ±0.75	18.17 ±1.60	17.83 ±1.94	47.54%

Table 22: Consistency Accuracy (%) of Grok across different sample sizes n on GPQA-Diamond.

n	2	3	4	5	6	7	8	9	**Cost**
Grok	58.00 ±2.28	59.83 ±2.86	60.33 ±2.25	60.83 ±2.32	62.50 ±2.59	63.33 ±1.63	63.17 ±1.94	63.00 ±1.67	117.55%

Table 23: Consistency Accuracy (%) of GPT across different sample sizes n on GPQA-Diamond.

n	2	3	4	5	6	7	8	9	10	11	12	13	14	15	**Cost**
GPT	39.33 ±3.14	42.83 ±1.72	44.00 ±1.41	45.83 ±3.19	47.33 ±1.75	46.33 ±2.73	46.83 ±3.60	46.33 ±2.73	47.83 ±1.60	47.00 ±2.10	46.00 ±2.37	46.67 ±2.42	46.83 ±2.64	46.33 ±2.42	102.71%

Table 24: Consistency Accuracy (%) of Mistral across different sample sizes n on GPQA-Diamond.

n	2	3	4	5	6	7	8	9	10	11	12
Mistral	42.67 ±2.94	42.33 ±2.58	43.67 ±1.97	43.33 ±2.34	43.50 ±0.84	44.17 ±1.33	44.33 ±1.03	43.83 ±0.75	43.83 ±0.41	43.50 ±0.55	44.33 ±1.03

n	13	14	15	16	17	18	19	20	21	22	**Cost**
Mistral	44.17 ±0.98	43.67 ±0.82	44.50 ±1.38	44.17 ±0.75	44.33 ±0.52	44.83 ±1.17	44.67 ±1.37	44.83 ±1.47	44.67 ±2.25	44.50 ±1.87	100.49%

Table 25: Consistency Accuracy (%) of Fuse-Extension across different sample sizes n on GPQA-Diamond.

n	2	3	4	5	6	7	8	9	10	11	12	**Cost**
Fuse-Extension	58.50 ±2.26	61.33 ±3.56	61.17 ±3.19	62.67 ±2.16	60.67 ±2.58	58.00 ±2.90	57.83 ±3.25	57.33 ±4.84	57.33 ±3.27	58.83 ±1.94	59.83 ±3.06	96.24%

Table 26: Consistency Accuracy (%) of Best-Extension across different sample sizes n on GPQA-Diamond.

n	2	3	4	5	6	7	8	9	10	11	**Cost**
Best-Extension	59.17 ±4.40	61.33 ±4.50	60.67 ±3.39	61.83 ±2.64	62.83 ±2.04	63.17 ±2.48	61.50 ±2.81	62.17 ±1.94	62.17 ±2.32	62.83 ±1.72	99.14%

Table 27: Consistency Accuracy (%) of Grok across different sample sizes n on MMLU-Pro.

n	2	3	4	5	6	7	8	9	**Cost**
Grok	57.83 ±1.47	59.00 ±1.41	60.00 ±0.63	60.17 ±1.47	61.00 ±1.55	61.00 ±1.41	60.50 ±1.52	60.33 ±1.37	115.57%

Table 28: Consistency Accuracy (%) of GPT across different sample sizes n on MMLU-Pro.

n	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	**Cost**
GPT	45.50 ±3.02	45.83 ±2.71	47.33 ±1.21	47.00 ±0.00	47.50 ±1.76	47.50 ±0.84	47.33 ±0.52	47.83 ±1.60	47.83 ±1.33	48.50 ±1.05	47.67 ±0.82	47.33 ±1.03	47.17 ±1.33	47.67 ±1.63	47.33 ±1.63	103.85%

Table 29: Consistency Accuracy (%) of Mistral across different sample sizes n on MMLU-Pro.

n	2	3	4	5	6	7	8	9	10
Mistral	57.50 ±2.17	58.83 ±1.72	58.00 ±1.79	58.33 ±2.25	58.67 ±3.39	58.33 ±2.25	58.33 ±2.50	58.17 ±2.64	58.67 ±2.34

n	11	12	13	14	15	16	17	18	19	**Cost**
Mistral	58.83 ±2.04	58.83 ±2.04	58.50 ±1.38	57.83 ±1.94	58.33 ±1.21	57.83 ±1.47	57.50 ±1.05	57.83 ±1.72	58.00 ±1.10	103.16%

Table 30: Consistency Accuracy (%) of Fuse-Extension across different sample sizes n on MMLU-Pro.

n	2	3	4	5	6	7	8	9	10	11	12	**Cost**
Fuse-Extension	57.67 ±2.58	59.00 ±1.55	59.83 ±1.33	60.83 ±1.17	62.33 ±2.25	64.33 ±2.07	64.33 ±2.25	66.17 ±2.32	65.83 ±2.23	64.50 ±1.76	64.50 ±2.07	99.23%

Table 31: Consistency Accuracy (%) of Best-Extension across different sample sizes n on MMLU-Pro.

n	2	3	4	5	6	7	8	9	10	11	**Cost**
Best-Extension	57.67 ±2.07	59.17 ±1.83	57.83 ±2.04	59.00 ±2.45	60.50 ±2.35	62.50 ±3.08	63.33 ±3.20	64.17 ±3.06	63.67 ±2.50	63.17 ±3.31	99.88%