

OrchDebate: Adaptive Orchestration for Multi-Agent Debate via Self-Evolving Skills

Kaiyu Zhou^{1,2}, Yongsen Zheng^{1,2*}, Xuanye Zhang^{1,2}, Kwok-Yan Lam^{1,2}

¹College of Computing and Data Science, Nanyang Technological University, Singapore

²Digital Trust Centre, Nanyang Technological University, Singapore

{kaiyu001, xuanye001}@e.ntu.edu.sg, {yongsen.zheng, kwokyan.lam}@ntu.edu.sg

Abstract

Multi-Agent Debate (MAD) has emerged as a promising paradigm for eliciting stronger inference-time reasoning, yet most existing systems still use fixed interaction protocols across instances. Such static designs can suffer from anchoring bias, overthinking on easy problems, and rigid coordination across tasks of varying difficulty. We present **OrchDebate**, a framework that recasts multi-agent debate as structured-state orchestration over reusable debate-control skills. Instead of following a pre-defined template, OrchDebate uses an orchestrator to examine the current debate state and select the next coordination action, such as introducing agents under different visibility settings or terminating deliberation with an adaptive readout rule. This skill-based view makes debate control modular and state-dependent. The orchestrator operates through a structured control interface rather than directly generating reasoning content, helping reduce reliance on early answers. OrchDebate further maintains a self-evolving skill bank whose reusable realizations are refined through outcome-guided updates on an evolution stream, without additional external training data. Across diverse benchmarks, OrchDebate delivers a better accuracy–cost trade-off than representative fixed and adaptive debate methods by allocating debate compute more selectively.

1 Introduction

Multi-Agent Debate (MAD), where multiple large language model (LLM) instances deliberate over a shared query before producing a final answer, has emerged as a promising paradigm for achieving stronger collective intelligence in inference-time reasoning (Xi et al., 2025; Luo et al., 2025; Guo et al., 2024; Tran et al., 2025; Yin et al., 2023; Chen et al., 2024; Zeng et al., 2025; Sun et al., 2025; Becker et al., 2025; Du et al., 2024; Xiong

et al., 2023; Xu et al., 2023; Li et al., 2024a). Yet a central limitation remains: most MAD systems still operate under a pre-specified and largely fixed interaction protocol, with fixed agent counts, interaction patterns, debate rounds, and decision rules (Guo et al., 2024; Tran et al., 2025; Tillmann, 2025; Cui et al., 2025; Zhang et al., 2026c).

However, this static design is increasingly mismatched to the heterogeneous nature of reasoning tasks. Recent evidence points to three recurring failure modes under fixed debate protocols: i) anchoring bias, where agents gravitate toward an initial answer instead of correcting it (Weng et al., 2025; Wynn et al., 2025; Cui et al., 2025; Wu et al., 2025b); ii) overthinking collapse, where unnecessary deliberation flips correct answers and wastes tokens on easy instances (Wu et al., 2025a; Zhang et al., 2025; Yang et al., 2025); and iii) protocol rigidity, where a one-size-fits-all interaction strategy fails to match the varying difficulty and structure of different problems (Eo et al., 2025; Kaesberg et al., 2025; Choi et al., 2025; Zhang et al., 2026b,c; Li et al., 2024a). These observations suggest that the core challenge in MAD is no longer simply how to make agents debate, but how to orchestrate debate adaptively.

Recent work has taken important steps toward more adaptive debate, including confidence-triggered invocation (Eo et al., 2025), dynamic role assignment (Zhang et al., 2026b; Fan et al., 2026; Zhang et al., 2026c), budget-aware structuring (Yang et al., 2026a), sparse or structured communication (Zeng et al., 2025; Sun et al., 2025; Li et al., 2024b; Liu et al., 2026), diversity-oriented deliberation (Liu et al., 2025; Li et al., 2026; Cui et al., 2025; Wu et al., 2025b), and dialogue-conditioned composition of discussion actions (Gu et al., 2026). Together, these methods show that useful debate behavior can be selected from the current context. Their adaptation mechanisms, however, are exposed through method-specific choices such as

*Corresponding author.

whether to debate, which participants or paths to use, or which discussion action to execute. In parallel, work on evolving skills in LLM agent systems has shown that agent capabilities can be externalized, reused, and refined over time (Yang et al., 2026b; Pei et al., 2025; Qiu et al., 2026; Zhang et al., 2026a). These directions motivate a unified debate-control layer that can coordinate the entire next step through one state and action interface, while improving reusable coordination strategies across episodes.

To address this gap, we introduce **OrchDebate**, a framework that reconceptualizes multi-agent debate as structured-state orchestration over a bank of reusable debate-control skills. Instead of committing to a fixed protocol before deliberation begins, OrchDebate treats debate as a sequential coordination problem. At each step, an orchestrator examines a compact control-facing Structured Debate State (SDS) and selects one legal debate-control skill. A spawn skill jointly specifies the next agent’s role, visibility policy, and focus scope, whereas a finalize skill specifies the readout rule. The entire next control step is therefore expressed as one structured decision rather than a collection of separate routing choices. Moreover, the orchestrator is functionally distinct from the reasoning agents and operates through a constrained control interface rather than directly generating task reasoning or acting as the answer judge. This design gives it a more independent view of debate flow and is intended to reduce reliance on early answers.

Across episodes, OrchDebate performs outcome-guided control-skill evolution on a separate evolution stream. Delayed episode-level outcomes and SDS diagnostics identify recurring control patterns, while a mutator makes local edits to reusable skill realizations. New candidates are evaluated only on later executions before promotion or retirement, and the resulting bank is frozen for held-out evaluation. The update target is therefore a reusable coordination skill rather than a task-answer prompt or an example-specific correction, and the process requires no additional external training data. In our main experiments, we compare OrchDebate against competitive prompting, fixed debate protocols, and adaptive debate methods (Liang et al., 2024; Zeng et al., 2025; Li et al., 2024b; Liu et al., 2025; Li et al., 2026; Liu et al., 2026; Eo et al., 2025; Fan et al., 2026) on six diverse benchmarks spanning scientific reasoning, logical reasoning, broad knowledge, mathematics, and multi-step reason-

ing (Rein et al., 2024; Liu et al., 2023; Hendrycks et al., 2021a; Cobbe et al., 2021; Hendrycks et al., 2021b; Sprague et al., 2024). OrchDebate achieves the best or tied-best accuracy on all six tasks under GPT-5-nano and on five of the six tasks under both Claude-Haiku-4.5 and Qwen-3-8B, while retaining a favorable inference-cost profile.

Taken together, our contributions are:

- We formulate multi-agent debate as structured-state orchestration over reusable debate-control skills, replacing fixed debate protocols with instance-adaptive, multi-axis coordination.
- We propose an adaptive orchestrator, functionally distinct from reasoning agents, that selects legal debate-control skills based on the Structured Debate State and includes mechanisms designed to limit reliance on early intermediate answers.
- We introduce an outcome-guided, self-evolving skill bank that refines reusable control-skill realizations across episodes through future-only probation, without requiring additional external training data.

2 Related Work

2.1 Multi-Agent Debate

MAD studies how multiple LLM agents can deliberate, critique, and aggregate reasoning to improve final-answer quality. Early work establishes the basic debate paradigm and its variants (Du et al., 2024; Liang et al., 2024; Xiong et al., 2023; Xu et al., 2023), while later studies extend it to cross-model communication and consensus-oriented interaction (Yin et al., 2023; Chen et al., 2024). More recent work improves debate efficiency or robustness through sparse communication, structured interaction, budget-aware design, and diversity-oriented deliberation (Zeng et al., 2025; Li et al., 2024b; Liu et al., 2026; Yang et al., 2026a; Liu et al., 2025; Cui et al., 2025). Adaptive methods further decide whether debate should be invoked (Eo et al., 2025; Fan et al., 2026), generate diverse reasoning paths and introduce verification under disagreement (Li et al., 2026), or compose the next discussion action from atomic acts conditioned on the dialogue context (Gu et al., 2026). At the same time, empirical analyses show that debate gains depend on when interaction is invoked, how agents are coordinated, and how final answers are selected, and identify anchoring, brittle self-correction, and overthinking as recurring failure modes (Choi et al., 2025; Kaesberg et al., 2025; Zhang et al., 2026c;

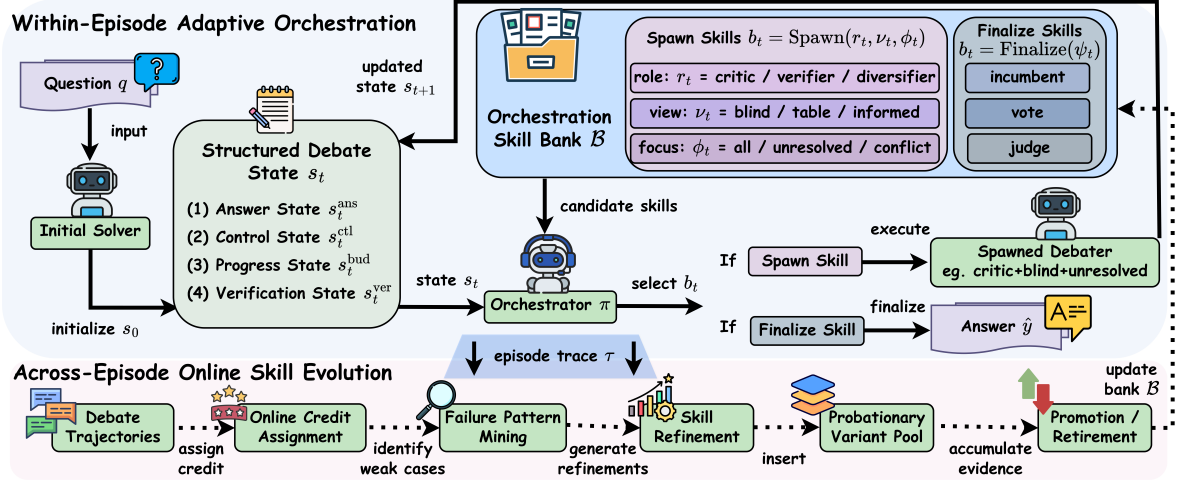


Figure 1: Overall framework of OrchDebate. Within each episode, the orchestrator reads SDS, selects the next debate-control skill from the skill bank, executes the resulting spawn or finalize action, and updates SDS. Across episodes, debate trajectories provide feedback for online skill evolution, which refines the skill bank over time.

Li et al., 2024a; Weng et al., 2025; Wynn et al., 2025; Wu et al., 2025a; Yang et al., 2025; Zhang et al., 2025; Wu et al., 2025b). OrchDebate follows this broader move toward instance-dependent interaction, but represents the entire next control step as one typed skill selected from the current legal set: spawn skills jointly specify role, visibility, and focus, while finalize skills specify readout. It additionally refines the reusable realizations of these skills across episodes.

2.2 Adaptive Coordination and Evolving Skills

Recent work on LLM agents shows that useful behaviors can be externalized as skills and improved over time through refinement, memory, and self-improvement mechanisms (Yang et al., 2026b; Qiu et al., 2026; Zhang et al., 2026a; Pei et al., 2025). This line of research suggests that agent capabilities need not be encoded solely in static prompts or monolithic policies, but can instead be represented, selected, and updated as modular skills. OrchDebate applies this view to coordination rather than task solving: the object being selected and refined is a reusable debate-control skill, not a task-answer prompt. Each candidate preserves the underlying control purpose and is evaluated through future-only probation before it can replace the incumbent. This connects state-conditioned skill selection within an episode with controlled realization-level refinement across episodes, complementing adaptive coordination mechanisms such as selective triggering, dynamic

role assignment, and context-aware communication (Eo et al., 2025; Zhang et al., 2026b; Fan et al., 2026; Sun et al., 2025; Zhou et al., 2025; Wang et al., 2026).

3 Methodology

OrchDebate reframes multi-agent debate as adaptive orchestration rather than a fixed protocol. The framework combines within-episode control over the next debate step with across-episode refinement of the reusable skill realizations that implement those decisions. Figure 1 provides an overview of the OrchDebate framework. We first introduce SDS in §3.1, then describe within-episode adaptive debate control in §3.2, next present across-episode online skill evolution in §3.3, and finally summarize the overall OrchDebate procedure in §3.4.

3.1 Structured Debate State

State abstraction. To decide what the orchestrator should observe, OrchDebate maintains a Structured Debate State (SDS), a compact control-facing summary that avoids conditioning on full multi-agent transcripts and retains only the signals needed for routing, verification, escalation, and stopping decisions.

Formally, at debate step t , we represent the state as

$$s_t = (s_t^{\text{ans}}, s_t^{\text{ver}}, s_t^{\text{ctl}}, s_t^{\text{bud}}), \quad (1)$$

where the four components summarize the answer landscape, verification progress, control-side risks, and resource budget, respectively. Concretely, s_t^{ans} records signals such as the current leading answer,

agreement level, answer margin, answer changes, and stability; s_t^{ver} tracks unresolved issues, verification coverage, and verification debt; s_t^{ctl} captures orchestration cues such as anchoring risk, response diversity, and stopping or escalation hints; and s_t^{bud} records the current debate step and remaining budget, with $\beta_t = B - C_t$ denoting the remaining token budget.

Leading answer and consensus dynamics. Let $\mathcal{A}_t = \{a_t^{(1)}, \dots, a_t^{(n_t)}\}$ denote the set of candidate answers currently produced by active reasoning agents. We define the leading answer by majority support:

$$a_t = \arg \max_a \sum_{i=1}^{n_t} \mathbf{1}[a_t^{(i)} = a], \quad (2)$$

and track its dynamics through

$$\delta_t = \mathbf{1}[a_t \neq a_{t-1}]. \quad (3)$$

Together with entropy, margin, and stability, these quantities tell the orchestrator how strong and how settled the current leader is.

3.2 Within-Episode Adaptive Debate Control

Debate-control skills. Instead of committing to a fixed debate template, OrchDebate operates over a reusable bank of debate-control skills, $\mathcal{B} = \{b_1, \dots, b_K\}$. A skill is a structured control primitive that specifies both what debate operation to perform and how it should be realized:

$$b = (\text{op}, r, \nu, \phi, \psi), \quad (4)$$

where $\text{op} \in \{\text{spawn}, \text{finalize}\}$ is the operation type, r is the role assigned to the next reasoning action (e.g., critic, verifier, diversifier), ν is the visibility policy (e.g., blind, table-only, informed), ϕ specifies the focus scope (e.g., all claims, unresolved points, conflict set, top- k issues), and ψ denotes the readout rule used at termination. For spawn skills, OrchDebate uses (r, ν, ϕ) to determine how the next reasoning step is constructed; for finalize skills, it uses ψ to determine how the final answer is read out from the current debate state.

Selecting the next control skill. Given the current SDS, the orchestrator adaptively determines how debate should proceed by selecting the next legal skill from the bank. Let $\mathcal{L}(s_t) \subseteq \mathcal{B}$ denote the set of legal skills under state s_t . The control decision is implemented as

$$\tilde{b}_t = \pi(s_t), \quad b_t = \text{Resolve}(\tilde{b}_t, \mathcal{L}(s_t)). \quad (5)$$

Here, π is an LLM-based controller conditioned on a structured textual rendering of SDS, $\tilde{b}_t$ is the raw controller proposal, and $\mathcal{L}(s_t) = \mathcal{L}(s_t)$ denotes the legal skill set under state s_t . The resolver maps the proposal to a legal skill or triggers a deterministic fallback when the proposal is malformed or illegal. The orchestrator does not decide the answer directly; it decides the next debate-control skill, which governs whether debate should continue, diversify, verify, or terminate.

Legal actions and fallback. Not every skill is legal at every step. In particular, finalization is allowed only after a minimum number of reasoning steps:

$$\mathcal{L}(s_t) = \mathcal{B}_{\text{spawn}} \cup \{b \in \mathcal{B}_{\text{final}} : t \geq t_{\min}\}. \quad (6)$$

Because the orchestrator is LLM-based, its output may occasionally be malformed or map to an illegal skill. OrchDebate therefore uses a deterministic fallback resolver π_{rule} whenever the proposed action cannot be matched to a valid skill in $\mathcal{L}(s_t)$.

Anchoring-aware control. An especially important case is blind spawning, which hides the current leading answer from the next agent and encourages an independent challenge rather than inheritance of the incumbent. This is particularly useful when the current leader is weak, unstable, or insufficiently verified, since the newly spawned agent can examine the problem without inheriting the incumbent answer.

3.3 Across-Episode Online Skill Evolution

Skill variants and bounded exploration. For every base skill b_k , OrchDebate maintains one validated incumbent realization v_k^* and a set of probationary candidates $\mathcal{P}_k$:

$$\mathcal{V}_k = \{v_k^*\} \cup \mathcal{P}_k. \quad (7)$$

The variants share the same semantic control function but may differ in wording, visibility phrasing, focus emphasis, or execution configuration. When b_k is selected, the incumbent is used by default and a probationary candidate is explored with bounded probability:

$$P(v \mid b_k) = \begin{cases} 1 - p, & v = v_k^*, \\ \frac{p}{|\mathcal{P}_k|}, & v \in \mathcal{P}_k. \end{cases} \quad (8)$$

This limits exposure to weak candidates while retaining a path for improving the initial bank. Evolution then proceeds through four stages.

1) Trajectory credit. After a completed evolution episode e , each executed variant receives outcome-guided credit

$$a_e(v) = (z_e - z_e^0) + \alpha \Delta_{\text{SDS}}(e) - \lambda c_e, \quad (9)$$

where z_e and z_e^0 are the terminal correctness of OrchDebate and direct solve, respectively, $\Delta_{\text{SDS}}(e)$ summarizes changes in verification coverage, unresolved issues, and vote stability, and c_e is normalized token cost. The signal therefore combines final outcome, control-state progress, and efficiency rather than rewarding additional deliberation by itself.

2) Diagnostic-driven refinement. Completed trajectories are aggregated into recurring control-level patterns, such as unresolved verification debt, unstable consensus, harmful answer changes, or excessive cost. A lightweight mutator receives the aggregated pattern and its source skill, then proposes a local, schema-preserving edit. The base operation, role identity, output schema, and control purpose remain fixed, so mutation changes how a reusable coordination skill is realized rather than constructing an answer prompt for a failed example.

3) Future-only probation. A new candidate enters $\mathcal{P}_k$ with $N(v) = 0$. The episodes that triggered its creation are not reused for validation; only subsequent episodes in which the candidate is actually executed update its exposure count, average advantage $\hat{A}(v)$, and degradation rate $\hat{D}(v)$. Probation is therefore based on future behavior under normal online usage rather than on the diagnostic examples that motivated the edit.

4) Promotion, retirement, and freezing. A candidate replaces the incumbent only after accumulating sufficient evidence, achieving positive average advantage, and remaining below the allowed degradation rate:

$$N(v) \geq n_{\min}, \quad \hat{A}(v) \geq \epsilon, \quad \hat{D}(v) \leq \tau_D. \quad (10)$$

Candidates that reach the evidence gate without satisfying the remaining conditions are retired. After the evolution stream, the resulting bank is frozen and held-out examples are used only for final evaluation. This yields two complementary timescales: the controller selects how debate proceeds within an episode, while outcome-guided evolution improves the reusable realizations that implement those decisions across episodes. Additional definitions and implementation details are provided in Appendix B.4.

3.4 Orchestrated Multi-Agent Debate

Initialization. Given a question q , OrchDebate first obtains an initial answer using the reasoning model $\mathcal{M}$ and initializes the SDS from the resulting debate state. This initial state gives the orchestrator a compact view of the current answer landscape, verification status, control-relevant cues, and remaining budget.

Iterative orchestration. At each step, the controller selects the next legal skill, executes the corresponding reasoning action, and updates the SDS accordingly. Formally, if v_t is the selected realization of skill b_t , the episode transition is

$$r_t = \text{Exec}(v_t, q, s_t; \mathcal{M}), \quad s_{t+1} = U(s_t, r_t) \quad (11)$$

where r_t denotes the resulting reasoning action or finalization outcome. The system repeats this loop until a legal finalize skill is selected, the step limit is reached, or the budget is exhausted. In this way, debate is not governed by a predetermined template; its trajectory is shaped online by a sequence of state-dependent control decisions.

Finalization. When a legal finalize skill is selected, OrchDebate commits to a final answer using the readout rule associated with that skill. If the step or budget limit is reached first, the system falls back to a readout from the current debate state. Algorithm 1 summarizes the full single-episode procedure.

Proposition 1 (Legality and finite-horizon termination). Assume that $\text{Resolve}(\tilde{b}_t, \mathcal{L}_t)$ returns a member of the current legal set and that the readout functions are defined for every reachable state. Then every executed control skill is legal, a selected finalize skill is executed only when $t \geq t_{\min}$, and Algorithm 1 returns after at most T control transitions or earlier when the token budget is reached. The result follows from legal resolution, the bounded loop, the explicit budget checks, and the fallback readout; Appendix B.3 gives the proof.

4 Experiments

4.1 Experimental Setup

Benchmarks. We evaluate on six benchmarks spanning complementary task types: GPQA (Rein et al., 2024) for graduate-level science question answering, LogiQA2.0 (Liu et al., 2023) for logical reasoning, MMLU (Hendrycks et al., 2021a) for broad knowledge and reasoning, GSM8K (Cobbe et al., 2021) for grade-school mathematics, MATH (Hendrycks et al., 2021b) for

Algorithm 1: OrchDebate: Orchestrated Multi-Agent Debate for One Episode

Input : Question q , reasoning model $\mathcal{M}$, orchestrator π , skill bank $\mathcal{B}$, token budget B , maximum steps T , minimum finalize step $t_{\min}$

Output : Final answer $\hat{y}$

```
1  $a_0 \leftarrow \mathcal{M}(q)$ ;  
2  $C_0 \leftarrow \text{Cost}(a_0)$ ;  
3  $s_0 \leftarrow \text{InitSDS}(q, a_0, C_0)$ ;  
4 for  $t = 0, 1, \dots, T - 1$  do  
5   if  $C_t \geq B$  then  
6      $\hat{y} \leftarrow \text{FallbackReadout}(s_t)$ ;  
7     return  $\hat{y}$ ;  
8    $\mathcal{L}_t \leftarrow \mathcal{L}(s_t)$ ;  
9    $\tilde{b}_t \leftarrow \pi(s_t)$ ;  
10   $b_t \leftarrow \text{Resolve}(\tilde{b}_t, \mathcal{L}_t)$ ;  
11  if  $b_t = \text{Finalize}(\psi)$  then  
12     $\hat{y} \leftarrow \text{Readout}(\psi, s_t)$ ;  
13    return  $\hat{y}$ ;  
14  sample  $v_t \sim P(\cdot \mid b_t)$ ;  
15   $r_t \leftarrow \text{Exec}(v_t, q, s_t; \mathcal{M})$ ;  
16   $C_{t+1} \leftarrow C_t + \text{Cost}(r_t)$ ;  
17   $s_{t+1} \leftarrow U(s_t, r_t, C_{t+1})$ ;  
18  if  $C_{t+1} \geq B$  then  
19     $\hat{y} \leftarrow \text{FallbackReadout}(s_{t+1})$ ;  
20    return  $\hat{y}$ ;  
21  $\hat{y} \leftarrow \text{FallbackReadout}(s_T)$ ;  
22 return  $\hat{y}$ ;
```

competition-style mathematical problem solving, and MuSR (Sprague et al., 2024) for multi-step reasoning. These benchmarks cover both settings where multi-agent deliberation is often beneficial and settings where unnecessary interaction can induce overthinking or answer drift. For OrchDebate, each benchmark is further split into an evolution stream and a held-out evaluation set; details are provided in Appendix C.1.

Baselines. We compare OrchDebate against single-agent prompting baselines, including direct chain-of-thought decoding (CoT) (Wei et al., 2022) and self-consistency (CoT-SC) (Wang et al., 2023), and representative multi-agent debate baselines, including MAD (Liang et al., 2024), S-MAD (Li et al., 2024b), S²-MAD (Zeng et al., 2025), GroupDebate (Liu et al., 2026), DMAD (Liu et al., 2025), and DynaDebate (Li et al., 2026). We additionally compare with DoWN (Eo et al., 2025) and iMAD (Fan et al., 2026), which adaptively decide whether debate should be invoked. We further study OrchDebate through internal ablations in Section 4.4, including w/o Orchestrator, w/o SDS, w/o Skills Bank, and w/o Self-Evolving. Detailed implementation settings of all compared baselines are provided in Appendix C.5.

Model Settings. Each LLM setting uses a single model family throughout OrchDebate, including orchestration, debate actions, and final judging when invoked. We use a bounded debate budget and enable online skill evolution only on the evolution split; the resulting skill bank is then frozen for all held-out evaluations. All methods are evaluated on the same held-out benchmark splits under a matched pipeline. Full configuration details are provided in Appendix C.

Evaluation Metrics. We evaluate methods along two primary axes: accuracy and token cost. We use task accuracy as the quality metric and the average total number of tokens consumed per instance as the cost metric. Unless otherwise stated, token counts include the full interaction cost under a unified accounting protocol, including debate turns, orchestration decisions, and judging steps when used.

4.2 Main Results

Table 1 reports results on six benchmarks under three LLM settings. Overall, OrchDebate achieves the strongest accuracy–efficiency trade-off among debate methods. It reaches the best or tied-best accuracy on all six benchmarks under GPT-5-nano, and the best accuracy on five of six benchmarks under both Claude-Haiku-4.5 and Qwen-3-8B. The gains are consistent on GPQA, MATH, and MuSR, where OrchDebate outperforms the strongest competing debate baseline under all three LLM settings. Exceptions are MMLU under Claude-Haiku-4.5, where S²-MAD remains slightly stronger (88.8 vs. 87.4), and GSM8K under Qwen-3-8B, where CoT-SC obtains the highest accuracy while OrchDebate ties the strongest debate baseline.

These gains do not simply reflect higher debate cost. Under GPT-5-nano, OrchDebate reaches the best or tied-best accuracy on all six benchmarks while using only 5.9k to 12.6k tokens, whereas strong fixed-topology baselines such as S²-MAD, DMAD, and GroupDebate typically require larger budgets. Similar patterns under Claude-Haiku-4.5 and Qwen-3-8B suggest that adaptive orchestration allocates debate computation more selectively.

OrchDebate also improves over direct CoT, particularly on harder benchmarks. Under GPT-5-nano, it improves over CoT by +4.3 on GPQA, +2.3 on MATH, and +8.0 on MuSR; the corresponding gains are +13.2, +14.4, and +7.0 under Claude-Haiku-4.5, and +5.6, +6.0, and +7.3 under Qwen-3-8B. On relatively easier settings such as

Model	Method	GPQA		LogiQA2.0		MMLU		GSM8K		MATH		MuSR	
		Acc(%)	Tok(k)	Acc(%)	Tok(k)	Acc(%)	Tok(k)	Acc(%)	Tok(k)	Acc(%)	Tok(k)	Acc(%)	Tok(k)
GPT-5-nano	CoT	66.9	8.8	72.2	4.4	84.5	3.5	94.5	2.8	79.3	4.4	65.3	6.8
	CoT-SC	65.6	94.1	72.8	46.2	84.5	37.4	94.8	28.6	79.5	44.0	65.7	75.2
	MAD	63.1	18.1	69.5	9.6	<u>80.2</u>	<u>8.1</u>	89.2	6.1	72.5	9.5	57.7	16.0
	S-MAD	66.9	10.9	73.2	9.7	<u>84.2</u>	<u>11.0</u>	95.5	12.6	72.2	11.6	68.3	21.7
	S ² -MAD	65.0	31.3	76.2	<u>17.0</u>	84.5	14.0	94.5	9.5	80.7	14.3	69.0	27.2
	DMAD	67.5	23.0	74.5	14.4	85.5	12.2	89.8	10.2	77.5	13.9	66.7	24.4
	GroupDebate	66.9	38.9	74.5	22.1	86.7	18.4	93.8	12.7	79.8	19.9	65.0	40.3
	DynaDebate	59.4	17.1	70.5	11.0	81.8	9.3	94.3	11.6	78.5	11.0	69.0	17.2
	OrchDebate	71.2	<u>9.6</u>	76.2	<u>8.8</u>	87.3	<u>8.1</u>	96.8	<u>5.9</u>	81.6	<u>8.3</u>	73.3	<u>12.6</u>
Claude-Haiku-4.5	CoT	56.2	2.2	77.8	1.4	86.2	1.1	95.5	0.9	64.2	1.7	68.7	3.4
	CoT-SC	60.0	25.1	81.3	17.5	87.8	14.3	96.8	10.4	70.5	18.6	68.0	39.7
	MAD	58.8	9.9	<u>78.2</u>	<u>4.1</u>	<u>84.5</u>	<u>3.4</u>	<u>92.5</u>	<u>2.7</u>	<u>65.2</u>	<u>4.4</u>	63.7	10.8
	S-MAD	66.2	37.5	80.5	26.1	80.8	21.6	96.2	9.7	67.7	17.4	70.0	48.1
	S ² -MAD	57.5	11.6	79.5	11.0	88.8	<u>11.9</u>	95.8	6.5	76.8	12.3	68.3	13.7
	DMAD	56.9	15.1	80.5	9.7	86.5	12.9	90.2	9.6	69.8	11.2	73.7	18.9
	GroupDebate	63.1	21.3	82.8	15.0	85.5	13.3	96.8	10.0	75.5	17.2	75.0	29.2
	DynaDebate	66.2	12.2	79.2	12.3	86.5	10.5	94.7	8.0	75.2	13.1	74.3	14.6
	OrchDebate	69.4	<u>7.8</u>	83.3	<u>8.1</u>	87.4	6.4	97.2	<u>4.9</u>	78.6	<u>9.5</u>	75.7	<u>10.2</u>
Qwen-3-8B	CoT	53.8	9.9	79.2	4.2	77.8	3.0	96.3	1.8	77.5	5.1	65.7	5.6
	CoT-SC	53.8	98.9	81.3	38.6	79.2	26.4	97.0	<u>18.7</u>	79.3	56.7	70.7	54.2
	MAD	52.5	27.6	78.7	11.8	80.2	9.0	<u>95.5</u>	<u>6.7</u>	79.3	18.7	65.3	16.9
	S-MAD	58.8	48.1	80.5	21.1	79.0	14.6	95.5	9.8	70.8	16.1	69.7	27.0
	S ² -MAD	56.2	41.9	82.2	14.3	79.8	10.4	96.0	11.7	80.8	25.9	65.0	18.9
	DMAD	54.4	45.2	79.8	22.6	80.2	17.9	95.5	14.6	79.8	28.8	71.3	28.4
	GroupDebate	57.5	60.0	82.3	23.6	79.2	17.6	96.5	14.7	81.8	40.2	69.0	33.2
	DynaDebate	54.4	18.0	79.3	10.4	78.3	13.2	95.8	9.0	80.0	11.7	68.3	13.3
	OrchDebate	59.4	<u>11.1</u>	84.0	<u>7.5</u>	81.3	<u>6.1</u>	96.5	7.4	83.5	<u>9.7</u>	73.0	<u>9.4</u>

Table 1: **Comparison of Token Cost and Accuracy Between OrchDebate and Other Methods.** The highest-accuracy results are shown in **bold**. For each dataset, we underline the result pair for the highest accuracy and the result pair for the lowest token cost, excluding CoT.

GSM8K, the margin is smaller, and CoT-SC can still achieve the best absolute result at much higher cost. Overall, these results show that OrchDebate improves answer quality while avoiding the unnecessary computation often incurred by static multi-agent protocols. Additional supplementary results for GPT-4o-mini are reported in Appendix D.

4.3 Comparison with Adaptive Debate Methods

To complement the broad comparison in Table 1, we evaluate DynaDebate, DoWN, and iMAD under GPT-5-nano. These methods respectively adapt reasoning paths and verification, use initial-answer confidence to invoke collaboration, and predict the need for debate from structured self-critique signals. All methods use the same held-out examples, answer extraction, scoring, and token accounting.

OrchDebate attains the highest accuracy on all six benchmarks and the best macro-average trade-off (81.1%/8.9k). Relative to iMAD, the strongest adaptive baseline by average accuracy, it improves the macro-average by 4.8 points while using approximately 18% fewer tokens. The per-task cost pattern reflects complementary forms of adaptivity: selective invocation is especially economical on several tasks, whereas whole-step orchestration

gives the strongest aggregate accuracy–efficiency balance. Full per-task results are reported in Appendix D.3.

Stability and call-level efficiency. Across three GPT-5-nano held-out runs, OrchDebate exceeds DynaDebate in mean accuracy on all six benchmarks and S²-MAD on five, while using fewer tokens than both throughout. Its run-to-run standard deviations are 0.3–0.7 accuracy points and 0.1–0.3k tokens. Call-level accounting further shows 23–29% fewer calls and 35–37% lower estimated hosted-model cost per query than DynaDebate, and approximately one quarter of the calls used by S²-MAD. Complete repeated-run and cost tables are provided in Appendices D.4 and D.5.

4.4 Ablation Study

To assess the contribution of each component in OrchDebate, we conduct module ablations on GPQA under three LLM settings, as shown in Table 2. Additional ablation results on MuSR and MMLU are provided in Appendix D.2. We consider four variants: w/o Orchestrator replaces adaptive orchestration with a fixed debate protocol with early stopping; w/o SDS keeps only a solver-confidence signal; w/o Skills Bank removes skill-based ac-

Method	GPT-5-nano		Claude-Haiku-4.5		Qwen-3-8B	
	Acc(%)	Tok(k)	Acc(%)	Tok(k)	Acc(%)	Tok(k)
MAD	63.1	18.1	58.8	9.9	52.5	27.6
OrchDebate	71.2	9.6 (-47.0%)	69.4	7.8 (-21.2%)	59.4	11.1 (-59.8%)
OrchDebate w/o Orchestrator	62.5	11.9 (-34.3%)	61.9	12.2 (+23.2%)	51.9	14.8 (-46.4%)
OrchDebate w/o SDS	67.5	16.5 (-8.8%)	60.6	14.3 (+44.4%)	52.5	18.9 (-31.5%)
OrchDebate w/o Skills Bank	66.2	13.2 (-27.1%)	62.5	9.8 (-1.0%)	54.4	16.3 (-40.9%)
OrchDebate w/o Self-Evolving	66.9	11.5 (-36.5%)	65.6	8.5 (-14.1%)	56.9	11.9 (-56.9%)

Table 2: Module ablation on GPQA across three LLM settings. We report accuracy and token cost, with token change in parentheses computed relative to MAD under the same model.

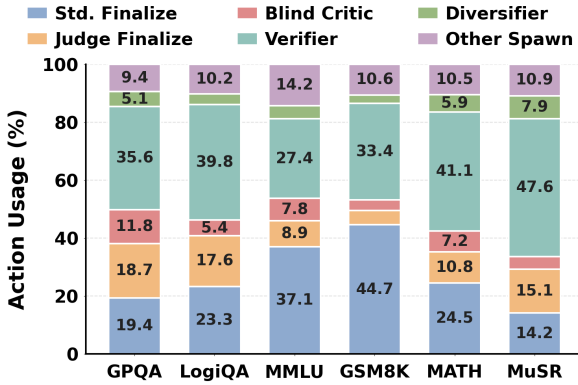


Figure 2: Distribution of OrchDebate controller actions across benchmarks. Each bar reports the percentage of decisions assigned to one of six action categories, summing to 100%.

tions; and w/o Self-Evolving disables online skill evolution.

Full OrchDebate performs best across all three LLMs, reaching 71.2%, 69.4%, and 59.4% under GPT-5-nano, Claude-Haiku-4.5, and Qwen-3-8B, respectively, while reducing token usage by 47.0%, 21.2%, and 59.8% relative to MAD. Removing either adaptive orchestration or SDS leads to clear performance drops, showing that effective debate control depends on both adaptive decisions and informative state signals. Removing the skills bank also hurts performance, suggesting that reusable debate-control skills provide more than simple continuation control. Finally, disabling self-evolution reduces accuracy from 71.2% to 66.9%, from 69.4% to 65.6%, and from 59.4% to 56.9%, showing that online evolution further improves the initial skill set.

4.5 Orchestration Behavior Analysis

To examine how OrchDebate adapts its control behavior, we analyze the distribution of controller

actions across benchmarks. For analysis, controller decisions are grouped into six categories: Standard Finalize, Judge Finalize, Blind Critic, Verifier, Diversifier, and Other Spawn. All results discussed in this subsection are based on GPT-5-nano. Figure 2 shows a clear relationship between task difficulty and the orchestrator’s action profile. On relatively easier benchmarks where the base solver is already strong, such as GSM8K and MMLU, the controller relies more on Standard Finalize, accounting for 44.7% and 37.1% of decisions, respectively, which suggests that OrchDebate avoids extending deliberation when evidence stabilizes early.

On harder benchmarks, the controller shifts more decisions toward continued scrutiny. Verifier becomes the dominant action on GPQA, LogiQA2.0, MATH, and MuSR, accounting for 35.6% to 47.6% of decisions, while Judge Finalize also rises to 15%–19% on GPQA, LogiQA2.0, and MuSR, indicating a greater reliance on arbitrated readout when evidence remains contested. Anchoring-aware actions are used more selectively: Blind Critic peaks at 11.8% on GPQA, and Diversifier is most visible on MuSR (7.9%) and MATH (5.9%). Taken together, these patterns suggest that OrchDebate does not uniformly prolong debate across tasks, but instead adapts its coordination strategy to the difficulty and disagreement structure of each benchmark.

4.6 Impact of Skill Evolution

As shown in Section 4.4, removing self-evolution hurts final performance. We next examine how this gain develops over time. Figure 3 compares the self-evolving skill bank with a fixed-bank counterpart on two representative hard benchmarks, GPQA and MuSR, under three LLM settings. We report cumulative disagreement win rate (CDWR) over five evolution checkpoints on the evolution stream,

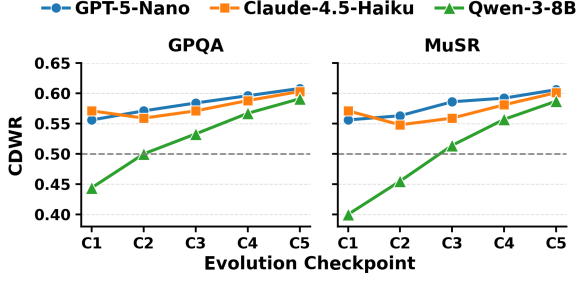


Figure 3: CDWR of OrchDebate against a fixed initial skill bank on GPQA and MuSR across five evolution checkpoints. The dashed line at 0.5 denotes parity.

denoted by C_1 – C_5 , where the dashed line at 0.5 marks parity on the subset of instances where the two variants disagree.

The overall trend is consistently positive. By C_5 , all six curves lie above parity, indicating that evolution more often helps than hurts once it changes the final outcome. The gains also accumulate over time rather than coming from isolated updates. This pattern is especially clear for Qwen-3-8B, whose CDWR rises from 44.4% to 59.1% on GPQA and from 40.0% to 58.7% on MuSR, while GPT-5-nano shows a nearly monotonic improvement on both benchmarks. Claude-Haiku-4.5 exhibits a small early dip, but recovers steadily and ends above 0.6 on both tasks. Since the evolving and fixed-bank variants often make the same final prediction, raw accuracy alone can understate the effect of bank updates. CDWR isolates the cases actually changed by evolution and shows that these changes become increasingly favorable as updates accumulate.

The update gates are also selective in practice. On the GPT-5-nano GPQA evolution stream, OrchDebate generated 11 candidates: four did not accumulate the minimum evidence, while seven reached the decision gate, of which three were promoted and four were retired; two retired candidates exceeded the degradation threshold. Re-running the complete evolution process under $\pm 50\%$ changes to the credit weights keeps frozen-bank accuracy within two held-out examples of the default, while removing SDS credit lowers accuracy and increases harmful answer flips. Appendix E reports the candidate outcomes and full sensitivity results.

4.7 Orchestrator–Debater Cross-Model Study

Finally, we examine whether OrchDebate’s gains come only from stronger debaters, or also from better orchestration. Figure 4 reports average ac-

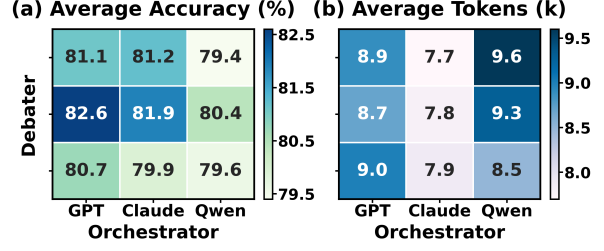


Figure 4: Cross-model OrchDebate results averaged over six benchmarks. Rows denote debaters and columns denote orchestrators. Left: accuracy (%); right: token cost (k).

curacy and token cost across the six benchmarks under different orchestrator–debater pairs.

Cross-model composition remains effective: the orchestrator and debater need not come from the same model family. However, the choice of orchestrator clearly matters. Averaged over all debaters, GPT-5-nano gives the highest accuracy as orchestrator (81.5%), while Claude-Haiku-4.5 is the most efficient, reaching comparable accuracy (81.0%) with the lowest token cost (7.8k). Qwen-3-8B is weaker in this role, with lower accuracy (79.8%) and higher cost (9.1k). The debater also matters: Claude-Haiku-4.5 is the strongest debater on average (81.6%), and the best pair uses Claude-Haiku-4.5 as debater with GPT-5-nano as orchestrator, reaching 82.6%. Overall, OrchDebate benefits from both capable debaters and effective state-conditioned orchestration.

5 Conclusion

We presented **OrchDebate**, a framework that re-frames multi-agent debate as state-conditioned orchestration over reusable debate-control skills rather than a fixed interaction protocol. By combining a structured debate state, adaptive skill routing, and a self-evolving skill bank, OrchDebate allocates debate computation more selectively and improves coordination from its own debate experience. Across six reasoning benchmarks and three LLM families, it consistently achieves a stronger trade-off between accuracy and efficiency than representative fixed and adaptive debate baselines. The controller shifts computation toward verification on harder instances while terminating more directly once evidence stabilizes, with gains that remain consistent across tasks and model families. These results suggest that future MAD systems should focus on orchestrating debate online rather than scaling static interaction structures.

6 Limitations

Although OrchDebate is designed to improve the trade-off between accuracy and efficiency in debate-based reasoning, it still incurs additional inference overhead relative to single-pass prompting, since solving a query may require multiple rounds of orchestration and agent interaction. This added cost may limit its applicability in highly latency-sensitive scenarios. In addition, while our experiments cover six representative benchmarks, they remain focused on structured reasoning tasks with relatively well-defined answers. The extent to which the same orchestration mechanism transfers to broader settings, such as open-ended generation, tool-using agents, or other long-horizon tasks, remains to be established in future work. Finally, skill evolution uses episode-level correctness on an evolution stream and may acquire task-family-specific coordination preferences; transferring a frozen bank to domains with different interaction patterns remains an open question.

7 Acknowledgements

This research is supported by the National Research Foundation, Singapore and Infocomm Media Development Authority under its Trust Tech Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore and Infocomm Media Development Authority.

References

- Jonas Becker, Lars Benedikt Kaesberg, Niklas Bauer, Jan Philip Wahle, Terry Ruas, and Bela Gipp. 2025. [Mallm: Multi-agent large language models framework](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 418–439, Suzhou, China. Association for Computational Linguistics.
- Justin Chen, Swarnadeep Saha, and Mohit Bansal. 2024. [Reconcile: Round-table conference improves reasoning via consensus among diverse llms](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7066–7085, Bangkok, Thailand. Association for Computational Linguistics.
- Hyeong Kyu Choi, Xiaojin Zhu, and Sharon Li. 2025. [Debate or vote: Which yields better decisions in multi-agent large language models?](#) In *Advances in Neural Information Processing Systems*, volume 38, pages 112802–112834. NeurIPS 2025 Spotlight.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. [Training verifiers to solve math word problems](#). *Preprint*, arXiv:2110.14168.
- Yu Cui, Hang Fu, Haibin Zhang, Licheng Wang, and Cong Zuo. 2025. [Free-mad: Consensus-free multi-agent debate](#). *Preprint*, arXiv:2509.11035. ICLR 2026 withdrawn submission.
- Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2024. [Improving factuality and reasoning in language models through multiagent debate](#). In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 11733–11763. PMLR.
- Sugyeong Eo, Hyeonseok Moon, Evelyn Hayoon Zi, Chanjun Park, and Heuseok Lim. 2025. [Debate only when necessary: Adaptive multiagent collaboration for efficient llm reasoning](#). *Preprint*, arXiv:2504.05047.
- Wei Fan, JinYi Yoon, and Bo Ji. 2026. [iMAD: Intelligent multi-agent debate for efficient and accurate llm inference](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(35):29403–29411.
- Wenyuan Gu, Haowen Wang, Jiale Han, Xiang Li, Zhixuan Wu, Hongru Xiao, and Bo Cheng. 2026. [Analyze–Compose–Execute: A dynamic dialogue framework for multi-agent debate](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(36):30825–30833.
- Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xi-angliang Zhang. 2024. [Large Language Model based multi-agents: A Survey of Progress and Challenges](#). In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pages 8048–8057. Survey Track.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. [Measuring massive multitask language understanding](#). In *International Conference on Learning Representations (ICLR)*.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. [Measuring mathematical problem solving with the MATH dataset](#). In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1.
- Lars Benedikt Kaesberg, Jonas Becker, Jan Philip Wahle, Terry Ruas, and Bela Gipp. 2025. [Voting or consensus? decision-making in multi-agent debate](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 11640–11671, Vienna, Austria. Association for Computational Linguistics.

- Junyou Li, Qin Zhang, Yangbin Yu, Qiang Fu, and Deheng Ye. 2024a. [More agents is all you need](#). *Transactions on Machine Learning Research*. Accepted by TMLR.
- Yunxuan Li, Yibing Du, Jiageng Zhang, Le Hou, Peter Grabowski, Yeqing Li, and Eugene Ie. 2024b. [Improving multi-agent debate with sparse communication topology](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7281–7294, Miami, Florida, USA. Association for Computational Linguistics.
- Zhenghao Li, Zhi Zheng, Wei Chen, Jielun Zhao, Yong Chen, Tong Xu, and Enhong Chen. 2026. [Dynadebate: Breaking homogeneity in multi-agent debate with dynamic path generation](#). *Preprint*, arXiv:2601.05746.
- Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. 2024. [Encouraging divergent thinking in large language models through multi-agent debate](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17889–17904, Miami, Florida, USA. Association for Computational Linguistics.
- Hanmeng Liu, Jian Liu, Leyang Cui, Zhiyang Teng, Nan Duan, Ming Zhou, and Yue Zhang. 2023. [Logiqa 2.0 — an improved dataset for logical reasoning in natural language understanding](#). *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:2947–2962.
- Tongxuan Liu, Xingyu Wang, Weizhe Huang, Wenjiang Xu, Yuting Zeng, Lei Jiang, Hailong Yang, and Jing Li. 2026. [Groupdebate: Enhancing the efficiency of multi-agent debate using group discussion](#). In *Proceedings of the 25th International Conference on Autonomous Agents and Multiagent Systems*, pages 1147–1155, Paphos, Cyprus.
- Yexiang Liu, Jie Cao, Zekun Li, Ran He, and Tieniu Tan. 2025. [Breaking mental set to improve reasoning through diverse multi-agent debate](#). In *The Thirteenth International Conference on Learning Representations*, Singapore. ICLR 2025 Poster.
- Junyu Luo, Weizhi Zhang, Ye Yuan, Yusheng Zhao, Junwei Yang, Yiyang Gu, Bohan Wu, Binqi Chen, Ziyue Qiao, Qingqing Long, Rongcheng Tu, Xiao Luo, Wei Ju, Zhiping Xiao, Yifan Wang, Meng Xiao, Chenwu Liu, Jingyang Yuan, Shichang Zhang, and 7 others. 2025. [Large Language Model Agent: A Survey on Methodology, Applications and Challenges](#). *Preprint*, arXiv:2503.21460.
- Zehua Pei, Hui-Ling Zhen, Shixiong Kai, Sinno Jialin Pan, Yunhe Wang, Mingxuan Yuan, and Bei Yu. 2025. [Scope: Prompt evolution for enhancing agent effectiveness](#). *Preprint*, arXiv:2512.15374.
- Libin Qiu, Zhirong Gao, Junfu Chen, Yuhang Ye, Weizhi Huang, Xiaobo Xue, Wenkai Qiu, and Shuo Tang. 2026. [Autorefine: From trajectories to reusable expertise for continual llm agent refinement](#). *Preprint*, arXiv:2601.22758.
- David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2024. [Gpqa: A graduate-level google-proof q&a benchmark](#). In *Proceedings of the First Conference on Language Modeling (COLM)*, Philadelphia, Pennsylvania, United States.
- Zayne Rea Sprague, Xi Ye, Kaj Bostrom, Swarat Chaudhuri, and Greg Durrett. 2024. [Musr: Testing the limits of chain-of-thought with multistep soft reasoning](#). In *The Twelfth International Conference on Learning Representations (ICLR)*. Spotlight.
- Yiliu Sun, Zicheng Zhao, Sheng Wan, and Chen Gong. 2025. [Cortexdebate: Debating sparsely and equally for multi-agent debate](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 9503–9523, Vienna, Austria. Association for Computational Linguistics.
- Arne Tillmann. 2025. [Literature review of multi-agent debate for problem-solving](#). *Preprint*, arXiv:2506.00066.
- Khanh-Tung Tran, Dung Dao, Minh-Duong Nguyen, Quoc-Viet Pham, Barry O’Sullivan, and Hoang D. Nguyen. 2025. [Multi-Agent Collaboration Mechanisms: A Survey of LLMs](#). *Preprint*, arXiv:2501.06322.
- Chao Wang, Han Lin, Huaze Tang, Huijing Lin, and Wenbo Ding. 2026. [RUMAD: Reinforcement-unifying multi-agent debate](#). In *Proceedings of the 25th International Conference on Autonomous Agents and Multiagent Systems*, pages 3695–3703, Paphos, Cyprus.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. [Self-consistency improves chain of thought reasoning in language models](#). In *The Eleventh International Conference on Learning Representations*, Kigali, Rwanda. ICLR 2023 Poster.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. [Chain-of-thought prompting elicits reasoning in large language models](#). In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. NeurIPS 2022 Main Conference Track.
- Zhiyuan Weng, Guikun Chen, and Wenguan Wang. 2025. [Do as we do, not as you think: the conformity of large language models](#). In *International Conference on Learning Representations*. ICLR 2025 Oral.
- Haolun Wu, Zhenkun Li, and Lingyao Li. 2025a. [Can llm agents really debate? a controlled study of multi-agent debate in logical reasoning](#). *Preprint*, arXiv:2511.07784.

- Jing Wu, Suiyao Chen, Inseok Heo, Sasha Gutfraind, Shengjie Liu, Chen Li, Bharathi Srinivasan, Xian Zhang, and Michael Sharps. 2025b. [Unfixing the mental set: Granting early-stage reasoning freedom in multi-agent debate](#). ICML 2025 Workshop on Multi-Agent Systems in the Era of Foundation Models.
- Andrea Wynn, Harsh Satija, and Gillian K. Hadfield. 2025. [Talk isn’t always cheap: Understanding failure modes in multi-agent debate](#). *Preprint*, arXiv:2509.05396.
- Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, and 10 others. 2025. [The Rise and Potential of Large Language Model Based Agents: A Survey](#). *Science China Information Sciences*, 68(2):121101.
- Kai Xiong, Xiao Ding, Yixin Cao, Ting Liu, and Bing Qin. 2023. [Examining inter-consistency of large language models collaboration: An in-depth analysis via debate](#). In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 7572–7590, Singapore. Association for Computational Linguistics.
- Zhenran Xu, Senbao Shi, Baotian Hu, Jindi Yu, Dongfang Li, Min Zhang, and Yuxiang Wu. 2023. [Towards reasoning in large language models via multi-agent peer review collaboration](#). *Preprint*, arXiv:2311.08152.
- Liming Yang, Junyu Luo, Xuanzhe Liu, Yiling Lou, and Zhenpeng Chen. 2026a. [BAMAS: Structuring budget-aware multi-agent systems](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(35):29802–29810. Oral paper.
- Yongjin Yang, Euiin Yi, Jongwoo Ko, Kimin Lee, Zhijing Jin, and Se-Young Yun. 2025. [Revisiting multi-agent debate as test-time scaling: A systematic study of conditional effectiveness](#). *Preprint*, arXiv:2505.22960.
- Yutao Yang, Junsong Li, Qianjun Pan, Bihao Zhan, Yuxuan Cai, Lin Du, Jie Zhou, Kai Chen, Qin Chen, Xin Li, Bo Zhang, and Liang He. 2026b. [Autoskill: Experience-driven lifelong learning via skill self-evolution](#). *Preprint*, arXiv:2603.01145.
- Zhangyue Yin, Qiushi Sun, Cheng Chang, Qipeng Guo, Junqi Dai, Xuanjing Huang, and Xipeng Qiu. 2023. [Exchange-of-thought: Enhancing large language model capabilities through cross-model communication](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15135–15153, Singapore. Association for Computational Linguistics.
- Yuting Zeng, Weizhe Huang, Lei Jiang, Tongxuan Liu, XiTai Jin, Chen Tianyang Tiana, Jing Li, and Xiaohua Xu. 2025. [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)*, pages 9393–9408, Albuquerque, New Mexico. Association for Computational Linguistics.
- Hangfan Zhang, Zhiyao Cui, Jianhao Chen, Xinrun Wang, Qiaosheng Zhang, Zhen Wang, Dinghao Wu, and Shuyue Hu. 2025. [Stop overvaluing multi-agent debate — we must rethink evaluation and embrace model heterogeneity](#). *Preprint*, arXiv:2502.08788.
- Haozhen Zhang, Quanyu Long, Jianzhu Bao, Tao Feng, Weizhi Zhang, Haodong Yue, and Wenya Wang. 2026a. [Memskill: Learning and evolving memory skills for self-evolving agents](#). *Preprint*, arXiv:2602.02474.
- Miao Zhang, Junsik Kim, Siyuan Xiang, Jian Gao, and Cheng Cao. 2026b. [Dynamic role assignment for multi-agent debate](#). *Preprint*, arXiv:2601.17152.
- Qian Zhang, Jinyi Liu, Yan Zheng, Hebin Liang, and Lanjun Wang. 2026c. [Key decision-makers in multi-agent debates: Who holds the power?](#) *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(35):29883–29891.
- Zeqi Zhou, Fang Wu, Shayan Talaei, Haokai Zhao, Meixin Cheng, Tinson Xu, Amin Saberi, and Yejin Choi. 2025. [When to trust context: Self-reflective debates for context reliability](#). *Preprint*, arXiv:2506.06020.

A Notation

We summarize the core symbols used throughout the method. q denotes an input question drawn from distribution $\mathcal{D}$. $\mathcal{M}$ denotes the LLM used in OrchDebate. In our main experiments, the same model family is used throughout the framework to instantiate debate agents, the controller π , and the final judge when needed. The controller π selects debate-control skills from the skill bank $\mathcal{B}$ based on a structured rendering of the current Structured Debate State (SDS). At step t , the debate is summarized by the Structured Debate State (SDS)

$$s_t = (s_t^{\text{ans}}, s_t^{\text{ver}}, s_t^{\text{ctl}}, s_t^{\text{bud}}),$$

where s_t^{ans} summarizes the current answer landscape, s_t^{ver} summarizes the current verification status, s_t^{ctl} collects control-relevant coordination cues, and s_t^{bud} tracks progress and remaining budget. In particular, a_t denotes the current leading answer, δ_t indicates whether the leader changed relative to the previous step, and $\beta_t = B - C_t$ is the remaining budget after cumulative cost C_t . A control skill

is denoted by $b_k \in \mathcal{B}$, and a concrete executable realization of that skill is a variant v . The final prediction is $\hat{y}$, the gold answer is y^* , and $C(q)$ is the total inference cost at termination.

B Extended Method Details

B.1 Structured Debate State Details

The main text defines the Structured Debate State (SDS) as a compact control-facing summary:

$$s_t = (s_t^{\text{ans}}, s_t^{\text{ver}}, s_t^{\text{ctl}}, s_t^{\text{bud}}), \quad (12)$$

where the four components correspond to answer status, verification status, control cues, and resource budget.

In implementation, s_t^{ans} records the current leading answer, agreement strength, answer margin, recent answer shifts, and short-term stability. s_t^{ver} tracks unresolved checks, verification coverage, and accumulated verification debt. s_t^{ctl} contains control-relevant signals such as anchoring risk, response diversity, and stop-or-escalate cues. Finally, s_t^{bud} records the debate step and the remaining budget $\beta_t = B - C_t$.

SDS is realized as a compact state dictionary grouped around these four components. The controller operates on a structured rendering of SDS rather than on the full debate transcript. This keeps orchestration cheaper and more stable while preserving the signals needed for routing, verification, escalation, and stopping.

B.2 Within-Episode Adaptive Debate Control

OrchDebate operates over a bank of typed debate-control skills,

$$b = (\text{op}, r, \nu, \phi, \psi), \quad (13)$$

where $\text{op} \in \{\text{spawn}, \text{finalize}\}$ is the operation type, r is the assigned reasoning role, ν is the visibility policy, ϕ is the focus scope, and ψ is the readout rule at termination. Spawn skills use (r, ν, ϕ) to construct the next reasoning action, whereas finalize skills use ψ to read out the final answer from the current state. The implementation instantiates a selected bank of executable skills from this typed space rather than treating every Cartesian-product combination as available.

At step t , the controller first proposes a skill $\tilde{b}_t = \pi(s_t)$. The executable skill is then obtained through

$$b_t = \text{Resolve}(\tilde{b}_t, \mathcal{L}_t), \quad \mathcal{L}_t = \mathcal{L}(s_t), \quad (14)$$

where $\mathcal{L}_t$ contains the skills that are legal under the current state. In particular,

$$\mathcal{L}(s_t) = \mathcal{B}_{\text{spawn}} \cup \{b \in \mathcal{B}_{\text{final}} : t \geq t_{\min}\}. \quad (15)$$

If the proposal can be matched to a valid member of $\mathcal{L}_t$, the resolver returns that skill; otherwise, it invokes the deterministic rule-based fallback $\pi_{\text{rule}}(s_t) \in \mathcal{L}_t$. The fallback covers malformed outputs, unknown skill identifiers, and state-incompatible actions.

The orchestrator is functionally separated from the reasoning agents and the final answer judge. It reads the compact SDS and chooses an executable control action, but does not generate task reasoning or directly determine the answer. This single constrained interface allows the controller to move among criticism, diversification, verification, and termination as the debate state changes.

B.3 Proof of Control Properties

The finite-horizon property in Section 3.4 follows directly from the legal resolver and the explicit stopping conditions in Algorithm 1. We assume that $\text{Resolve}(\tilde{b}_t, \mathcal{L}_t) \in \mathcal{L}_t$ for every reachable state, that cumulative cost is non-decreasing, and that the readout functions are defined on every reachable state.

Proof. At every control step, the executed skill is the output of Resolve and therefore belongs to $\mathcal{L}_t$, establishing legality. By Eq. 15, finalize skills belong to the legal set only when $t \geq t_{\min}$, so a selected finalize skill cannot be executed earlier. Each non-finalizing iteration performs exactly one control transition and advances the loop index, while the loop contains at most T iterations. The procedure returns immediately when either explicit budget check succeeds; if neither check succeeds and no finalize skill is selected, the fallback readout after the loop returns from s_T . Hence the episode returns after at most T control transitions or earlier at the budget limit. $\square$

B.4 Across-Episode Online Skill Evolution

For each base skill b_k , OrchDebate maintains one incumbent realization v_k^* and a probation set $\mathcal{P}_k$:

$$\mathcal{V}_k = \{v_k^*\} \cup \mathcal{P}_k. \quad (16)$$

When b_k is selected, the incumbent is normally executed and a probationary candidate is explored

with bounded probability:

$$P(v | b_k) = \begin{cases} 1 - p, & v = v_k^*, \\ \frac{p}{|\mathcal{P}_k|}, & v \in \mathcal{P}_k. \end{cases} \quad (17)$$

When $\mathcal{P}_k$ is empty, v_k^* is selected deterministically. The evolution loop follows four stages.

Trajectory credit. For evolution episode e , let z_e denote OrchDebate correctness, z_e^0 denote direct-solve correctness, and c_e denote normalized token cost. An executed variant receives

$$a_e(v) = (z_e - z_e^0) + \alpha \Delta_{\text{SDS}}(e) - \lambda c_e, \quad (18)$$

where $\Delta_{\text{SDS}}(e)$ summarizes changes in verification coverage, unresolved issues, and vote instability. The accumulated candidate statistics are

$$\begin{aligned} \hat{A}(v) &= \frac{1}{N(v)} \sum_{e:v \in e} a_e(v), \\ \hat{D}(v) &= \frac{1}{N(v)} \sum_{e:v \in e} \mathbf{1}[z_e^0 = 1 \wedge z_e = 0]. \end{aligned} \quad (19)$$

Diagnostic-driven refinement. Recurring patterns such as degradation relative to direct solve, unresolved verification debt, unstable votes, or excessive cost identify a source skill b_k . The mutator receives the source control skill and an aggregated recurring diagnostic pattern, but not the gold answer, a reference rationale, or an item-level corrective target. It proposes a local realization edit to wording, visibility phrasing, focus emphasis, or execution configuration while preserving the skill’s operation, role identity, control purpose, and output schema.

Future-only probation. A newly generated candidate enters $\mathcal{P}_k$ with $N(v) = 0$. The episodes that triggered its creation are not reused to score it. Only later evolution episodes in which the candidate is actually executed update $N(v)$, $\hat{A}(v)$, and $\hat{D}(v)$.

Promotion, retirement, and freezing. A candidate reaches the decision gate only after sufficient future exposure and is promoted when

$$N(v) \geq n_{\min}, \quad \hat{A}(v) \geq \epsilon, \quad \hat{D}(v) \leq \tau_D. \quad (20)$$

A promoted candidate replaces the incumbent realization; a candidate that reaches the gate without satisfying the advantage and degradation conditions is retired. After the evolution stream, the resulting bank is frozen and no mutation, credit update, promotion, or configuration selection is performed on held-out examples.

Algorithm 2: OrchDebate: Across-Episode Online Skill Evolution

Input : Evolution stream $\mathcal{E}$, skill bank $\mathcal{B}$, mutator μ , exposure probability p , thresholds $n_{\min}, \epsilon, \tau_D$

- 1 **foreach** episode $e \in \mathcal{E}$ in stream order **do**
- 2 execute OrchDebate with incumbent/candidate sampling and record trajectory τ_e ;
- 3 compute trajectory credit and update future evidence for candidates executed in e ;
- 4 update recurring diagnostic patterns from completed evolution episodes;
- 5 **foreach** new recurring pattern g **do**
- 6 identify source skill b_k ;
- 7 $\tilde{v} \leftarrow \mu(g, b_k)$;
- 8 add $\tilde{v}$ to $\mathcal{P}_k$ with $N(\tilde{v}) = 0$;
- 9 **foreach** candidate v with $N(v) \geq n_{\min}$ **do**
- 10 **if** $\hat{A}(v) \geq \epsilon$ **and** $\hat{D}(v) \leq \tau_D$ **then**
- 11 promote v and replace the incumbent for its base skill;
- 12 **else**
- 13 retire v ;
- 14 freeze the resulting skill bank before held-out evaluation;

C Reproducibility and Experimental Details

C.1 Evolution Stream, Held-out Evaluation, and Feedback Access

For each benchmark, we use a 60/40 split between an evolution stream and a held-out evaluation set. The evolution stream is used for diagnostic aggregation, episode-level credit assignment, mutation, future-only probation, promotion or retirement, and configuration selection. After this stage, the skill bank and all configurations are frozen. The held-out examples are used only for final scoring and never for mutation, credit assignment, promotion, or model-selection decisions.

Evolution relies on delayed episode-level correctness relative to direct solve, together with SDS improvement, token cost, and the separate degradation condition. It requires no additional external corpus or human-written correction traces. The mutator receives neither the gold answer nor a reference rationale or item-level corrective target; it receives only the source control skill and an aggregated recurring diagnostic pattern. We therefore characterize the mechanism as outcome-guided control-skill evolution without additional external training data, rather than item-level answer-prompt optimization.

C.2 Model, Decoding, and Runtime Configuration

Each primary setting uses one model family throughout OrchDebate: GPT-5-nano, Claude-Haiku-4.5, or Qwen-3-8B is used for orchestration, debate actions, and judging when required. All calls are issued through a unified chat-style interface, and structured controller outputs are enforced through JSON-only prompting rather than native response-format APIs. Solver, orchestrator, and judge calls use deterministic decoding, while debate-agent calls use a higher decoding temperature to encourage diverse counterarguments. Separate output caps are used for solver, debate-agent, orchestration, and judge calls.

OrchDebate uses a bounded debate horizon, a bounded total inference budget, a bounded number of spawned actions, and a small set of focused verification issues. Judge-style and other non-incumbent finalization paths are enabled only after at least one debate action has been executed. Stable-stop decisions additionally depend on answer stability, verification progress, and disagreement signals.

C.3 Prompt Interfaces and Initial Skill Bank

The implementation uses separate prompt families with explicit information boundaries. The initial solver receives only the question and returns an initial answer with a concise reasoning brief. The SDS constructor and updater receive the current answers, checks, control cues, and budget, and return the structured fields in Eq. 12. The orchestrator receives the compact SDS together with the legal skill identifiers and returns one typed skill proposal; it is not asked to produce task reasoning or a final answer.

Critic, verifier, and diversifier prompts receive the question together with the visibility and focus information allowed by the selected spawn skill, and return one role-specific reasoning action. Readout prompts receive the current debate state and the selected incumbent-, vote-, or judge-based rule and return $\hat{y}$ without initiating another debate action. The diagnostic aggregator consumes completed evolution trajectories and returns an aggregated control-level pattern. The mutator receives only this pattern and the source skill realization, and returns a local schema-preserving edit.

The executable action vocabulary uses roles from {critic, verifier, diversifier}, visibility poli-

cies from {blind, table-only, informed}, focus scopes including all claims, unresolved issues, conflict sets, and top- k issues, and readout rules including incumbent, vote-based, and judge-based finalization. The initial skill bank is a selected subset of this typed space, with one incumbent realization per included base skill.

C.4 Evolution Hyperparameters and Mutation Settings

The main credit configuration uses $\alpha_0 = 0.25$ for SDS improvement and $\lambda_0 = 0.10$ for token cost. Both the SDS-change and cost terms are normalized before weighting. The exposure probability p and promotion parameters $n_{\min}$, ϵ , and τ_D are fixed before held-out evaluation and held constant throughout the sensitivity study in Appendix E.2.

Mutation is restricted to local realization-level changes in wording, visibility phrasing, focus emphasis, or execution configuration. It may not change the base operation, role identity, semantic control purpose, or output schema. New candidates start with no inherited evidence, and their trigger episodes are excluded from probationary evaluation.

C.5 Baseline Implementations and Fairness

All baselines use the same underlying model family, held-out examples, answer extraction, scoring rules, and full token-accounting protocol as OrchDebate. Because the compared methods are defined by different interaction structures, we do not force identical agent counts or debate rounds. Instead, each method retains its characteristic mechanism, while its configuration is selected only on the evolution split from a small fixed set and then frozen before held-out evaluation. Cross-provider result blocks test robustness across model families rather than ranking providers.

CoT (Wei et al., 2022). Chain-of-Thought uses a single reasoning call that asks the model to reason step by step before producing a final answer.

CoT-SC (Wang et al., 2023). CoT with self-consistency samples 20 independent Chain-of-Thought responses at temperature 0.7 and returns the majority answer.

MAD (Liang et al., 2024). MAD uses an affirmative debater, a negative debater, and a moderator for up to two rounds. The moderator checks whether

a clear preference has emerged after each round, with a final judging step used when needed.

S-MAD (Li et al., 2024b). S-MAD uses six agents connected by a sparse communication topology for five rounds. Each agent observes only its graph neighbors, and the final answer is determined by majority vote.

S²-MAD (Zeng et al., 2025). S²-MAD uses five agents in two groups for up to four rounds, with two rounds per stage. Agents debate within groups and exchange group summaries across stages, with early stopping when group-level summaries align.

DMAD (Liu et al., 2025). DMAD runs direct answering, context-aware chain-of-thought, and decomposition-based chain-of-thought in parallel at each round, followed by a selector. We use two rounds.

DynaDebate (Li et al., 2026). DynaDebate generates three reasoning paths, assigns one to each agent, and runs up to two debate rounds. Persistent disagreement may trigger one bounded verification step whose evidence is used in the subsequent round.

GroupDebate (Liu et al., 2026). GroupDebate uses five agents split into two groups for four rounds, with two intra-group rounds per stage. Group summaries mediate inter-group exchange, and the final answer is selected by majority vote.

DoWN (Eo et al., 2025). DoWN uses confidence from the initial answer to decide whether debate is necessary. When debate is triggered, agents refine their responses using peer outputs and confidence signals; otherwise, the initial response is retained. The trigger configuration is selected on the evolution split and frozen before held-out evaluation.

iMAD (Fan et al., 2026). iMAD derives interpretable features from a structured self-critique and applies its lightweight debate-decision classifier to determine whether MAD should be invoked. We follow the released selective-debate mechanism and freeze all evaluation-time settings before scoring the held-out split.

C.6 Benchmark Processing and Cost Accounting

The orchestrator consumes SDS together with a compact rendering of recent vote patterns, unresolved verification items, and short recent-agent

traces rather than the full raw transcript. Spawn actions are parameterized by role, visibility policy, and focus scope; finalize actions are parameterized by incumbent, vote-based, or judge-based readout. Issue construction is lightweight and can derive focused verification items directly from the solver brief without an additional issue-miner call on every example.

The evaluation pipeline normalizes tasks into standard multiple-choice, open-answer math, and entity-choice interfaces. It applies task-specific answer extraction, stronger normalization for open-answer mathematics, and candidate canonicalization for entity-choice predictions. The same processing rules are used for direct solve and all debate methods.

For hosted model m , estimated per-query cost is computed from recorded input and output tokens across all calls:

$$\text{Cost}_m(q) = \sum_j (n_j^{\text{in}} p_m^{\text{in}} + n_j^{\text{out}} p_m^{\text{out}}), \quad (21)$$

where p_m^{in} and p_m^{out} are the provider rates used for the experiment. Qwen-3-8B is locally deployed, so no provider cost is reported. We report calls and token-derived cost rather than wall-clock latency because hosted APIs and local inference are not directly comparable under network, queueing, and hardware effects.

D Extended Experimental Results

This section collects supplementary quantitative results omitted from the main text for space reasons.

D.1 Additional Results under GPT-4o-mini

Table 3 reports the corresponding results under GPT-4o-mini. Although these results are presented only as supplementary evidence, they provide a useful reference point for cross-model consistency under the same external baselines and benchmark protocol.

Under GPT-4o-mini, OrchDebate attains the highest accuracy on five of the six benchmarks, including GPQA, LogiQA2.0, GSM8K, MATH, and MuSR, while S-MAD remains strongest on MMLU. At the same time, MAD continues to be the lowest-token debate baseline across all six benchmarks. These results are broadly consistent with the main findings in the paper: adaptive orchestration yields a stronger balance between answer accuracy and inference cost than fixed debate

Model	Method	GPQA		LogiQA2.0		MMLU		GSM8K		MATH		MuSR	
		Acc(%)	Tok(k)	Acc(%)	Tok(k)	Acc(%)	Tok(k)	Acc(%)	Tok(k)	Acc(%)	Tok(k)	Acc(%)	Tok(k)
GPT-4o-mini	CoT	38.1	0.9	61.8	0.7	75.2	0.6	92.2	0.5	60.5	0.5	62.7	2.7
	CoT-SC	40.6	9.5	61.3	7.9	76.5	6.3	92.8	5.0	62.8	5.3	63.0	28.2
	MAD	41.2	4.4	58.8	2.8	72.7	2.5	91.7	2.3	58.0	3.8	66.3	8.2
	S-MAD	46.9	25.7	64.5	17.1	80.2	<u>14.5</u>	94.2	11.6	62.3	11.2	67.0	37.0
	S ² -MAD	41.9	9.1	64.8	8.5	79.5	6.9	94.8	5.8	64.0	12.8	63.7	12.1
	DMAD	40.6	12.6	64.2	8.6	79.5	7.9	93.2	7.0	61.7	12.5	66.3	13.1
	GroupDebate	40.6	15.6	67.2	11.1	78.2	9.4	95.5	13.2	62.3	14.7	66.0	23.0
	DynaDebate	43.1	11.3	63.5	8.8	78.5	8.8	92.7	9.4	63.7	8.9	66.7	13.2
	OrchDebate	48.1	<u>7.5</u>	67.5	<u>6.8</u>	79.3	5.9	95.7	<u>5.1</u>	65.8	<u>7.1</u>	67.3	<u>11.7</u>

Table 3: **Additional main-table-style results under GPT-4o-mini.** We report the same benchmarks and metrics as in Table 1. The highest-accuracy results are shown in **bold**. For each dataset, we underline the result pair for the highest accuracy and the result pair for the lowest token cost, excluding CoT.

topologies, while the strongest static baseline can still vary across tasks.

D.2 Additional Ablation Results

To examine whether the component effects observed on GPQA also generalize to other task types, we further report module ablations on MuSR and MMLU under the same three LLM settings, as shown in Table 4. The overall trends remain broadly consistent with the main ablation. On MuSR, full OrchDebate performs best across all three LLMs, reaching 73.3%, 75.7%, and 73.0% under GPT-5-nano, Claude-Haiku-4.5, and Qwen-3-8B, while also reducing token usage by 21.3%, 5.6%, and 44.4% relative to MAD. Removing the orchestrator again causes the largest degradation, dropping accuracy to 66.7%, 69.7%, and 66.7%, which suggests that adaptive orchestration remains especially important when problems require sustained multi-step coordination. Removing SDS or the skills bank also consistently hurts performance, indicating that effective control on MuSR still depends on both structured state and reusable debate-control actions. Finally, disabling self-evolution reduces accuracy from 73.3% to 70.3%, from 75.7% to 74.3%, and from 73.0% to 71.0%, showing that the gains from online evolution extend beyond GPQA.

On MMLU, the pattern is similar but the margins are smaller, which is consistent with the fact that broader knowledge-and-reasoning questions often require less extended deliberation than harder reasoning-focused benchmarks. Even so, full OrchDebate still performs best among the ablation variants under all three LLM settings, reaching 87.3%, 87.4%, and 81.3%. Removing the orchestrator again leads to the most substantial drop, reducing accuracy to 84.4%, 84.8%, and 79.2%,

while removing SDS or the skills bank also weakens performance in most cases. Disabling self-evolution yields a smaller but still consistent decline, from 87.3% to 86.8%, from 87.4% to 86.9%, and from 81.3% to 81.0%. Compared with MuSR, the cost advantage on MMLU is less pronounced and can even reverse under some settings, but the full system still provides the strongest overall balance between answer quality and inference cost among the ablation variants in this comparison.

D.3 Comparison with Adaptive Debate Methods

We compare representative adaptive debate methods under GPT-5-nano. DoWN uses initial-answer confidence to invoke collaboration only when needed, iMAD uses structured self-critique features and a lightweight debate-decision classifier, and DynaDebate adapts reasoning paths with bounded verification under persistent disagreement. All methods use the same held-out examples, answer extraction, scoring, and token accounting.

Table 5 shows that OrchDebate attains the highest accuracy on all six benchmarks and the lowest macro-average token cost. Relative to iMAD, the strongest adaptive baseline by average accuracy, it improves the macro-average by 4.8 points while using approximately 18% fewer tokens.

D.4 Repeated-Run Evaluation

We further assess run-to-run variation through three GPT-5-nano held-out evaluations on all six benchmarks.

As shown in Table 6, OrchDebate achieves higher mean accuracy than DynaDebate on all six benchmarks and S²-MAD on five, while using fewer tokens than both throughout. Its macro-average performance is 80.8% accuracy with 8.8k tokens, compared with 78.5%/19.0k for S²-MAD

Benchmark	Method	GPT-5-nano		Claude-Haiku-4.5		Qwen-3-8B	
		Acc(%)	Tok(k)	Acc(%)	Tok(k)	Acc(%)	Tok(k)
MuSR	MAD	57.7	16.0	63.7	10.8	65.3	16.9
	OrchDebate	73.3	12.6 (-21.3%)	75.7	10.2 (-5.6%)	73.0	9.4 (-44.4%)
	OrchDebate w/o Orchestrator	66.7	15.6 (-2.5%)	69.7	13.4 (+24.1%)	66.7	14.0 (-17.2%)
	OrchDebate w/o SDS	69.3	16.0 (0.0%)	72.0	14.0 (+29.6%)	68.7	14.8 (-12.4%)
	OrchDebate w/o Skills Bank	68.0	14.1 (-11.9%)	72.3	12.1 (+12.0%)	69.0	12.3 (-27.2%)
	OrchDebate w/o Self-Evolving	70.3	13.6 (-15.0%)	74.3	11.0 (+1.9%)	71.0	10.8 (-36.1%)
MMLU	MAD	80.2	8.1	84.5	3.4	80.2	9.0
	OrchDebate	87.3	8.1 (0.0%)	87.4	6.4 (+88.2%)	81.3	6.1 (-32.2%)
	OrchDebate w/o Orchestrator	84.4	9.5 (+17.3%)	84.8	7.6 (+123.5%)	79.2	8.2 (-8.9%)
	OrchDebate w/o SDS	85.8	11.0 (+35.8%)	85.6	8.4 (+147.1%)	79.9	9.1 (+1.1%)
	OrchDebate w/o Skills Bank	85.1	9.2 (+13.6%)	85.0	6.7 (+97.1%)	80.1	7.8 (-13.3%)
	OrchDebate w/o Self-Evolving	86.8	9.2 (+13.6%)	86.9	6.9 (+102.9%)	81.0	7.4 (-17.8%)

Table 4: Additional module ablation results on MuSR and MMLU across three LLM settings. We report accuracy and token cost, with token change in parentheses computed relative to MAD under the same model.

Method	GPQA		LogiQA2.0		MMLU		GSM8K		MATH		MuSR		Avg.	
	Acc(%)	Tok(k)	Acc(%)	Tok(k)	Acc(%)	Tok(k)	Acc(%)	Tok(k)	Acc(%)	Tok(k)	Acc(%)	Tok(k)	Acc(%)	Tok(k)
DynaDebate	59.4	17.1	70.5	11.0	81.8	9.3	94.3	11.6	78.5	11.0	69.0	17.2	75.6	12.9
DoWN	65.6	13.4	70.3	11.7	83.3	<u>6.9</u>	93.8	<u>3.2</u>	73.3	<u>6.9</u>	65.7	15.2	75.4	9.6
iMAD	65.6	11.7	72.7	10.2	84.5	10.5	92.8	8.4	76.0	8.5	66.3	15.9	76.3	10.9
OrchDebate	71.2	<u>9.6</u>	76.2	<u>8.8</u>	87.3	8.1	96.8	5.9	81.6	8.3	73.3	<u>12.6</u>	81.1	<u>8.9</u>

Table 5: Comparison with adaptive debate methods under GPT-5-nano. Avg. denotes the macro-average over six benchmarks. The highest accuracy is shown in **bold**, and the lowest token cost is underlined.

and 75.5%/12.9k for DynaDebate. Across benchmarks, OrchDebate’s standard deviations remain within 0.3–0.7 accuracy points and 0.1–0.3k tokens.

D.5 LLM Calls and Estimated Cost

To complement token counts, we report the number of LLM calls and estimated cost per query for the methods in the main comparison.

As shown in Table 7, relative to DynaDebate, OrchDebate uses 23–29% fewer calls and reduces estimated hosted-model cost per query by 35–37%. It also uses approximately one quarter of the calls required by S²-MAD. We report calls and token-derived cost rather than wall-clock latency because hosted APIs and local deployment are not directly comparable under network and queueing effects.

E Analysis of Online Skill Evolution

E.1 Candidate-Level Outcomes

We next summarize all candidate outcomes from the GPT-5-nano GPQA evolution stream.

Each candidate enters probation with $N(v) = 0$ and is evaluated only on subsequent executions. Of the 11 candidates, four remained below the minimum-evidence gate at the end of the stream. Among the seven that reached the gate, three were

promoted and four were retired; two retired candidates exceeded the degradation threshold. These results show that mutation does not imply replacement, since promotion requires sufficient future evidence and must satisfy both the advantage and degradation conditions.

E.2 Credit-Weight Sensitivity

We examine the sensitivity of the evolution process to the SDS-improvement weight α and token-cost weight λ .

For each setting in Table 9, we rerun the complete evolution process while fixing the initial bank, split and stream order, seed, prompts, mutation rules, and promotion gates. We vary one credit weight at a time, freeze the resulting bank, and evaluate it on the same 160 held-out GPQA examples. Across the $\pm 50\%$ settings, accuracy remains within two held-out examples of the default despite different promotion paths. Removing SDS credit lowers accuracy and increases harmful flips, while varying λ mainly changes token cost. All settings remain above the w/o Self-Evolving result of 66.9% accuracy with 11.5k tokens.

Benchmark	S ² -MAD		DynaDebate		OrchDebate	
	Acc(%)	Tok(k)	Acc(%)	Tok(k)	Acc(%)	Tok(k)
GPQA	65.4 ± 1.0	31.5 ± 0.3	60.4 ± 1.3	17.5 ± 0.4	71.2 ± 0.6	9.8 ± 0.1
LogiQA2.0	76.4 ± 0.5	17.1 ± 0.1	70.2 ± 0.7	11.2 ± 0.3	76.2 ± 0.4	9.0 ± 0.2
MMLU	85.1 ± 0.3	14.2 ± 0.2	81.3 ± 0.1	9.3 ± 0.3	87.3 ± 0.7	8.1 ± 0.3
GSM8K	94.4 ± 0.3	9.5 ± 0.1	94.3 ± 0.3	11.4 ± 0.1	96.6 ± 0.4	5.7 ± 0.2
MATH	80.8 ± 0.4	14.3 ± 0.1	78.1 ± 0.4	11.2 ± 0.4	81.0 ± 0.6	8.1 ± 0.1
MuSR	68.8 ± 0.5	27.4 ± 0.2	68.7 ± 1.0	17.1 ± 0.4	72.7 ± 0.3	12.3 ± 0.3

Table 6: Three-run held-out results under GPT-5-nano. We report mean ± sample standard deviation for accuracy and token cost. Across runs, we fix the examples, prompts, budgets, decoding settings, and evolved skill bank, while varying only the debate-agent seed.

Method	GPT-5-nano		Claude-Haiku-4.5		Qwen-3-8B	
	Calls	Cost	Calls	Cost	Calls	Cost
CoT	1.00	1.8	1.00	6.3	1.00	–
CoT-SC	20.00	17.2	20.00	52.7	20.00	–
MAD	4.08	2.8	3.17	16.0	4.23	–
S-MAD	30.00	2.1	30.00	53.3	30.00	–
S ² -MAD	22.00	3.1	22.00	24.9	22.00	–
DMAD	8.00	3.1	8.00	30.3	8.00	–
GroupDebate	22.00	3.9	22.00	37.9	22.00	–
DynaDebate	7.28	3.0	7.57	30.1	7.44	–
OrchDebate	5.51	1.9	5.35	19.6	5.72	–

Table 7: Average LLM calls and estimated cost per query, macro-averaged over six benchmarks. Cost is reported in mUSD and is derived from recorded input/output tokens and provider prices. Qwen-3-8B is locally deployed, so provider cost is not reported.

Generated	Below $n_{\min}$	Promoted	Retired
11	4	3	4

Table 8: Candidate outcomes from the GPT-5-nano GPQA evolution stream.

F Additional Comparison with Adaptive and Dynamic MAD

F.1 Unit, Interface, and Timescale of Adaptation

Table 10 positions OrchDebate against closely related adaptive and dynamic MAD methods along the object, interface, scope, and timescale of adaptation.

F.2 Relation to ACE and Selective-Debate Methods

ACE and OrchDebate share the premise that the next interaction should depend on the current dialogue rather than repeat one fixed act for a predetermined number of rounds. ACE analyzes current agent responses, selects atomic acts from a predefined library, composes them into a discussion action, and executes that action in the next round. OrchDebate instead exposes the next con-

trol step through a typed skill interface: a spawn skill jointly specifies role, visibility, and focus, while a finalize skill specifies the readout. The proposed action is additionally resolved against the legal set before execution. Thus, ACE provides within-episode dynamic discussion-action composition, whereas OrchDebate emphasizes legality-constrained whole-step skill selection and additionally updates reusable skill realizations across episodes.

DoWN and iMAD primarily adapt whether debate should be invoked, using initial confidence or structured self-critique signals, respectively. DynaDebate adapts reasoning-path initialization and invokes verification when disagreement persists. OrchDebate uses these kinds of mechanisms as possible coordination primitives, but places criticism, diversification, verification, and termination behind one SDS-conditioned control interface. Its second timescale then evaluates candidate realizations only on future executions before promotion or retirement. Together, these methods span complementary forms of adaptation; OrchDebate is distinguished by its unit of control, constrained interface, and online cross-episode skill refinement.

Credit Setting	Accuracy (%)	Tokens (k)	DS-correct $\rightarrow$ wrong	Promoted / Gate-reached
$\alpha = 0$	69.4 [111/160]	10.2	3.8% [6/160]	2/7
$0.5\alpha_0$	70.6 [113/160]	9.9	2.5% [4/160]	3/8
Default (α_0, λ_0)	71.2 [114/160]	9.6	2.5% [4/160]	3/7
$1.5\alpha_0$	70.6 [113/160]	9.8	1.9% [3/160]	4/8
$0.5\lambda_0$	71.9 [115/160]	10.5	3.1% [5/160]	4/9
$1.5\lambda_0$	70.0 [112/160]	8.9	1.9% [3/160]	2/7

Table 9: Credit-weight sensitivity under GPT-5-nano on GPQA. The main setting uses $\alpha_0 = 0.25$ and $\lambda_0 = 0.10$. “DS-correct $\rightarrow$ wrong” denotes harmful flips where direct solve is correct but the frozen evolved bank is wrong; “Gate-reached” denotes candidates with $N(v) \geq n_{\min}$.

Method	Primary adaptive object	Control signal / interface	Within-episode decision scope	Across-episode update
DoWN (Eo et al., 2025)	Debate invocation	Initial-answer confidence	Skip debate or run confidence-guided collaboration	None in evaluation
iMAD (Fan et al., 2026)	Debate invocation	Structured self-critique features and a learned classifier	Skip debate or invoke MAD	Offline-trained trigger; no online evaluation-time update
DynaDebate (Li et al., 2026)	Reasoning-path diversity and verification	Generated paths and disagreement	Allocate paths, conduct process-centric debate, and trigger verification	None reported
ACE (Gu et al., 2026)	Next-round discussion action	Current agent responses and dialogue context	Select atomic acts from ADAL, compose them, and execute the resulting action	None reported
OrchDebate	Entire next control step and its reusable realization	SDS, typed skills, and the current legal set	Select a spawn or finalize skill jointly specifying role, visibility, focus, or readout	Future-only probation, promotion, retirement, and bank freezing

Table 10: Conceptual comparison of representative adaptive and dynamic MAD methods. “Across-episode update” distinguishes online modification during the reported inference/evolution process from an offline-trained trigger.