DGTE: DISAGREEMENT-GATED TEST-TIME EVOLUTION OF ROLES AND TOPOLOGIES IN MULTI-AGENT SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model-driven multi-agent systems (MAS) use role specialization and communication to solve complex tasks, yet their test-time evolution typically relies on validation performance, task rewards, or executable feedback. For new tasks where task labels or trusted correctness scores are unavailable during generation, existing methods lack a reliable internal signal for deciding whether to continue evolving and how to modify the system. We therefore propose Disagreement-Gated Test-Time Evolution of Roles and Topologies (DGTE). For each task, DGTE starts from a frozen base system, runs multiple independent solver groups, and separately measures semantic disagreement among complete answers and operational disagreement caused by incomplete outputs. Their maximum forms a disagreement gate that determines whether to stop early, allocate another evolution round, or select a historical system state when the budget is exhausted. Execution traces, tool outputs, self-tests, and failure diagnostics guide bounded modifications to prompts, roles, helper agents, and communication topology. We validate DGTE on four benchmarks, including Humanity's Last Exam-Math, DeepResearchBench, BrowseComp-Plus, and Plancraft, and achieve SOTA performance on multiple benchmarks. The results show that DGTE achieves competitive performance across diverse task types; further analysis indicates that disagreement improves process completion, allocates additional computation, and helps repair incomplete outputs. DGTE has a modular design that can be seamlessly integrated into existing multi-agent methods without retraining the base model, improving system performance and process reliability with controlled overhead.

1 INTRODUCTION

Large language models (LLMs) are increasingly organized as multi-agent systems (MAS) in which specialized agents decompose tasks, use tools, inspect intermediate work, and exchange complementary lines of reasoning. The quality of such a system depends not only on its base model, but also on how its agents are instructed and connected. Prompts define local behavior; roles and communication edges determine how information and responsibility move through the system. Based on this observation, prior work has studied how to evolve prompts, graph structures, and other system components to improve performance across different tasks (Zhuge et al., 2024; Hu et al., 2025a; Zhang et al., 2025a; Zhou et al., 2026; Wang et al., 2026; Xu et al., 2026). The resulting systems are no longer fixed workflows. They are evolvable systems.

During evolution, an MAS must determine whether its current system warrants another evolution round. Existing system-evolution methods often support this decision with feedback such as validation performance, task reward, executable outcomes, trajectory assessment, or experience accumulated from previously scored tasks (Zhang et al., 2025a; Zhou et al., 2026; Lin et al., 2026; Xu et al., 2026; Zhang et al., 2026). Such signals are informative when available. They are not generally available for a new instance whose answer cannot be checked during generation. In open-ended analysis, research synthesis, and other tasks with sparse or delayed feedback, a system can produce several plausible answers without having an accessible gold answer or a trusted correctness score to guide its own evolution.

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.

This raises a question: how can we find an internal signal to guide the system's evolution? Disagreement is a natural candidate because it is observable before correctness is known. Agreement among sampled reasoning paths already supports answer selection (Wang et al., 2023; Chen et al., 2023), while semantic variation across free-form generations provides a label-free uncertainty signal (Kuhn et al., 2023). In multi-agent settings, related work uses consensus to stop discussion (Chen et al., 2024a), divergence to influence communication (Gou & Liu, 2026), and disagreements among models over entity boundaries and entity types to evolve instructions without gold annotations (Kim & Yoon, 2026). Taken together, these studies show that disagreement can support answer selection and stopping decisions, as well as the evolution of communication and instructions. In existing methods, however, these functions still serve separate decision stages and have not been unified to control the evolution of an MAS.

We therefore propose DGTE, which uses disagreement among multiple independent solver groups as the gate for system evolution. Every task begins from the same frozen base system. Several solver groups independently produce answers, and the controller measures both semantic incompatibility among complete answers and operational failure to produce usable answers. Their composite disagreement is the sole decision signal for allocating another evolution round, stopping early, and selecting a historical state when the evolution budget is exhausted. Within a round, the system may evolve its prompts, roles, helper agents, and the connections among them under explicit evolution constraints. Task information such as execution traces, tool outputs, and self-tests guides system evolution, while disagreement determines the number of evolution rounds.

This paper makes three contributions. First, it establishes a new result: disagreement can serve as an internal signal for guiding task-local MAS evolution when gold answers are unavailable. Second, it proposes DGTE, a framework that uses disagreement among solver groups to determine whether the system should continue evolving and how long evolution should continue, while using information generated during the current task-solving process to guide the evolution of prompts, roles, helper agents, and topology. Third, it conducts extensive experiments on DGTE across expert mathematics, long-form research, and retrieval-intensive question answering, validating the ability of disagreement gating to guide system evolution across different tasks and determine how many evolution rounds to perform.

2 Related Work

2.1 Fixed-at-Test-Time MAS Design

Early LLM-based MAS relied on manually specified roles and communication protocols, using fixed prompts, roles, and communication structures at test time. Subsequent work proposed automated design methods that replace part of this manual process with search or learning. GPTSwarm represents agents as computational graphs and evolves node prompts and graph connectivity using task objectives (Zhuge et al., 2024). ADAS uses a meta-agent to program candidate agentic systems in code, evaluates them on tasks, and retains successful discoveries to guide subsequent search (Hu et al., 2025a). AFlow applies Monte Carlo tree search to code-represented workflows and evolves their nodes, prompts, and control flow using execution feedback (Zhang et al., 2025a). MetaAgent constructs specialized agents and a finite-state-machine controller from a task description, then merges compatible agents and states without external training data (Zhang et al., 2025b). Mass interleaves local prompt evolution, topology search within a pruned design space, and global prompt evolution (Zhou et al., 2026). Skill-MAS uses multi-trajectory rollout and selective reflection to distill performance experience into a persistent Meta-Skill for generating future systems (Lin et al., 2026). Except for MetaAgent's task-description-driven construction, these optimization methods require scored tasks: they use performance results on development tasks to guide evolution and produce system designs intended for reuse (Zhuge et al., 2024; Hu et al., 2025a; Zhang et al., 2025a; Zhou et al., 2026; Lin et al., 2026).

2.2 Feedback-Driven Test-Time MAS Evolution

Unlike systems that remain fixed at test time, a separate line of work treats test-time solving of a single task instance as a dynamic process and uses information generated during the current solution process to guide MAS evolution at test time. Existing methods differ in the objects and scales of evolution.

MetaGen uses an Architect to generate and filter query-conditioned roles, initializes a constrained collaboration graph, and uses execution feedback to evolve low-utility role prompts and selected connections (Wang et al., 2026). TacoMAS uses trajectory-level contribution scores and meta-LLM diagnoses to evolve agent expertise every round, and periodically evolves agent membership and communication edges (Xu et al., 2026). EVOCHAMBER performs evolution at individual, team, and population scales: task failure or disagreement triggers collaborative reflection and asymmetric knowledge transfer, while reward and performance pressure guide team selection and population evolution (Zhang et al., 2026). These methods show that an MAS can adjust role capabilities, prompts, membership, or communication structures while solving a single task instance, thereby improving performance (Wang et al., 2026; Xu et al., 2026; Zhang et al., 2026). However, their evolution policies often rely on task feedback, rewards, performance assessments, or combinations of signals (Wang et al., 2026; Xu et al., 2026; Zhang et al., 2026). Therefore, when gold answers or external scores are unavailable at test time, how the system should evolve remains a problem that must be addressed.

2.3 Disagreement as a Gold-Free Control Signal

Discussion and debate methods organize agents into repeated exchanges of answers, explanations, and critiques, allowing each agent to adjust its answer state in response to others (Du et al., 2024; Smit et al., 2024). ReConcile repeats these exchanges until consensus or a round limit and then uses confidence-weighted voting for the team answer (Chen et al., 2024a). Semantic early stopping regulates writer–critic loops through semantic change between consecutive drafts, while adaptive stability detection estimates when further debate is unlikely to alter the result (Shrivastava, 2026; Hu et al., 2025b). These methods use agreement or stability to allocate discussion or computation rather than evolve the MAS itself.

Agreement and disagreement also provide gold-free signals outside deliberation. Self-consistency selects the most consistent sampled reasoning path, and Universal Self-Consistency extends this principle to free-form outputs (Wang et al., 2023; Chen et al., 2023). Semantic uncertainty groups generations by meaning, while cross-model disagreement has been used for correctness monitoring and routing (Kuhn et al., 2023; Gorbett & Jana, 2026). DiZiNER uses disagreements over entity boundaries and types to evolve annotation instructions without gold labels (Kim & Yoon, 2026). DySCo combines answer divergence with reliability, confidence, task relevance, and communication cost to select sparse edges, stopping when trust-weighted consensus stabilizes (Gou & Liu, 2026). These works assign disagreement to answer selection, uncertainty estimation, stopping, communication evolution, or instruction evolution, but these roles remain tied to separate decisions.

3 Method

3.1 Overview

Figure 1 presents a high-level overview of the DGTE framework. DGTE organizes agents into solver groups as the basic units of execution and comparison, while evolving the system as a whole at test time. For each task instance, DGTE starts from the same frozen base configuration, executes several solver groups, and measures disagreement among their answers. The evolution has two defining properties. First, it is *bounded*: fixed budgets limit both the number of evolution rounds and the configuration change permitted in each round and from the frozen base system. Second, it is *disagreement-driven*: a low disagreement value stops evolution, whereas a high value allocates another evolution round. Information produced while solving the task tells a controller how to evolve the system, but cannot override the disagreement gate.

The design combines solver groups with distinct problem-solving roles and label-free process diagnosis. Conflicting answers or incomplete outputs trigger bounded changes to prompts, roles, and communication structure; evolution stops early when the combined disagreement is at or below a preset threshold. This design aims to identify tasks requiring further processing and support task-local repair. Disagreement is a control signal for the solution process, not an estimate of answer correctness. Shared helpers and a common base model can still induce correlated errors across groups.

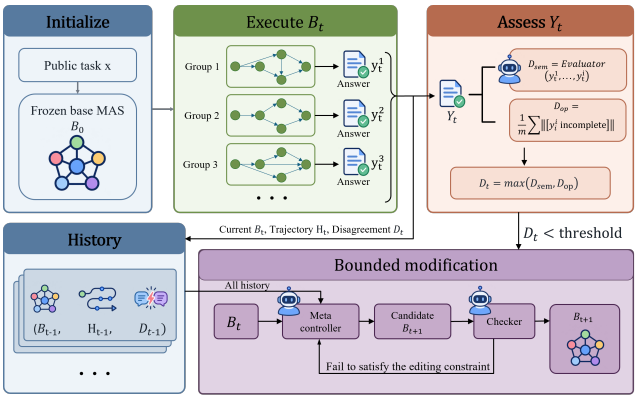

Figure 1: High-level conceptual overview of DGTE, showing multi-agent solution generation, answer checking, evolving communication topology, and disagreement-guided test-time updates.

3.2 Task-Local MAS State and Solver Groups

Let x denote a task instance and let $B_0 = (\mathcal{A}_0, \mathcal{E}_0, z)$ be a frozen base MAS. Here, $\mathcal{A}_0$ is a set of agents, $\mathcal{E}_0$ is a directed acyclic set of communication edges, and z is the task-specific answer schema. An agent $a \in \mathcal{A}_0$ has an identifier, a type (helper or solver), a role, and a prompt. Solver agents are terminal: they receive information from upstream helpers and synthesize these inputs into a final answer that conforms to the required schema. Accordingly, at evolution round t, the current system is $B_t = (\mathcal{A}_t, \mathcal{E}_t, z)$.

We organize multiple agents into a *solver group*. Each group contains one terminal solver, all of its helper ancestors, and the induced communication edges. Helpers may be shared across groups, but each group exposes exactly one complete answer through its terminal solver. This construction compares end-to-end solution processes rather than isolated agent messages. Additionally, each task begins evolution from a base system B_0, and all states B_t produced during the task are local to the current task x. Assume that there are m terminal solvers. Executing B_t produces group outputs $Y_t = \{y_t^1, \ldots, y_t^m\}$. Each output is marked as complete or incomplete; the latter includes length-limit, empty-content, invalid-JSON, and answer-schema failures.

3.3 Disagreement Gate

DGTE separates semantic incompatibility from failure to produce a usable answer. For the complete group outputs, we use a large language model (LLM) as a semantic evaluator. It jointly compares all final answers and, based on their semantic differences, returns a semantic disagreement score $D_{\text{sem}}(Y_t) \in [0, 1]$. If only one solver group produces a complete answer, we set $D_{\text{sem}}(Y_t) = 0$. A value of zero indicates that all complete answers are semantically equivalent, while larger values indicate increasingly incompatible answers.

To determine the representative answer for the current round, the evaluator clusters semantically equivalent answers and selects one solver-group output from the largest cluster. If multiple clusters have the same largest size, the cluster containing the earliest solver in the predefined order is preferred. If the stopping condition defined below is satisfied, this representative answer is returned as the final answer.

Additionally, incomplete groups carry distinct information: the system may be poorly organized even when the remaining complete answers agree. We therefore define operational disagreement as

$$D_{\text{op}}(Y_t) = \frac{1}{m} \sum_{i=1}^{m} \mathbb{I}[y_t^i \text{ is incomplete}], \tag{1}$$

where $\mathbb{I}[\cdot]$ is the indicator function. The gate value is

$$D_t = \max(D_{\text{sem}}(Y_t), D_{\text{op}}(Y_t)) \,. \tag{2}$$

Taking the maximum prevents agreement among a small surviving subset from hiding operational failure elsewhere in the system. The disagreement score determines when evolution stops. Before reaching the maximum number of evolution rounds K, the system stops early only when the disagreement score does not exceed τ, where τ is a fixed threshold. DGTE does not require the disagreement score to decrease monotonically. If this condition is not satisfied and $t < K$, evolution continues. If it is still not satisfied at $t = K$, the evolution budget is exhausted, evolution stops, and the procedure applies the historical-state selection rule below.

3.4 Task-Local System Update

The update operator acts on the complete task-local MAS configuration rather than on an isolated agent. Let $\mathcal{H}_t$ denote the history accumulated through round t, including the configuration, solver and helper outputs, execution traces, tool results, self-tests, failure diagnostics, label-free critiques, and the disagreement assessment. When $D_t > \tau$ and $t < K$, a meta-controller receives the public task x, the frozen base system B_0, the current system B_t, and $\mathcal{H}_t$, and returns a complete candidate configuration $\widetilde{B}_{t+1}$:

$$\widetilde{B}_{t+1} = \mathcal{U}(x, B_0, B_t, \mathcal{H}_t), \tag{3}$$

where $\mathcal{U}$ denotes the meta-controller's update operator.

The controller uses task-local evidence to determine where and how to modify the solution process. The candidate configuration may change an agent's prompt, role, or type; add or remove helper agents; and add or remove communication edges. The answer schema z remains fixed, and terminal solvers remain the only externally exposed outputs. Solver groups are not edited as independent objects. Instead, after a candidate configuration is constructed, each group is re-derived from one terminal solver, all of its upstream helper ancestors, and the induced communication edges. Consequently, modifying a shared helper or an edge can change the end-to-end computation of every solver group that contains that component.

Before application, the candidate must satisfy the structural validity conditions and the step and base-distance budgets defined below. In particular, the configuration must contain at least two terminal solvers, preserve an acyclic topology, and contain only valid communication edges. A validated candidate is applied as the next complete system B_{t+1}, after which the entire directed acyclic graph is executed again and the resulting evidence is appended to the task-local history. This update forms a closed loop: the current system is executed, semantic disagreement and operational failures are diagnosed, a bounded whole-system configuration is proposed, and the resulting system is re-executed. Task evidence determines the content of an update, whereas D_t determines whether another update is allocated and when evolution stops.

3.5 Constrained Test-Time Evolution and State Selection

To keep evolution bounded, define a discrete configuration distance

$$\begin{aligned} d(B, B') = {} & |\mathcal{A} \triangle \mathcal{A}'| + |\mathcal{E} \triangle \mathcal{E}'| \\ & + \sum_{a_i \in \mathcal{A} \cap \mathcal{A}'} \big(\mathbb{I}[k_i \neq k_i'] + \mathbb{I}[r_i \neq r_i'] + \mathbb{I}[p_i \neq p_i'] \big), \end{aligned} \tag{4}$$

where $\mathcal{A}$ and $\mathcal{A}'$ are the agent sets of B and B', respectively; k_i, r_i, and p_i are the type, role, and prompt of agent a_i; and $\triangle$ denotes symmetric difference. A candidate $\widetilde{B}_{t+1}$ is accepted only if

$$d(B_t, \widetilde{B}_{t+1}) \leq b_{\text{step}} \quad \text{and} \quad d(B_0, \widetilde{B}_{t+1}) \leq b_{\text{base}}. \tag{5}$$

The first bound limits abrupt round-to-round changes; the second prevents unbounded drift from the frozen starting system.

If the evolution budget is exhausted before any round satisfies the stopping condition, DGTE selects

$$t^* = \operatorname*{arg\,min}_{0 \leq t \leq K} (D_t, t), \tag{6}$$

using lexicographic order: it minimizes disagreement and breaks ties in favor of the earliest state. The final answer is the complete output of the representative solver selected by the disagreement evaluator at t^*. This prevents later evolution rounds from displacing an equally good earlier state merely because they consumed more computation. In summary, the DGTE procedure is given below.

DGTE procedure

1. Initialize B_0 and an empty history $\mathcal{H}$.
2. For $t = 0, \ldots, K$, execute all m solver groups in B_t to obtain $Y_t = \{y_t^1, \ldots, y_t^m\}$, and mark each output as complete or incomplete.
3. Use an LLM evaluator to jointly compare the complete outputs and return $D_{\text{sem}}(Y_t) \in [0, 1]$. If only one solver group produces a complete answer, set $D_{\text{sem}}(Y_t) = 0$. The evaluator also clusters semantically equivalent answers and selects a representative output from the largest cluster, breaking ties by the predefined solver order.
4. Compute operational disagreement directly from the incomplete-output ratio: $D_{\text{op}}(Y_t) = m^{-1} \sum_{i=1}^{m} \mathbb{I}[y_t^i \text{ is incomplete}]$. Set $D_t = \max(D_{\text{sem}}(Y_t), D_{\text{op}}(Y_t))$.
5. Append (B_t, Y_t, D_t) to $\mathcal{H}$. If $D_t \leq \tau$, return the representative answer and stop; otherwise, proceed to Step 6.
6. Use task-local evidence to propose B_{t+1}. Accept it only if $d(B_t, B_{t+1}) \leq b_{\text{step}}$ and $d(B_0, B_{t+1}) \leq b_{\text{base}}$. Otherwise, reject the candidate and set $B_{t+1} = B_t$.
7. Return the representative answer from the earliest state in $\mathcal{H}$ attaining the minimum disagreement.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Benchmarks and metrics. We evaluate four complementary benchmarks: Humanity's Last Exam-Math (HLE-Math) (Phan et al., 2025), DeepResearchBench (DRB) (Du et al., 2025), BrowseComp-Plus (BCP) (Chen et al., 2025), and Plancraft (Dagan et al., 2024). HLE-Math evaluates expert-level mathematical reasoning. Following Skill-MAS (Lin et al., 2026), we use 168 problems sampled from the original dataset as the evaluation set and report accuracy. DRB evaluates the ability to produce long-form research reports that synthesize information across diverse topics. We use its released 84-item test split and take the mean normalized score from Reference-based Adaptive Criteria-driven Evaluation (RACE) as the evaluation metric, covering comprehensiveness, insight, instruction following, and readability. BCP tests entity disambiguation, multi-hop retrieval, and evidence aggregation in retrieval-intensive question answering, for which we report accuracy. Plancraft evaluates multi-step planning, feasibility judgment, and interaction with an executable crafting environment over 580 tasks, including impossible goals. Following TacoMAS (Xu et al., 2026), we report the mean fraction of correctness and contradiction rubric items passed as the primary metric, with strict executable success reported separately.

Baselines. We compare against fixed-topology, offline-evolved, per-instance graph design, and within-instance evolution methods, as detailed in Section 4.2. Our local runs use DeepSeek-V4-Flash as the base model and `gemini-3-flash-preview` through the Gemini native API as the judge.

Implementation. DGTE uses `deepseek-v4-flash` at temperature 1.0, with no explicit client output-token cap. For each task, DGTE runs three solver groups in parallel and uses disagreement among their final answers to determine whether evolution should continue. We set the disagreement stopping threshold to $\tau = 0.1$ and allow at most $K = 10$ evolution updates. Under the configuration distance defined in Equation 4, each update may introduce at most three counted changes relative to the preceding system, while the resulting system may differ from the frozen initial system by at most six counted changes.

4.2 MAIN EXPERIMENT: CROSS-BENCHMARK COMPARISON

Drawing on the adaptation-time classification approach of **TacoMAS** (Xu et al., 2026), we divide the baselines into four categories. **Fixed-topology local probes** keep the agent roles and communication

Method	HLE Acc. (%)	DRB RACE (%)	BCP Acc. (%)	Plancraft rubric score (%)
Fixed-topology local probes				
MAS-Independent (Xu et al., 2026)	40.00	33.10	37.00	71.0
MAS-Decentralized (Xu et al., 2026)	30.00	28.33	44.00	60.0
MAS-Centralized (Xu et al., 2026)	38.00	0.96	40.00	56.0
MAS-Hybrid (Xu et al., 2026)	38.00	14.09	43.00	54.0
Offline-evolved				
MetaGPT (Hong et al., 2024)	7.78	41.19	17.26	65.61
AFlow (Zhang et al., 2025a)	21.43	49.93	15.48	69.87
Skill-MAS (Lin et al., 2026)	26.79	**51.69**	22.62	50.17
Per-instance graph design				
AgentVerse (Chen et al., 2024b)	10.71	43.75	17.86	67.52
MetaAgent (Zhang et al., 2025b)	33.33	46.14	24.40	66.31
MetaGen (Wang et al., 2026)	7.14	35.44	1.19	57.72
AOrchestra (Ruan et al., 2026)	19.64	41.03	18.45	77.85
Within-instance evolution				
SelfOrg (Tastan et al., 2026)	19.64	51.45	22.62	71.2
EvoAgent (Yuan et al., 2024)	9.52	49.10	16.67	67.6
DGTE (ours)	**47.02**	47.73	**50.60**	**85.60**

Table 1: Results on four benchmarks. HLE denotes Humanity's Last Exam, DRB denotes DeepResearchBench, BCP denotes BrowseComp-Plus, and RACE denotes Reference-based Adaptive Criteria-driven Evaluation. Some baseline results are taken from **Skill-MAS** (Lin et al., 2026).

graph fixed during inference and provide a controlled comparison of common multi-agent coordination patterns. **Offline-evolved** methods optimize an agent system before deployment and then reuse the resulting system across test instances. **Per-instance graph design** methods construct or optimize a separate agent graph for each task. **Within-instance evolution** methods modify the system during the solution of an individual task (Xu et al., 2026).

Table 1 reports the end-to-end performance of the complete systems. **DGTE** achieves state-of-the-art (SOTA) results on HLE-Math, BCP, and Plancraft. The experiments demonstrate that the end-to-end results stem from the combination of multiple solution paths and task-local process diagnosis. Distinct solver roles can expose different candidate answers, while semantic conflicts and incomplete outputs provide signals for further processing without access to task labels. Bounded changes to roles and information flow help groups produce usable outputs.

Meanwhile, on DRB, **DGTE** scores below **Skill-MAS** on long-form research reports.[1] This indicates that agreement on a conclusion does not mean that shared omissions or shallow analysis have been resolved; report quality depends on coverage and insight as well as consistency. Therefore, in open-ended synthesis tasks, answer consistency alone does not adequately represent report quality; by comparison, answer disagreement is better suited as a signal for further processing in tasks with clearly defined answers.

4.3 Component Ablations

To determine whether DGTE's performance arises from the stopping gate itself and to distinguish the roles of semantic disagreement, operational disagreement, and system evolution, we conduct component ablations. The experiment asks whether using semantic and operational disagreement jointly provides a more complete process signal, and whether the frozen base system can achieve comparable performance without test-time evolution.

The full DGTE uses semantic disagreement among complete solver-group answers and the fraction of incomplete solver groups, taking their maximum as the stopping-gate input. The semantic-only variant removes operational disagreement and makes decisions from semantic differences among complete answers. The operational-only variant removes semantic comparison and makes decisions from the fraction of incomplete solver groups. The no-evolution variant executes the frozen base system once. All variants use the same solver groups, answer-selection rule, worker model, temperature, and computation budgets, so the comparison focuses on the stopping-gate signals and the evolution process. Table 2 reports the results on the HLE-Math test set. Full DGTE achieves the highest

[1]The official DRB scorer issued five criterion-text matching fallbacks and used its average-weight fallback; we retain the resulting score.

Variant	Accuracy (%)
DGTE (full gate)	**47.02**
Semantic-only	45.31
Operational-only	44.81
No-evolution	43.17

Table 2: HLE-Math gate ablation results.

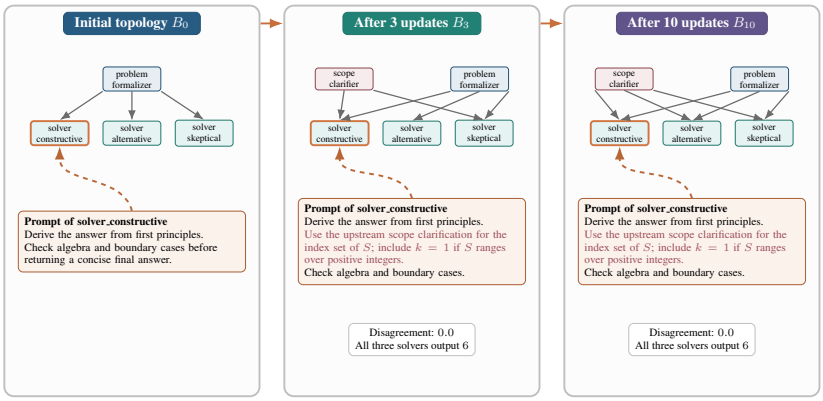

Figure 2: Illustrative topology and prompt evolution from an HLE-Math validation trace. The three panels show the initial system B_0, the system after three updates B_3, and the system after ten updates B_{10}. The highlighted `solver_constructive` node and its prompt card show the addition of `scope_clarifier`, the corresponding prompt update, and the subsequent expansion of its connection range.

accuracy, followed by semantic-only, operational-only, and no-evolution. The mechanism is that semantic and operational disagreement capture two different failure modes. Semantic disagreement identifies substantive conflicts among complete answers, whereas operational disagreement identifies whether solver groups successfully produced usable outputs. Taking their maximum as the stopping-gate input allows DGTE to allocate further evolution whenever either type of problem occurs. The two signals are therefore complementary by design.

Figure 2 illustrates a corresponding task-local evolution trace, in which both the communication topology and a solver prompt change across successive system states.

4.4 Adaptive Computation and Cost–Quality Trade-offs

To test whether DGTE's adaptive stopping mechanism can allocate test-time evolution computation according to disagreement signals observed during task execution, and to distinguish gains from adaptive decisions from gains obtained simply by running more evolution steps, we conduct an adaptive-computation and cost–quality trade-off experiment. The experiment also evaluates whether adaptive stopping can avoid unnecessary language-model calls and token consumption while maintaining comparable task performance, and whether additional evolution budget continues to yield corresponding performance gains. Specifically, we compare adaptive stopping with fixed maximum evolution counts $K \in \{0, 3, 10\}$. Here K counts system evolution updates after the initial execution of B_0. Fixed variants use the same controller, edit constraints, and historical-state selection. For the larger-K setting, even when disagreement has already decreased, execution continues until the prescribed number of evolution steps is reached.

Figure 3 places the four DGTE computation settings and the other multi-agent baselines in the same cost–accuracy space. All methods' total inference costs are converted to U.S. dollars. The horizontal axis shows mean inference cost per task on a logarithmic scale, and the vertical axis shows task accuracy. The fixed-evolution settings provide a reference for the effect of computation budget. Increasing the fixed evolution count improves accuracy, but requires progressively higher inference cost. The additional computation therefore yields performance gains at a growing cost. The adaptive setting lies between the low-budget and high-budget fixed settings, indicating that DGTE can avoid

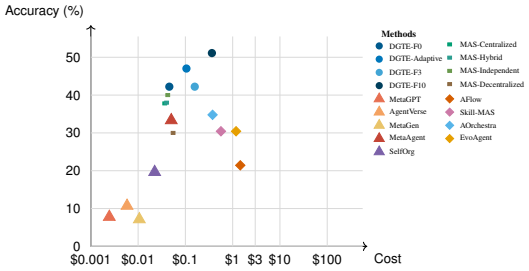

Figure 3: Cost–accuracy comparison for four DGTE computation settings and multi-agent baselines. The horizontal axis shows mean inference cost per task in U.S. dollars on a logarithmic scale.

unconditionally executing the maximum number of evolution updates while retaining a competitive accuracy level.

Compared with the baselines, DGTE occupies a favorable region of the cost–accuracy space. Low-cost fixed-topology baselines remain below DGTE in accuracy, whereas the more expensive offline-evolved and per-instance methods do not achieve higher accuracy at comparable cost. Taken together, the figure shows that disagreement-gated evolution provides a controllable way to trade additional computation for performance, allowing DGTE to maintain a balance between task accuracy and total inference cost.

4.5 Limitations and Future Work

DGTE relies on differences among solver groups to identify tasks requiring further processing, but distinct roles do not guarantee independent reasoning errors. Groups that share a base model, helper information, or reasoning assumptions remain susceptible to common biases, and the disagreement gate cannot detect errors that all groups consistently accept. Moreover, semantic disagreement is assessed by a model, whose evaluation stability also affects stopping decisions and answer selection. Future work could compare different combinations of base models, information-sharing arrangements among solver groups, and semantic evaluators to examine how these factors affect the reliability of the disagreement signal.

DGTE uses a preset stopping threshold and evolution constraints to control computation, but these choices also restrict the range of system adjustments that can be explored. Tasks differ in the depth of reasoning and organizational structure they require, so a single parameter setting need not suit every task. When effective solving requires substantial changes to the initial structure, bounded modifications around the base system also limit exploration. Future work could use parameter sensitivity experiments to characterize the applicability of these settings and investigate dynamic allocation of evolution budgets based on task characteristics and information observed during execution.

Finally, DGTE evolves only within individual tasks, and each new task starts from the frozen base system. It therefore does not yet accumulate effective role assignments and collaboration structures from previous tasks as reusable experience, requiring repeated exploration when solving similar tasks. Future work could combine within-task evolution with experience accumulated across tasks, distill reusable structural adaptation strategies on development tasks, and use independent tests to evaluate their transfer and their effect on adaptation to new tasks.

5 Conclusion

This paper introduces DGTE, a disagreement-gated test-time evolution framework that adapts multi-agent roles and communication topology per task without task labels or trusted correctness feedback. Semantic disagreement among complete solver groups and operational failures among incomplete groups determine whether to continue evolution, while traces and diagnostics guide bounded updates. Across HLE-Math, DeepResearchBench, BrowseComp-Plus, and Plancraft, DGTE achieves the strongest listed results on three benchmarks and competitive performance on the fourth.

REFERENCES

Justin Chih-Yao Chen, Swarnadeep Saha, and Mohit Bansal. ReConcile: Round-table conference improves reasoning via consensus among diverse llms. *arXiv preprint arXiv:2309.13007*, 2024a. URL https://arxiv.org/abs/2309.13007.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chen Qian, Chi-Min Chan, Yujia Qin, Yaxi Lu, Ruobing Xie, et al. AgentVerse: Facilitating multi-agent collaboration and exploring emergent behaviors in agents. *arXiv preprint arXiv:2308.10848*, 2024b. URL https://arxiv.org/abs/2308.10848.

Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. Universal self-consistency for large language model generation. *arXiv preprint arXiv:2311.17311*, 2023. URL https://arxiv.org/abs/2311.17311.

Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, Mingyi Su, et al. Browsecomp-plus: A more fair and transparent evaluation benchmark of deep-research agents. *arXiv preprint arXiv:2508.06600*, 2025. URL https://arxiv.org/abs/2508.06600.

Gideon Dagan, Felix Keller, and Anna Lascarides. Plancraft: An evaluation dataset for planning with llm agents. *arXiv preprint arXiv:2412.21033*, 2024. URL https://arxiv.org/abs/2412.21033.

Mingxuan Du, Benfeng Xu, Chiwei Zhu, Xiaorui Wang, and Zhendong Mao. Deepresearchbench: A comprehensive benchmark for deep research agents. *arXiv preprint arXiv:2506.11763*, 2025. URL https://arxiv.org/abs/2506.11763.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024. URL https://arxiv.org/abs/2305.14325.

Matt Gorbett and Suman Jana. Cross-model disagreement as a label-free correctness signal. *arXiv preprint arXiv:2603.25450*, 2026. URL https://arxiv.org/abs/2603.25450.

Wanshuang Gou and Zihan Liu. Dynamic trust-aware sparse communication topology for llm-based multi-agent consensus. *arXiv preprint arXiv:2606.01828*, 2026. URL https://arxiv.org/abs/2606.01828. Dynamic sparse edge selection uses reliability, confidence, answer divergence, task relevance, and cost; consensus stability controls early stopping.

Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*, 2024. URL https://arxiv.org/abs/2308.00352.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *International Conference on Learning Representations (ICLR)*, 2025a. URL https://arxiv.org/abs/2408.08435.

Tianyu Hu, Zhen Tan, Song Wang, Huaizhi Qu, and Tianlong Chen. Multi-agent debate for llm judges with adaptive stability detection. *arXiv preprint arXiv:2510.12697*, 2025b. URL https://arxiv.org/abs/2510.12697.

Siun Kim and Hyung-Jin Yoon. DiZiNER: Disagreement-guided instruction refinement via pilot annotation simulation for zero-shot named entity recognition. *arXiv preprint arXiv:2604.15866*, 2026. URL https://arxiv.org/abs/2604.15866.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://arxiv.org/abs/2302.09664.

Hehai Lin, Qi Yang, and Chengwei Qin. Skill-MAS: Evolving meta-skill for automatic multi-agent systems. *arXiv preprint arXiv:2606.18837*, 2026. URL https://arxiv.org/abs/2606.18837.

Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, et al. Humanity's last exam. *arXiv preprint arXiv:2501.14249*, 2025. URL https://arxiv.org/abs/2501.14249.

Jianhao Ruan, Zhihao Xu, Yiran Peng, Fashen Ren, Zhaoyang Yu, Xinbing Liang, Jinyu Xiang, Bang Liu, Chenglin Wu, Yuyu Luo, and Jiayi Zhang. AOrchestra: Automating sub-agent creation for agentic orchestration. *arXiv preprint arXiv:2602.03786*, 2026. URL https://arxiv.org/abs/2602.03786.

Sahil Shrivastava. Semantic early-stopping for iterative llm agent loops: A judge-efficient study of when to halt. *arXiv preprint arXiv:2606.27009*, 2026. URL https://arxiv.org/abs/2606.27009.

Andries Smit, Nathan Grinsztajn, Paul Duckworth, Thomas D. Barrett, and Arnu Pretorius. Should we be going mad? a look at multi-agent debate strategies for llms. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024. URL https://arxiv.org/abs/2311.17371.

Nurbek Tastan, Samuel Horváth, and Karthik Nandakumar. Stochastic self-organization in multi-agent systems. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://openreview.net/forum?id=rS3Jb9AAej.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://arxiv.org/abs/2203.11171.

Yimeng Wang, Jiaxing Zhao, Hongbin Xie, Hexing Ma, Yuzhen Lei, Shuangxue Liu, Xuan Song, Zichen Zhang, and Haoran Zhang. Metagen: Self-evolving roles and topologies for multi-agent llm reasoning. *arXiv preprint arXiv:2601.19290*, 2026. URL https://arxiv.org/abs/2601.19290. Local arXiv v1 metadata verified. MetaGen revises query-conditioned roles and topology during inference using lightweight feedback signals.

Chen Xu, Yicheng Hu, Ruizi Wang, Xinyu Lin, Wenjie Wang, Dongrui Liu, and Fuli Feng. Tacomas: Test-time co-evolution of topology and capability in llm-based multi-agent systems. *arXiv preprint arXiv:2605.09539*, 2026. URL https://arxiv.org/abs/2605.09539. Critical concurrent prior. Two-time-scale co-evolution with meta-judge and replicator dynamics.

Siyu Yuan, Kaitao Song, Jiangjie Chen, Xu Tan, Dongsheng Li, and Deqing Yang. Evoagent: Towards automatic multi-agent generation via evolutionary algorithms. *arXiv preprint arXiv:2406.14228*, 2024. URL https://arxiv.org/abs/2406.14228.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xiong-Hui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating agentic workflow generation. In *International Conference on Learning Representations (ICLR)*, 2025a. URL https://arxiv.org/abs/2410.10762.

Yaolun Zhang, Xiaogeng Liu, and Chaowei Xiao. MetaAgent: Automatically constructing multi-agent systems based on finite state machines. In *International Conference on Machine Learning (ICML)*, 2025b. URL https://arxiv.org/abs/2507.22606. Author and venue metadata verified against arXiv v1.

Yaolun Zhang, Tianyi Xu, Shengyu Dai, Zhenwen Shao, Qingyun Wu, and Huazheng Wang. EVOCHAMBER: Test-time co-evolution of multi-agent system at individual, team, and population scales. *arXiv preprint arXiv:2605.11136*, 2026. URL https://arxiv.org/abs/2605.11136. Test-time evolution at individual, team, and population scales. Failure or disagreement triggers collaborative reflection, while reward and performance pressure govern persistent evolution.

Han Zhou, Xingchen Wan, Ruoxi Sun, Hamid Palangi, Shariq Iqbal, Ivan Vulić, Anna Korhonen, and Sercan O. Arik. Multi-agent design: Optimizing agents with better prompts and topologies. *International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2502.02533. Published at ICLR 2026. Mass interleaves local prompt, topology, and global prompt optimization using task evaluation.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. GPTSwarm: Language agents as optimizable graphs. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024. URL https://arxiv.org/abs/2402.16823.