Review

Not peer-reviewed version

LLM-Based Multi-Agent Systems for Mathematical Problem Solving: A Comprehensive Literature Review

Bektur Toktobekov * and Burul Shambetova

Posted Date: 12 December 2025

doi: 10.20944/preprints202512.1105.v1

Keywords: large language models; multi-agent systems; mathematical reasoning; multi-agent reinforcement learning; chain-of-thought; agent collaboration; hierarchical architectures

 doi:10.20944/preprints202512.1105.v1

Review

LLM-Based Multi-Agent Systems for Mathematical Problem Solving: A Comprehensive Literature Review

Bektur Toktobekov * and Burul Shambetova

Faculty of Engineering and Computer Science, Ala-Too International University

* Correspondence: bektur.toktobekov@alatoo.edu.kg

Abstract

While large language models (LLM) have demonstrated significant advances in natural language processing, complex mathematical reasoning remains a challenging task, often revealing their limitations in multi-stage calculations and logical consistency. Multi-agent systems have become a promising paradigm for overcoming these limitations by distributing cognitive tasks between interacting agents, reflecting the dynamics of human problem solving. This paper provides a comparative review of the literature on nineteen different multi-agent architectures for solving mathematical problems. Our main research question is: "How do various LLM-based multi-agent architectures enable or improve mathematical problems, and what are their comparative advantages, limitations, and design trade-offs?" Through a systematic analysis of the roles of agents, interaction mechanisms, and training methods, we have identified several key findings. We observe the evolution of architecture from unstructured debate-based systems to more efficient hierarchical and self-optimizing frameworks. We highlight persistent problems that hinder progress, including agent homogeneity, when agents working on the same LLM cannot generate truly diverse reasoning, and the problem of "lazy agents", when some agents contribute minimal to consistent collaboration. This review contributes to a structured understanding of the current situation and lays the foundation for future research aimed at developing more reliable, efficient, and complex multi-agent reasoning systems.

Keywords: large language models; multi-agent systems; mathematical reasoning; multi-agent reinforcement learning; chain-of-thought; agent collaboration; hierarchical architectures

1. Introduction

Mathematical thinking is the cornerstone of human intelligence and serves as the most important benchmark for the creation of artificial general intelligence. Although modern Large Language Models (LLM) have achieved significant success in solving various linguistic problems, their proficiency in solving complex mathematical problems remains inconsistent. Single-LLM architectures, despite their scale, are often prone to logical errors, computational inaccuracies and error propagation in multi-stage reasoning chains. These inherent limitations highlight the strategic importance of exploring alternative computational paradigms that can complement and structure the reasoning process.

Multi-agent systems represent a powerful and increasingly popular approach to improving the reasoning abilities of LLMs. Drawing an analogy to human collaboration, where teams of experts with different skills work together to solve complex tasks, these systems break down a complex task into manageable subtasks. By combining a group of specialized or heterogeneous agents, multi-agent architectures can increase accuracy, reliability, and even creativity. Agents can be assigned specific roles, such as generating solutions, verifying steps, providing critical feedback or planning a strategy, which allows for a structured workflow that eliminates the weaknesses of the monolithic model.

 doi:10.20944/preprints202512.1105.v1

The main purpose of this paper is a systematic review and comparison of nineteen multi-agent architectures for mathematical thinking. This comparative analysis is based on the main research question: How do various LLM-based multi-agent architectures enable or improve mathematical problems, and what are their comparative advantages, limitations, and design trade-offs?

2. Methodology

This section describes in detail the methodological approach used in conducting a comparative literature review, providing a systematic, transparent and reliable analysis of selected multi-agent architectures. A clear methodology is needed to determine the validity of our comparisons and the reliability of our conclusions.

2.1. Inclusion Criteria

The purpose of this review is to conduct a fundamental comparative analysis of LLM-based Multi-Agent System (MAS) architectures to solve mathematical problems. To achieve this goal, we have established systematic selection criteria, paying special attention to works that solve unique problems at the intersection of complex thinking and collaborative artificial intelligence systems.

Primary inclusion criteria:

1. Multi-agent LLM systems for justification: We have chosen platforms where multiple LLMs work together or competitively, explicitly excluding single-agent systems unless they are the basis for making fundamental architectural decisions.
2. Focus on math tests: Inclusion was limited to studies demonstrating the empirical application of mathematical problems datasets, in particular competitive-level problems such as MATH and complex arithmetic reasoning problems such as GSM8K. This focus is crucial because mathematical problems require strict logical consistency and precision in the execution of a sequence of actions.
3. Architectural and mechanistic transparency: We have prioritized papers that provide clear information about the architecture of the system, including the roles of agents (e.g. Planner, Verifier), topology governing interaction (fixed or dynamic), communication protocols and strategies for obtaining a final answer (e.g. voting, consensus, judging).

The architectures under review are:

- AgenticMath
- MALT
- MARS-PO
- MARS (Socratic Guidance)
- MARS (Efficient Collaboration)
- FMAD
- DiMo
- SIER
- ECON
- LbMAS
- MA ToT
- Dr. MAMR
- ReMA
- ANN
- ILR
- MathChat
- MACM
- MLPO
- AdCo

 doi:10.20944/preprints202512.1105.v1

2.2. Analytical Framework

We used a consistent analytical framework to deconstruct and analyze each architecture. This framework included extracting key characteristics for each system, including:

- Agent Roles: Specific functions assigned to each agent in the system (e.g. Generator, Verifier, Critic).
- Paradigms of Interaction: The main mode of communication and cooperation between agents (e.g. debates, hierarchical review, evolutionary selection).
- Orchestration Mechanisms: A method used to manage the workflow and sequence of actions of agents (e.g. sequential pipeline, coordination based on game theory).
- Training Methodologies: Methods used to train or fine-tune agents according to their roles (e.g. Supervised Fine-Tuning (SFT), Direct Preference Optimization (DPO), Multi-Agent Reinforcement Learning (MARL)).
- Reported Performance: Empirical results published by the authors on standard mathematical benchmarks.

To streamline the synthesis of information across 19 architectures, we utilized Google NotebookLM as an organizational aid. However, all analytical judgments, comparisons, and critical assessments were conducted independently by the authors through direct examination of the primary sources.

2.3. Challenges in Comparative Analysis

Performing a direct, one-to-one comparison of these architectures is inherently challenging due to a lack of standardization across studies. The primary confusing variables include:

- Different Base LLMs: Architectures are implemented and tested on a wide range of foundation models, from the GPT series to the Llama, Qwen, and Mixtral families, each with different baseline capabilities.
- Non-Standardized Computational Budgets: Studies often do not report or control for the total computational cost (e.g., token consumption, inference time), making it difficult to compare the efficiency of different methods.
- Variations in Implementation: Minor differences in prompting, sampling strategies (e.g., temperature) and the number of agents can significantly impact performance.

3. Overview of Architectures

The research landscape on leveraging Large Language Models (LLMs) for complex mathematical problem-solving is rich with specialized multi-agent system (MAS) designs. These architectures move beyond traditional single-agent approaches like Chain-of-Thought (CoT) by introducing collaboration, structured deliberation, optimization and rigorous verification mechanisms.

Here is an introduction to the surveyed LLM multi-agent architectures that target reasoning and mathematical problem solving:

1. AgenticMath: This platform is a new automated multi-agent pipeline designed specifically to generate high-quality mathematical question and answer pairs to enhance LLM SFT. It provides quality control at every stage, aiming to overcome the limitations associated with poor-quality or poorly formulated synthetic mathematical data. The system operates in four sequential stages: (1) Seed Question Filter (selecting the most meaningful, complex and understandable questions); (2) Agentic Question Rephrase (the multi-agent system generates diverse, logically consistent paraphrases); (3) Answer Augment (the solver agent creates complete CoT reasoning chains); (4) Quality Assessment (summing multidimensional scores to save only the best pairs). By focusing on the targeted generation of high-quality data, AgenticMath demonstrates an improved mathematical thinking ability in fine-tuned LLMs.

 doi:10.20944/preprints202512.1105.v1

2. MALT (Multi-agent LLM Learning): MALT is a new post-training strategy that expands the possibilities of logical thinking by breaking the decision-making process into three sequential stages performed by heterogeneous agents: generation, verification and refinement. During data generation, agents are repeatedly selected to build a multi-agent search tree. It is important to note that MALT uses an off-policy approach combined with value iteration to propagate reward signals back to role-conditioned models, allowing each agent to specialize, learning from right and wrong reasoning trajectories automatically, without human control. Such specialization in a decentralized system is aimed at optimizing the usually large spaces of joint actions of LLM agents. MALT has achieved significant relative performance improvements in math tests such as MATH and GSM8K.
3. MARS-PO (Multi-Agent Reasoning System Preference Optimization): This architecture works in the context of preference optimization (PO), focusing on mathematical analysis problems where the answer consists of Chain-of-Thought reasoning steps and a final answer. The main function of this multi-agent system is the coordinated generation of diverse and representative response samples that form high-quality preference pairs necessary for subsequent preference optimization.
4. MARS (Multi-Agent Framework Incorporating Socratic Guidance): This platform is specifically designed for Automated Prompt Optimization (APO). To eliminate the limitations associated with fixed patterns and inefficient search in prompt spaces, MARS uses a centralized “Planner” agent to autonomously plan unique optimization paths for various tasks. Its key mechanism is the Socratic dialogue-guidance (‘Teacher-Critic-Student’), which iteratively refines the prompt. There are seven agents involved in the system: “Manager”, “User Proxy”, “Planner”, “Teacher”, “Critic”, “Student” (optimizes the prompt) and “Target” (checks the optimized prompt). It has demonstrated excellence in specialized areas that require logical thinking, such as the GSM8K mathematical problems.
5. MARS (Toward More Efficient Multi-Agent Collaboration For LLM Reasoning): Known as the Multi-Agent Review System (MARS), this architecture is a role-based collaboration framework inspired by the academic review process. Its main trade-off is efficiency, as it is designed to reduce the computational costs associated with traditional Multi-Agent Debates (MAD) by avoiding reviewer-to-reviewer interactions. The system includes an “Author” agent (generates an initial response using CoT), several independent “Reviewer” agents (evaluates the response and provides solutions, comments, and confidence scores), and a centralized “Meta-Reviewer” agent (collects feedback, resolves conflicts, and directs revision).
6. FMAD (Fine-grained Multi-Agent Debate): FMAD is an automated annotation framework focused on building highly validated reasoning steps, eliminating the need for expensive human annotations. It uses a debate framework inspired by MAD to test the correctness of individual steps of reasoning. The participants include “Debater A” (claims that the step is correct/useful), “Debater B” (claims that the step is incorrect/useless) and “Judge” (determines the winner by the number of confidence points). This mechanism was used to create “MMATH-Data” that enriches datasets such as GSM8K and MATH by annotating each step with a history of debates and judge decisions.
7. DiMo (Multi-Agent Collaboration Platform for Diverse Thinking Modes): DiMo enhances performance and interpretability of logical thinking tasks through structured debate that simulate two complementary cognitive protocols: Divergent Thinking Mode and Logical Thinking Mode. The Logical Thinking Mode, which is useful for solving mathematical problems, focuses on step-by-step verification and local refinement. The framework uses specialized LLM agents (e.g. Generator, Evaluator) that iteratively think over solutions and generate explicit, verifiable chains of reasoning. DiMo has achieved significant success in complex mathematical benchmarks such as GSM-hard.
8. SIER (Swarm Intelligence Enhanced Reasoning): This framework introduces a new agent-based Swarm Intelligence (ASI) paradigm, conceptualizing LLM reasoning as an optimization problem

 doi:10.20944/preprints202512.1105.v1

in which agents collaboratively search for a global optimum in the solution space. SIER uses a density-driven strategy, including Kernel Density Estimation (KDE) andNon-Dominated Sorting, to jointly optimize the quality and diversity of solutions at a step-by-step level, which prevents agents from approaching local optima. The multi-criteria selection of reasoning steps allows population of find high-quality and diverse solutions, demonstrating high efficiency in complex mathematical reasoning benchmarks.

9. ECON (Effective Coordination via Nash Equilibrium): ECON is a hierarchical reinforcement learning paradigm that models multi-LLM coordination as a game with incomplete information aimed at achieving Bayesian Nash Equilibrium (BNE). It operates in accordance with the "Coordinator-Executor" architecture, in which several Execution LLMs operate independently and locally based on probabilistic beliefs, guided by the Coordinator LLM. This belief-based coordination mechanism avoids the high computational costs associated with direct inter-agent interaction, making it scalable. It has been empirically proven that ECON is superior to existing single-agent and multi-agent approaches in solving complex reasoning and planning tasks, including MATH.
10. LbMAS (Blackboard-based LLM Multi-Agent System): This architecture implements the traditional blackboard architecture in LLM-based systems to solve complex, poorly structured tasks where fixed workflows are unavailable. LbMAS has a central "Blackboard" (public and private sections), which serves as the only communication channel and shared memory for all agents. The "Control Unit" (LLM agent) dynamically selects specialized agents (including planner, decider, critic, cleaner, and conflict-resolver) iteratively based on the current blackboard content. This scheme allows system to adapt the collaboration mechanism in a timely manner.
11. MA ToT (Multi-Agent Tree-of-Thought): This approach integrates a Tree of Thought (ToT) strategy into a multi-agent environment. The main mechanism includes Reasoner agents that build a tree structure of thoughts (reasoning stages), allowing parallel exploration of various reasoning paths, which is an improvement over the linear CoT path. This multi-agent structure allows for dynamic path evaluation. The main goal is to purposefully solve problems through a variety of thought exploration.
12. ReMA (Reinforced Meta-thinking Agents): ReMA uses MARL to explicitly encourage meta-thinking in LLMs, effectively separating the high-level strategic process ("thinking about thinking") from low-level execution. The architecture consists of a high-level meta-thinking agent (generates strategic controls/plans) and a low-level reasoning agent (performs detailed execution). This separation distributes the research space, allowing agents to explore more structurally and effectively during training. ReMA adapts RL algorithms and reward functions to align goals, demonstrating superior performance in complex mathematical analysis tasks, especially in out-of-distribution (OOD) benchmarks.
13. The Doctor. MAMR (Multi-Agent Meta-Reasoning Done Right): Dr. MAMR is a multi-agent framework specifically designed to solve the critical problem of "lazy agent" behavior, when one agent dominates and the other makes minimal contributions, which can lead to the transformation of multi—turn meta-reasoning systems into an inefficient single agent system. As in ReMA, it uses a hierarchical structure with a high-level meta-thinking agent and a low-level reasoning agent. However, Dr. MAMRE introduces a stable measurement of cause and effect and a verifiable reward mechanism that encourages deliberation. This mechanism allows the reasoning agent to discard noisy responses, consolidate instructions, and resume its reasoning if necessary, preventing short cuts and contributing to a balanced contribution.
14. ANN (Agentic Neural Network): ANN considers multi-agent collaboration as a layered neural network architecture, where agents are nodes and layers are collaborative teams performing specific subtasks. ANN uses a new two-phase optimization: the Forward Phase (dynamic task decomposition and team selection) and the Backward Phase (textual feedback backpropagation). Critical text analysis acts like gradient signals, guiding agents to develop their roles, prompts,

 doi:10.20944/preprints202512.1105.v1

and coordination independently. This neuro-symbolic approach aims at full automation, eliminating the need for manual prompt engineering and providing superior performance when performing tasks, including mathematical reasoning (MATH).

15. ILR (Interactive Learning for LLM Reasoning): ILR is a collaborative learning framework aimed at enhancing the individual ability to solve problems through dynamic multi-agent interaction during training. The framework includes Dynamic Interaction (where models exchange ideas, challenge reasoning and provide feedback) and Perception Calibration (which uses an automated method to integrate the characteristics of one LLM's reward data distribution into the reward function of his colleagues, providing continuous, fine-grained incentive signals). This process is based on discussions with human colleagues to help LLMs overcome individual blind spots and develop stronger reasoning abilities.
16. MathChat: Is an zero-shot conversational framework specifically designed for solving complex mathematical problems through the interaction of an LLM agent (for example, GPT-4) and a supporting User Proxy Agent. The user proxy agent manages the interaction and operation of the tool by recognizing code blocks as requests and returning either correct execution results or error messages, allowing the LLM agent to dynamically adjust its strategy or correct errors. This design aggregates prompting methods and is very effective for solving complex problems based on datasets such as MATH benchmark.
17. MACM (Multi-Agent System for Conditional Mining): MACM is a multi-agent prompting method designed to solve complex mathematical problems by continuously extracting and expanding known conditions relative to the overall objective. This structure deliberately departs from the hierarchical relationships found in methods such as ToT. The system uses a three-role team: a "Thinker" (to generate ideas), a "Judge" (to verify and make decisions), and a "Executor" (for computation), which together minimize potential errors in reasoning and calculations. This generalized structure has demonstrated a high level of accuracy in solving the most complex fifth-level problems in a MATH dataset.
18. MLPO (Multi-agent guided Leader Policy Optimization): MLPO is a hierarchical architecture that uses the collective intelligence of multiple agents, minimizing training costs. Only one LLM "Leader" has been trained, while the team of supporting agents remains untrained. The Leader interviews the team, summarizes their proposing solutions and synthesizes the final answer. MLPO introduces a new GRPO-based task that implicitly evaluates and synthesizes agent responses, ensuring high efficiency of collaborative reasoning and high performance in benchmarks like MATH.
19. AdCo (Adaptive Coopetition): This new inference-time framework introduces an adaptive "coopetition" mechanism driven by Upper Confidence Bound (UCB) logic. AdCo allows LLM agents to dynamically switch between collaboration and competition strategies based on coarse verifier signals in each round. This mechanism promotes robust reasoning by leveraging a knowledge variety of models and enabling uncertainty-driven exploration, especially in environments where agents have comparable capabilities. The architecture has achieved significant relative performance improvements (up to 20%) in complex reasoning benchmarks and is stable in various configurations. The Worker agents coordinate their actions through the Pub/Sub channel and achieve consensus by majority voting.

The complexity of Multi-Agent Large Language Model (MA-LLM) systems currently precludes definitive, clean numerical comparisons across the field. Architectural designs differ fundamentally. Some focus on enhancing training data quality (e.g., AgenticMath), others on inference-time structured debate (e.g., DiMo), and yet others on policy optimization via reinforcement learning (e.g., ReMA, Dr. MAMR). Furthermore, the heterogeneity of experimental setups—ranging from the base LLM model used (Llama vs. Qwen vs. proprietary APIs), fine-tuning strategies (SFT, DPO, RL), to evaluation metrics (Pass@1 vs. Pass@K) that renders direct score-based ranking unreliable.

To address the core challenge of incomparability, we separate the analysis into two layers. First, we provide a qualitative and structural comparison (Key Dimensions Bullet Grid) to establish the

 doi:10.20944/preprints202512.1105.v1

architectural principles and resource commitments of each system. Second, we present numerical results in performance tables, organizing scores into comparable groups and providing explicit annotations to control for differences in testing environments.

4. Comparative Results on Mathematical Reasoning Benchmarks

This synthesized view establishes the intrinsic design and the extrinsic evaluation environment for all 19 architectures, providing the essential context required for analyzing subsequent performance metrics.

AgenticMath

- Agent Types: Rephrase agent, Review agent, Revise agent, Solver agent, Evaluator agent.
- Coordination Strategy: Sequential pipeline across four stages: Seed Filtering, Problem Synthesis, Solution Generation, and Quality Evaluation. The core synthesis involves a collaborative review–revise loop.
- Planning: Structured workflow, guided by explicit quality-control criteria, to control prompt and solution creation.
- Verification: Rigorous multi-dimensional scoring (Clarity, Correctness, Diversity) performed by the Evaluator agent. Final selection retains only top-scoring samples.
- External Tools: N/A.
- Purpose: Generate a high-quality, data-efficient dataset (AgenticMathQA) for Supervised Fine-Tuning (SFT) based on the "Less Is More" principle.
- Model Used: GPT-4o-mini (data generation/Evaluator); DeepSeekMath-7B, Mistral-7B, Llama3-8B (evaluation/SFT targets).
- Training Style: Multi-agent synthetic data generation combined with SFT.
- Benchmarks: MATH, GSM8K, CollegeMath, DM Mathematics, OlympiadBench, TheoremQA.
- Prompting: Role-specific prompts are used during the data creation process.
- Fine-tuning?: Yes (SFT performed on the generated AgenticMathQA dataset).
- Inference Architecture: Single LLM (The resulting fine-tuned model).

MALT

- Agent Types: Generator (G), Verifier (V), Refiner (R).
- Coordination Strategy: Sequential chaining (G -> V -> R) during inference. Training involves tree-based sampling for specialized trajectories.
- Planning: Implicit sequential planning defined by the three-stage G -> V -> R specialized workflow.
- Verification: Dedicated Verifier agent (V) provides explicit feedback; credit assignment is sparse outcome reward-based but refined using a value-iteration strategy.
- External Tools: N/A.
- Purpose: Improve reasoning by jointly training specialized agents using SFT and DPO to learn role-specific reasoning meta-strategies.
- Model Used: Llama-3.1-8B-Instruct, Qwen-2.5-1.5B-Base.
- Training Style: Role-specific SFT + DPO post-training.
- Benchmarks: GSM8K, MATH, CommonsenseQA (CSQA).
- Prompting: Fixed role conditioning prompts are applied.
- Fine-tuning?: Yes (SFT and DPO applied to individual agents).
- Inference Architecture: Sequential Multi-Agent (G -> V -> R).

MARS-PO

- Agent Types: K agents used during the training data generation phase.

 doi:10.20944/preprints202512.1105.v1

- Coordination Strategy: Collaborative response generation where agents' high-quality outputs are merged to construct robust preference pairs.
- Planning: Implicit (focused on sampling diverse reasoning chains for preference data creation).
- Verification: Final answer correctness against ground truth and an external reward model score candidate solutions.
- External Tools: N/A.
- Purpose: Enhance mathematical reasoning by generating high-quality preference data for Hybrid Preference Optimization fine-tuning.
- Model Used: Llama3.1-8B-Instruct.
- Training Style: Hybrid Preference Optimization (DPO-like training).
- Benchmarks: GSM8K, MATH.
- Prompting: CoT reasoning steps are utilized.
- Fine-tuning?: Yes (Hybrid DPO applied to the target LLM).
- Inference Architecture: Single LLM (Fine-tuned model).

MARS (Review System)

- Agent Types: Author agent, Reviewers (K), Meta-reviewer.
- Coordination Strategy: Review Process (Propose -> Review -> Feedback -> Update), characterized by independent reviewer assessment to avoid costly reviewer-to-reviewer interactions.
- Planning: N/A (Focus is on efficient inference-time collaboration and error detection).
- Verification: Reviewer agents evaluate the Author's response and assign a confidence level; the Meta-reviewer integrates this feedback for the final decision/revision guidance.
- External Tools: N/A.
- Purpose: Efficient inference-time collaboration that matches the accuracy of Multi-Agent Debate (MAD) while reducing token consumption and inference time by approximately 50%.
- Model Used: gpt-3.5-turbo, mixtral-8x22b.
- Training Style: Inference-only collaboration strategy.
- Benchmarks: GPQA, MMLU, GSM8K.
- Prompting: CoT is enabled for all models; role-based prompts define the Author/Reviewer/Meta-reviewer functions.
- Fine-tuning?: No (Inference-time approach).
- Inference Architecture: Review System Multi-Agent collaboration.

MARS (Socratic Guidance)

- Agent Types: Planner, Manager, UserProxy, Teacher, Critic, Student, Target.
- Coordination Strategy: Socratic dialogue pattern (Teacher-Critic-Student) for iterative prompt optimization.
- Planning: Autonomous path planning by the Planner agent to ensure tailored optimization for different tasks.
- Verification: Iterative validation in the Target agent via performance metrics guides the optimization process.
- External Tools: N/A.
- Purpose: Automated Prompt Optimization (APO) to search for optimal, robust prompts for a target LLM across diverse domains.

 doi:10.20944/preprints202512.1105.v1

- Model Used: GPT-4o-mini (for agents/APO).
- Training Style: Automated Prompt Optimization via multi-agent dialogue.
- Benchmarks: GSM8K (Math); BBH/MMLU/Domain-Specific.
- Prompting: Iteratively refined and optimized prompts are the primary output.
- Fine-tuning?: No (Optimizes prompts only).
- Inference Architecture: Multi-Agent Collaborative Framework (during optimization); final inference uses the target model with the optimized prompt.

FMAD

- Agent Types: Debater A, Debater B, Judge.
- Coordination Strategy: Multi-round debate (N=2 rounds) focused on step-wise correctness for data creation.
- Planning: Implicit (focused on generating sequential reasoning steps for debate).
- Verification: Judge reviews debate history, selects a winner, and assigns a confidence score (, 0-100%) to indicate the likelihood of the reasoning step being correct.
- External Tools: N/A.
- Purpose: Automated annotation of fine-grained, step-level reasoning data (MMATH-Data) to train a Multi-Agent Debate Reward Model (MRM) for RL fine-tuning.
- Model Used: Qwen2.5-14B-Instruct (Debaters), LLaMA-3-8B-Instruct (Judge). MMATH-LLM is the final trained model (8B class).
- Training Style: Multi-agent synthetic data creation + RL fine-tuning (MMATH-LLM) using the MRM.
- Benchmarks: GSM8K, MATH (MATH-500 subset).
- Prompting: Structured debate prompts for role definition and argumentation.
- Fine-tuning?: Yes (RL fine-tuning on MMATH-Data).
- Inference Architecture: Single LLM (Fine-tuned MMATH-LLM).

DiMo

- Agent Types: Generator, Evaluator, Refiner, Judger.
- Coordination Strategy: Structured debate dynamically switched between Divergent Thinking Mode (for breadth) and Logical Thinking Mode (for step-wise deduction, effective for math).
- Planning: Explicit mode selection mechanism based on task affinity.
- Verification: Logical Thinking Mode enforces an evaluate–refine–judge loop for step-wise verification.
- External Tools: N/A (The framework is designed to be semantics-aware and Web-native, yielding URL-annotated evidence chains, though these external resources are not explicitly used in the math task setup described).
- Purpose: Enhance performance and achieve process transparency (interpretability) by simulating structured debate using diverse thinking modes.
- Model Used: LLaMA-3-8B, Qwen-2.5-32B (same checkpoint shared across roles).
- Training Style: Inference-only structured debate.
- Benchmarks: GSM8K, GSM-hard.
- Prompting: Role-specialized prompts and mode-specific decoding temperatures are used.
- Fine-tuning?: No (Inference-time approach).
- Inference Architecture: Structured Debate (4 agents operating in specific modes).

 doi:10.20944/preprints202512.1105.v1

SIER

- Agent Types: Generator (G), Evaluator (E).
- Coordination Strategy: Swarm Intelligence Enhanced Reasoning (SIER) based on Density-Driven Pareto Front Selection, guiding agents to collaboratively search for optimal solutions.
- Planning: Reasoning is framed as an optimization problem where agents search for global optima in the solution space.
- Verification: Step-level Pareto front selection jointly optimizes quality (PRM) and diversity (KDE) metrics; uses step-level quality evaluation for error correction.
- External Tools: N/A.
- Purpose: Enhance solution diversity and quality by mitigating model convergence to local optima, efficiently exploring the solution space.
- Model Used: Not explicitly stated, tested against CoT and RGS baselines.
- Training Style: Inference-time search strategy.
- Benchmarks: AIME-2024, AIME-2025, MATH-500, MATH-500 (level-5), GSM8K, LiveMathBench, MMLU-STEM.
- Prompting: Implicitly based on generating multiple solution paths for evaluation.
- Fine-tuning?: No (Inference-time search strategy).
- Inference Architecture: Agent-based Swarm Intelligence optimization loop.

ECON

- Agent Types: Coordinator LLM (1), Execution LLMs (N).
- Coordination Strategy: Hierarchical Coordination aiming for a Bayesian Nash Equilibrium (BNE). Execution LLMs operate independently based on their belief networks, removing costly direct communication.
- Planning: Coordinator proposes a centralized strategy (embedding) which controls generation behavior of Execution LLMs.
- Verification: Verification is implicit via the optimization towards BNE, providing theoretical convergence guarantees.
- External Tools: N/A.
- Purpose: Efficient, scalable coordination with theoretical convergence guarantees by modeling coordination as an incomplete-information game.
- Model Used: LLaMA3.1 8B/70B/405B, Mistral-7B, Mixtral-8x22B, Qwen1.5 110B, GPT4 turbo.
- Training Style: Hierarchical Reinforcement Learning (RL) to optimize agent policy/prompt embeddings to achieve BNE.
- Benchmarks: GSM8K, GSM-Hard, MATH, SVAMP (Math); StrategyQA, Travelplanner (General/Planning).
- Prompting: Zero-shot setting used for comparison, with a general prompt for execution agents.
- Fine-tuning?: Yes (LLM parameters or prompt embeddings optimized via RL).
- Inference Architecture: Hierarchical (Coordinator + Execution LLMs).

LbMAS

- Agent Types: Query-related experts, Planner, Decider, Critic, Cleaner, Conflict-resolver, Control unit (LLM agent).
- Coordination Strategy: Centralized communication via Blackboard architecture (shared context/memory). The LLM Control Unit iteratively selects agents to interact.

 doi:10.20944/preprints202512.1105.v1

- Planning: Dynamic agent selection and execution loop based on the shared context in the Blackboard.
- Verification: Critic, Cleaner (redundancy management), and Conflict-resolver agents monitor the process and blackboard content.
- External Tools: N/A.
- Purpose: Dynamic problem-solving utilizing a shared public storage space (Blackboard) to replace individual memory modules, resulting in competitive and token-economical performance.
- Model Used: Llama-3.1-70b-Instruct, Qwen-2.5-72b-Instruct (random selection).
- Training Style: Inference-only execution based on the Blackboard system.
- Benchmarks: MATH, GSM8K, MMLU, ARC-Challenge, GPQA, BBH.
- Prompting: Expert prompts define roles and expertise; unique system prompt for predefined agents.
- Fine-tuning?: No.
- Inference Architecture: Blackboard Architecture (dynamic agent orchestration).

MA ToT

- Agent Types: Reasoner agents (parallel), Thought Validator.
- Coordination Strategy: Parallel Tree-of-Thought (ToT) generation followed by filtering and consensus voting.
- Planning: ToT enables broad exploration of reasoning paths.
- Verification: Dedicated Thought Validator evaluates and filters reasoning branches to check faithfulness and reliability.
- External Tools: N/A.
- Purpose: Enhance complex arithmetic reasoning by systematically evaluating and eliminating early-stage inferential errors.
- Model Used: GPT-3.5-turbo, GPT-4o-mini, Llama 3.1 8B, Llama 3.1 70B.
- Training Style: Inference-only search strategy (ToT + Validation).
- Benchmarks: GSM8K.
- Prompting: CoT and ToT prompts used for Reasoners.
- Fine-tuning?: No.
- Inference Architecture: Multi-agent Tree-of-Thought structure with dedicated validation.

Dr. MAMR

- Agent Types: Meta-thinking Agent, Reasoning Agent.
- Coordination Strategy: Hierarchical, multi-turn interaction addressing the "lazy agent" problem; employs a verifiable reward mechanism for adaptive deliberation (discarding noisy outputs, restarting).
- Planning: Meta-thinking agent proposes plans and monitors progress.
- Verification: Causal Influence (CI) estimation (Shapley-inspired) measures agent contribution for precise credit assignment. Verifiable Reward (RB) encourages deliberation.
- External Tools: N/A.
- Purpose: Mitigate the "lazy agent" issue arising in hierarchical RL multi-agent setups like ReMA, unlocking the full potential for complex reasoning.
- Model Used: Qwen2.5 3B/7B/14B Instruct.

 doi:10.20944/preprints202512.1105.v1

- Training Style: Multi-turn GRPO + Causal Influence + Verifiable Reward (RL fine-tuning).
- Benchmarks: MATH500, GSM8K, AIME, AMC23, GaoKao2023En, Minerva Math, OlympiadBench.
- Prompting: VRP (CoT) prompting.
- Fine-tuning?: Yes (RL fine-tuning).
- Inference Architecture: Hierarchical Multi-Agent.

ReMA

- Agent Types: High-level (Meta-thinking) agent, Low-level (Reasoning) agent.
- Coordination Strategy: Hierarchical Multi-Agent Reinforcement Learning (MARL) where agents are separated by role-specific system prompts.
- Planning: High-level agent generates a complete meta-thinking trace/instructions.
- Verification: Final outcome reward is assigned to guide policy optimization via multi-turn GRPO.
- External Tools: N/A.
- Purpose: Elicit metacognition by explicitly separating strategic oversight (meta-thinking) from detailed execution (reasoning) using MARL.
- Model Used: Llama3-8B-Instruct, Llama3.1-8B-Instruct, Qwen2.5-7B-Instruct.
- Training Style: Multi-agent RL (Multi-turn GRPO).
- Benchmarks: MATH500, GSM8K, AIME, AMC23, GaoKao2023En, Minerva Math, Olympiad Bench.
- Prompting: VRP (CoT) prompting.
- Fine-tuning?: Yes (RL fine-tuning).
- Inference Architecture: Hierarchical Multi-Agent.

ANN

- Agent Types: Layered Agent Teams (Nodes); roles are dynamically generated and refined.
- Coordination Strategy: Layered Neural Network Architecture (Forward/Backward optimization phases). Collaboration is optimized through continuous textual gradients.
- Planning: Dynamic task decomposition (Forward Phase); Adaptive role refinement (Backward Phase).
- Verification: Textual Backpropagation refines roles, prompts, and coordination based on a textual loss signal (error signal from validation).
- External Tools: N/A.
- Purpose: Self-evolving, highly flexible MAS optimized through textual gradients, unifying prompt engineering, architecture selection, and optimization.
- Model Used: GPT-4o mini, GPT-3.5-turbo (training/optimization); GPT-3.5, GPT-4o-mini, GPT-4 (evaluation).
- Training Style: Textual Backpropagation (Optimizes prompts, roles, and collaboration structure).
- Benchmarks: MATH, HumanEval, Creative Writing, DABench.
- Prompting: Automated prompt tuning/refinement.
- Fine-tuning?: No (Optimizes prompts/roles textually using LLMs).
- Inference Architecture: Layered Neural Network Architecture.

 doi:10.20944/preprints202512.1105.v1

ILR

- Agent Types: Multiple LLMs (paired in groups, e.g., Group1: Llama-3.1-8B-Instruct paired with itself).
- Coordination Strategy: Dynamic Interaction (DI) mechanism selects Cooperation or Competition mode based on estimated question difficulty. Interaction follows the Idea3 framework (Sharing, Analysis, Fusion).
- Planning: Difficulty estimation via self-ranking prompts and Item Response Theory guides mode selection.
- Verification: Perception Calibration (PC) module integrates peer LLM's reward distribution into the agent's reward function, allowing agents to perceive peer solution quality automatically.
- External Tools: N/A.
- Purpose: Enhance the independent problem-solving capability of LLMs through collaborative learning/training, resulting in a single fine-tuned model for inference.
- Model Used: Llama-3.1-8B-Instruct, Qwen2.5-7B-Instruct, Qwen2.5-14B-Instruct.
- Training Style: Interactive Learning (MARL via Group Relative Policy Optimization - GRPO).
- Benchmarks: MATH-500, GSM8K, Minerva Math, Olympiad Bench, AIME24&25.
- Prompting: Idea3 prompts for cooperation/competition modes.
- Fine-tuning?: Yes (GRPO optimization applied to all participating LLMs).
- Inference Architecture: Single LLM (Fine-tuned model solves problems independently).

MathChat

- Agent Types: Conversational agents (implicit user/LLM roles).
- Coordination Strategy: Zero-shot conversational prompting encourages step-by-step reasoning verified by external tools.
- Planning: Implicit iterative refinement based on tool execution results.
- Verification: Verification of each intermediate step using external computational tools (Python) or knowledge bases is a core mechanism.
- External Tools: Python interpreter, external databases.
- Purpose: Enhance mathematical problem-solving robustness and reliability via external computational tool integration, without requiring prompt tailoring per problem.
- Model Used: GPT-4.
- Training Style: Inference-only (Conversational + Tool Use).
- Benchmarks: 24-point game (primary focus), sampled subsets of typical math topics.
- Prompting: Zero-shot conversational prompts focusing on the tool-using format.
- Fine-tuning?: No (Zero-shot prompting).
- Inference Architecture: Conversational multi-agent system.

MACM

- Agent Types: Thinker, Judge, Executor.
- Coordination Strategy: Iterative Conditional Mining (ICM) process, where agents collaborate through clearly defined identities and specific formatting instructions.
- Planning: Abstracting conditions and objectives from the problem into a dynamic Condition List.
- Verification: Judge inspects reasoning; Executor ensures numerical computation correctness.
- External Tools: N/A (Executor handles computation internally).

 doi:10.20944/preprints202512.1105.v1

- Purpose: Solve complex math problems (Level 5) by ensuring depth of thinking and eliminating calculation errors, while achieving strong generalizability.
- Model Used: GPT-4 Turbo, GPT-3.5.
- Training Style: Inference-only (Conditional Mining prompting method).
- Benchmarks: MATH (overall and Level 5), 24-point game.
- Prompting: MACM role-defined prompts ensure consistent response formats.
- Fine-tuning?: No.
- Inference Architecture: Multi-Agent System (Thinker, Judge, Executor).

MLPO

- Agent Types: Leader LLM (explicitly trained), Agent Team (Peer LLMs, typically untrained and heterogeneous).
- Coordination Strategy: Hierarchical collaboration where the Leader is trained to implicitly evaluate and synthesize peer responses, guiding the team towards the right direction.
- Planning: Leader orchestrates and guides the collaborative reasoning.
- Verification: Leader selection and synthesis process, optimized using the MLPO objective (Multi-agent Guided GRPO), serves as implicit verification.
- External Tools: N/A.
- Purpose: Enhance multi-agent collaboration by training a policy (the Leader) to leverage diversity and synthesize high-quality outcomes from an agent team.
- Model Used: Qwen2.5 7B Instruct (Leader); Llama 3.1 8B Instruct, Gemma2 9B Instruct, Qwen2.5 7B Instruct (Agent Team).
- Training Style: Hierarchical RL based on Multi-agent Guided GRPO (MLPO).
- Benchmarks: BBH, MATH, MMLU.
- Prompting: N/A (Training approach focused on policy optimization).
- Fine-tuning?: Yes (Leader model is fine-tuned via RL; Peers are off-the-shelf and untrained).
- Inference Architecture: Hierarchical (Trained Leader + Untrained Peer Team).

AdCo

- Agent Types: Agent Worker Cluster (Multiple Agents), Verifier model.
- Coordination Strategy: Adaptive Coopetition (Collaboration or Competition) guided by a UCB (Upper Confidence Bound) algorithm; agents iteratively refine reasoning based on peer feedback.
- Planning: Uncertainty-driven exploration determines whether to collaborate or compete at each round.
- Verification: A coarse verifier signal, the Process Reward (PR) derived from Qwen2.5-Math-PRM-7B, guides the adaptive decision-making.
- External Tools: N/A (Built using the AutoGen framework).
- Purpose: Enhance resilience and robustness in inference-time reasoning by switching strategy based on uncertainty, without relying on high-performance verifiers.
- Model Used: DeepSeek/DeepSeek-v3-0324, Gemma-3-27b-it, GPT-4o (Agents); Qwen2.5-Math-PRM-7B (Verifier).
- Training Style: Inference-time search strategy (Adaptive Coopetition).
- Benchmarks: DeepMath-103K (sampled subsets).
- Prompting: N/A (Strategy comparison focused).

 doi:10.20944/preprints202512.1105.v1

- Fine-tuning?: No (Inference-time strategy).
- Inference Architecture: Adaptive Agent Cluster.

5. Performance Table With Explicit Disclaimers

5.1. GSM8K Benchmark Accuracy (Focusing on ≈ 7B-8B Models)

This group evaluates performance on grade school arithmetic reasoning for mid-sized open LLMs (typically 7B or 8B parameters), making comparisons of training efficacy (SFT/RL) and inference strategies (MAS vs. fine-tuned single agent) more relevant.

Framework	**Model(s)**	**Reported Metric**	**Score (%)**
DiMo (Logical Thinking Mode)	LLaMA-3-8B	Accuracy	90.7
DiMo (Divergent Mode)	LLaMA-3-8B	Accuracy	79.9
MALT	Llama-3.1-8B-Instruct	Accuracy	90.50
ILR (Group1)	Llama-3.1-8B-Instruct	Accuracy	89.39
ILR (Group1)	Qwen2.5-7B-Instruct	Accuracy	93.40
ReMA	Llama3.1-8B-Instruct	Accuracy	87.26
FMAD	MMATH-LLM (8B class)	Accuracy	83.4
AgenticMath	Llama3-8B	Accuracy	80.1
AgenticMath	Mistral-7B	Accuracy	82.3
ReMA	Qwen2.5-7B-Instruct	Accuracy	90.60
MARS-PO (iter3)	Qwen2.5-Math-7B-Instruct	Accuracy	95.82
Mars-PO (iter1)	Qwen2.5-Math-7B-Instruct	Accuracy	95.75
Dr. MAMR	Qwen2.5-7B-Instruct	Accuracy	92.12
ECON	LLaMA3.1 8B	Accuracy	92.70
MA ToT	Llama3.1-8B	Accuracy	89.0
SIER	Qwen2.5-7B-Instruct, Qwen2.5-Math-PRM-72B	Pass@8	97.4

5.2. GSM8K Benchmark Accuracy (Focusing on Larger/Inference-Time MAS)

This group highlights the potential of larger open models ($>$14B) or proprietary models, often used directly within multi-agent inference systems for maximum performance.

Framework	**Model**	**Reported Metric**	**Score (%)**
DiMo (Logical Mode)	Qwen-2.5-32B	Accuracy	98.4
LbMAS	Qwen-2.5-72b-Instruct	Accuracy	96.05%
ILR (Group3)	Qwen2.5-14B-Instruct	Accuracy	95.30

 doi:10.20944/preprints202512.1105.v1

Framework	Model	Reported Metric	Score (%)
MARS (Review System)	mixtral-8x22b	Accuracy	90.33
MARS (Socratic Guidance	GPT-4o (Base Model)	Accuracy	90.97
ECON	LLaMA3.1 70B	Accuracy	92.70
MA ToT	Llama3.1-70B	Accuracy	94.8
MA ToT	Gpt-4o-mini	Accuracy	92.2
AdCo	GPT-4o, Gemma-3-27b-it, DeepSeek-v3-0324	Accuracy (5k samples)	91.8 (GSM-Symbolic)
Dr. MAMR	Qwen2.5-14B-Instruct	Accuracy	93.69

5.3. MATH Benchmark Accuracy (Competition-Level Reasoning, ≈ 7B-8B Models)

The MATH benchmark demands complex symbolic and multi-step logical reasoning. This group assesses how multi-agent training and architectural decisions impact performance on this challenging task for mid-sized LLMs.

Framework	Model	Reported Metric	Score (%)
ILR (group 3)	Qwen2.5-7B-Instruct	Accuracy (MATH-500)	78.00
ILR (group 1)	Qwen2.5-7B-Instruct	Accuracy (MATH-500)	77.60
ILR (group 1)	Llama-3.1-8B-Instruct	Accuracy (MATH-500)	55.80
MARS-PO (iter 3)	Llama3.1-8B-Instruct	Accuracy	57.82
Mars-PO (iter 1)	Llama3.1-8B-Instruct	Accuracy	55.48
MALT	Llama-3.1-8B-Instruct	Accuracy	57.25
AgenticMath	DeepSeekMath-7B	Accuracy	55.0
AgenticMath	AgenticMath-Mistral-7B (60K)	Accuracy	39.5
FMAD	MMATH-LLM (8B class)	Accuracy	45.1
ReMA	Llama3.1-8B-Instruct	Accuracy (MATH500)	53.20
ReMA	Qwen2.5-7B-Instruct	Accuracy (MATH500)	74.40
LbMAS	Qwen-2.5-7B-Instruct	Accuracy	78.6
Dr. MAMR	Qwen2.5-7B-Instruct	Accuracy	78.60
MLPO	Qwen2.5 7B Instruct(Leader Model), Llama-3.1-8B-Instruct, Gemma 2 9B Instruct	Accuracy	76.2
ECON	LLaMA3.1 8B, LLaMA3 8B, Mixtral 7B	Accuracy	74.24
ECON	LLaMA3.1 8B	Accuracy	67.70
SIER	Qwen2.5-7B-Instruct, Qwen2.5-Math-PRM-72B	pass@8 (MATH-500)	93.0

5.4. MATH Benchmark Accuracy (Competition-Level Reasoning, Larger/Proprietary Models)

 doi:10.20944/preprints202512.1105.v1

This group aggregates results achieved on the highly challenging MATH dataset (or specialized, difficult subsets like MATH-500 or Level 5), specifically featuring architectures utilizing larger open-source models (typically N≥32B parameters) or proprietary, high-capacity models (e.g., GPT-4 family). These architectures often rely on multi-agent inference or extensive fine-tuning powered by large models.

Framework	Model	Benchmark	Reported Metric	Score (%)
ANN	GPT-4o-mini	MATH	Accuracy	82.8
ILR (group 3)	Qwen2.5-14B-Instruct	MATH-500	Accuracy	82.60
SIER	Qwen2.5-7B-Instruct, Qwen2.5-Math-PRM-72B	MATH-500	Pass@8	93.0
ECON	Mixtral 8×22B, Qwen1.5 110B, LLaMA3.1 405B	MATH	Accuracy	85.46
ANN	GPT-4	MATH	Accuracy	80.0
LbMAS	Qwen-2.5-72b-Instruct	MATH	Accuracy	78.6
MACM	GPT-4 Turbo	MATH	Accuracy	87.92
AdCo	GPT-4o, Gemma-3-27b-it, DeepSeek-v3-0324	DeepMath-103K	Accuracy	54.0
Dr. MAMR	Qwen2.5-14B-Instruct	MATH500	Accuracy	80.40

5.5. Notes on Comparability

To ensure a critical and accurate interpretation of the numerical results presented in the comparative performance tables, we must account for the following methodological variations and limitations that directly impact the measured scores:

- Pass@k vs. Pass@1: Many inference-time search and optimization systems, such as SIER (Swarm Intelligence Enhanced Reasoning), report performance using Pass@k (e.g., Pass@8). This metric significantly inflates the reported accuracy compared to standard Accuracy, which is used by most fine-tuned models and hierarchical MASs (e.g. ReMA, ILR, ECON).
- The use of diverse backbone models introduces an uncontrolled variable that often overshadows architectural benefits. Proprietary Model Ceiling: Architectures utilizing proprietary models (e.g., GPT-4 Turbo in MACM, GPT-4o in ANN) often achieve scores that are fundamentally unattainable by open-source models (e.g., Llama/Qwen 7B/8B families), regardless of architectural sophistication. Agent Team Diversity: Certain systems, such as LbMAS and ECON (for ablation studies), explicitly leverage heterogeneous agent teams, randomly drawing agents from different LLM families (e.g., Llama-3.1-70b-Instruct and Qwen-2.5-72b-Instruct in LbMAS). This deliberate heterogeneity maximizes diverse perspectives but makes comparison against homogeneous MAS setups difficult, as the score reflects a fusion of multiple foundation models' capabilities.
- Results are often selectively reported on subsets optimized for the demonstrated method's strengths, making universal interpretation insecure. Specialized and Niche Benchmarks: Other research uses highly specialized, non-standard datasets, such as AdCo on DeepMath-103K, which are chosen to avoid saturation observed in public benchmarks but lack external comparative baselines.

6. Discussion

 doi:10.20944/preprints202512.1105.v1

A study of nineteen different LLM-based multi-agent architectures has revealed a profound paradigm shift to solve complex mathematical problems. While early efforts were aimed at overcoming the disadvantages inherent in monolithic LLMs related to multi-stage calculations and logical sequencing, the modern architectural landscape reflects a mature emphasis on optimizing collaboration, information flows and, most importantly, specialization. Our main question about how these architectures can be improved focuses on the critical design trade-offs needed to overcome the specific system constraints inherent in coordination between multiple LLMs.

The main point of is the evolution from unstructured debates to rigorous specialized optimization frameworks. Debate-oriented models such as FMAD, which uses a debate structure to verify the correctness of step-by-step actions, and DiMo, which use structured debates to switch between divergent and logical thinking, provide the necessary transparency and verification mechanisms. However, the most advanced systems use hierarchical learning and reinforcement learning (RL) paradigms to structure the reasoning process, moving complexity from inference prompts to supervised training. Architectures such as ReMA and Dr. MAMR use hierarchical schemes that clearly separate high-level strategy (meta-thinking agent) and low-level execution (logical agent). This separation is vital, as evidenced by Dr. MAMR's concrete success in solving the "lazy agent" problem, where the contribution of one agent is minimal, by introducing a verifiable reward mechanism and measuring causality. Similarly, MLPO achieves high efficiency by training only one LLM Leader to synthesize the results of a team of untrained agents, minimizing training costs while leveraging the diversity of the team.

The constant problem of agent homogeneity, where agents working on the same basic LLM cannot generate truly diverse reasoning, has stimulated the development of innovative coordination strategies. To counter this, systems are exploring ways to apply specialized thinking or adaptive uncertainty management. ECON models coordination as a game with incomplete information aimed at achieving BNE, allowing Execution LLMs to work independently based on probabilistic assumptions, which avoids the high computational costs of direct interaction between agents. SIER considers logic as an optimization task, using a density-driven strategy to jointly optimize the quality and diversity of solutions at a step-by-step level, actively preventing convergence to local optima. Meanwhile, AdCo is implementing a logical inference-time mechanism in which agents dynamically switch between cooperation and competition. Guided by uncertainty, using the Upper Confidence Bound (UCB) logic, which promotes reliable reasoning, even if agents have comparable capabilities.

Future design outcomes should aim to create holistic, self-improving frameworks that systematically manage heterogeneity, cost, and validation. The effectiveness of systems that include external tools such as MathChat, which uses a User Proxy agent to control the operation of the tool and correct errors, confirms the constant need for reliable, proven calculations that do not depend on the reasoning abilities of LLM. Moreover, the difference between fine-tuning methods (MALT, ILR) and inference-time strategies (MTOT, MARS (Review System)) highlights an important trade-off: investing computing resources in training specialized agents (interactive ILR training) versus optimizing inference costs (MARS (Review System) reduces token consumption by 50% compared to traditional debates). For high-stakes mathematical reasoning, future systems must integrate the verifiable reward mechanisms described in Dr. MAMR, the effectiveness of belief-based coordination from ECON, and the flexibility of self-evolving frameworks such as ANN, while standardizing reporting metrics to overcome current challenges related to changes in core LLMs and the use of Pass@k metrics. Further success depends on developing an architecture that not only promotes a variety of reasoning, but also has the inherent ability to autonomously evaluate, refine, and verify that reasoning against strict mathematical principles.

7. Conclusion

The systematic review of these nineteen architectures confirms that multi-agent systems enable substantial improvement in mathematical problem-solving by structuring reasoning tasks and enforcing logical coherence. The core mechanism of improvement lies in distributing cognitive load

 doi:10.20944/preprints202512.1105.v1

through specialized agent roles—ranging from structured debate (FMAD, DiMo) to centralized, hierarchical control (Dr. MAMR, ReMA, MLPO). The comparative trade-offs center on balancing computational cost, particularly token consumption during inference (MARS (Review System), ECON), against the need to generate genuine solution diversity and mitigate agent homogeneity. Architectures that succeed best either employ sophisticated policy optimization frameworks (ILR, MARS-PO) or enforce stringent quality control pipelines (AgenticMath). Answering the central research question, these architectures demonstrate comparative advantages by effectively separating strategic meta-thinking (ReMA, Dr. MAMR) from execution, while their limitations often stem derive from the complexity of training coordination policies to large agent teams. Future multi-agent system design must prioritize autonomous self-improvement, demonstrated by ANN's textual backpropagation for evolving roles, and integrate verified computational tools (MathChat) to ensure mathematical accuracy. Moving forward, the development of scalable, adaptive architectures capable of dynamic heterogeneity and standardized verification remains crucial.

8. Data Availability Statement

This literature review is based on publicly available research papers and does not involve the collection or generation of new experimental data. All architectures, performance metrics, and comparative analyses presented in this study were extracted from the 23 peer-reviewed publications cited in the references section. The original papers and their associated datasets (including MATH, GSM8K, and other benchmarks) are publicly accessible through their respective publishers and repositories.

9. Conflicts of Interest

The authors declare no conflicts of interest.

References

1. Adaptive Coopetition: Leveraging Coarse Verifier Signals for Resilient Multi-Agent LLM Reasoning
2. From Debate to Equilibrium: Belief-Driven Multi-Agent LLM Reasoning via Bayesian Nash Equilibrium
3. MALT: Improving Reasoning with Multi-Agent LLM Training
4. Unleashing Diverse Thinking Modes in LLMs through Multi-Agent Collaboration
5. AgenticMath: Enhancing LLM Reasoning via Agentic-based Math Data Generation
6. Mars-PO: Multi-Agent Reasoning System Preference Optimization
7. Stop Overvaluing Multi-Agent Debate—We Must Rethink Evaluation and Embrace Model Heterogeneity
8. ReMA: Learning to Meta-think for LLMs with Multi-agent Reinforcement Learning
9. LLMs Working in Harmony: A Survey on the Technological Aspects of Building Effective LLM-Based Multi Agent Systems
10. MARS: toward more efficient multi-agent collaboration for LLM reasoning
11. MARS: A Multi-Agent Framework Incorporating Socratic Guidance for Automated Prompt Optimization
12. Interactive Learning for LLM Reasoning
13. Agentic Neural Networks: Self-Evolving Multi-Agent Systems via Textual Backpropagation
14. Debate4MATH: Multi-Agent Debate for Fine-Grained Reasoning in Math
15. MathChat: Converse to Tackle Challenging Math Problems with LLM Agents
16. Swarm Intelligence Enhanced Reasoning: A Density-Driven Framework for LLM-Based Multi-Agent Optimization
17. Exploring Advanced LLM Multi-Agent Systems Based on Blackboard Architecture
18. Literature Review Of Multi-Agent Debate For Problem-Solving
19. A Comprehensive Survey on Multi-Agent Cooperative Decision-Making: Scenarios, Approaches, Challenges and Perspectives
20. Unlocking the Power of Multi-Agent LLM for Reasoning: From Lazy Agents to Deliberation
21. MACM: Utilizing a Multi-Agent System for Condition Mining in Solving Complex Mathematical Problems

 doi:10.20944/preprints202512.1105.v1

22. Improving LLM Reasoning with Multi-Agent Tree-of-Thought Validator Agent
23. How to Train a Leader: Hierarchical Reasoning in Multi-Agent LLMs