The University of Texas at Austin
Cockrell School of Engineering

OMAC: A Holistic Optimization Framework for LLM-Based Multi-Agent Collaboration

Shijun Li, Hilaf Hasson, Joydeep Ghosh

Presented by Shijun Li
Department of Electrical and Computer Engineering, The University of Texas at Austin
05/12/2026

Introduction & Background

Multi-Agent Collaboration

- Multiple agents for task division
  - E.g., in code generation task, the agents can include: Problem Analyser, Algorithm Developer, Architecture Designer, Programmer, Tester, etc.
- Agent communication to bolster collaboration
  - E.g, to resolve mathematical problems, one agent can provide its own solution based on analysing and summarizing the answers from other agents.
- Multi-stage process of agent collaboration
  - Decompose a task into multiple steps, where the agents on each step target to resolve the problem based on results in previous steps.

Key Questions

- Agent creation and optimization
  - What types of agents should be integrated into the collaboration process, and how can we create and/or optimize them?

- Communication structure determination
  - How do the agents communicate with each other during the collaboration? Which agents should be involved at each step of the process?

Target & Method

Our Target

- Propose a general framework to optimize a multi-agent collaboration process
  - Our method can be generally applied to resolving complex tasks, within a multi-stage collaboration framework consisting of multiple LLM agents.
- Optimize both the communication (structure) and agents (functionality)
  - Given the task, optimize the functionality of the agents, as well as the agent collaboration and communication structure.

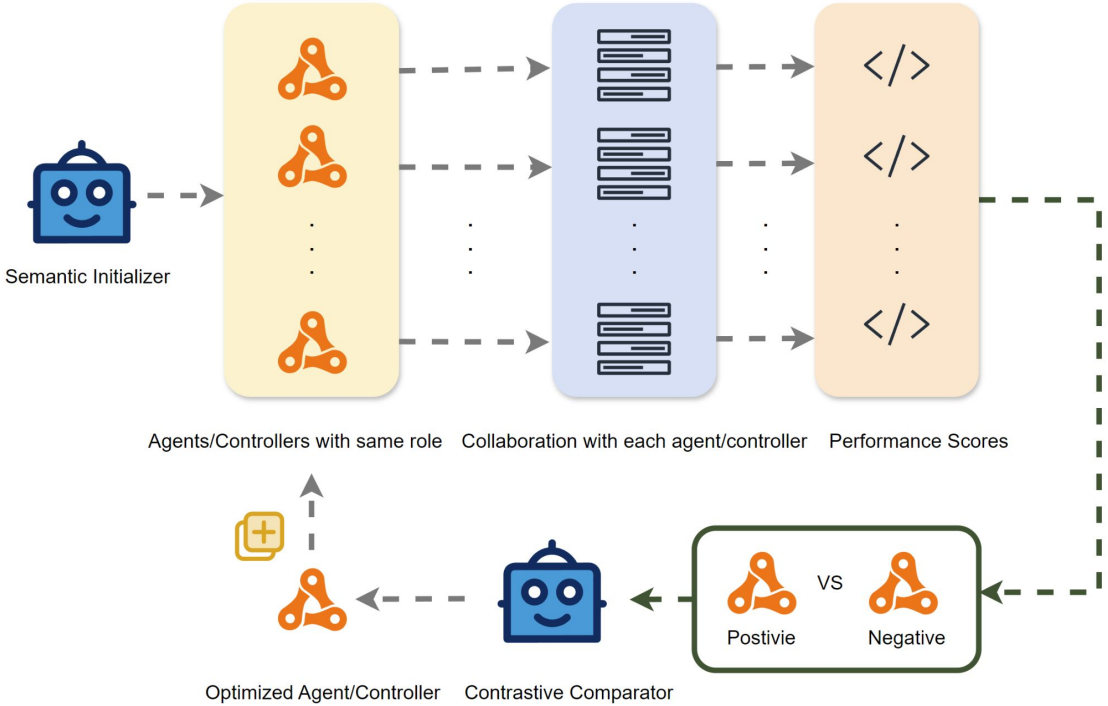
Five Key Dimensions for Optimization

- Functional Optimization - optimize the agents
  - Fun-1: Optimize existing candidate agents. E.g., optimize the instruction prompts and/or few-shot examples of these agents.
  - Fun-2: Generate and optimize a new agent to participate in the collaboration process.

Five Key Dimensions for Optimization

- Structural Optimization - optimize 3 types of controllers to control communication
  - Str-1: Choose the candidate agents from all existing agents before collaboration with an optimizable LLM.
  - Str-2: Select agents for participation and collaboration during certain step with an optimizable LLM.
  - Str-3: Use and optimize an LLM to decide how agents communicate with each other.

Optimization Algorithm for a Single Dimension



- any of the five dimensions for optimization

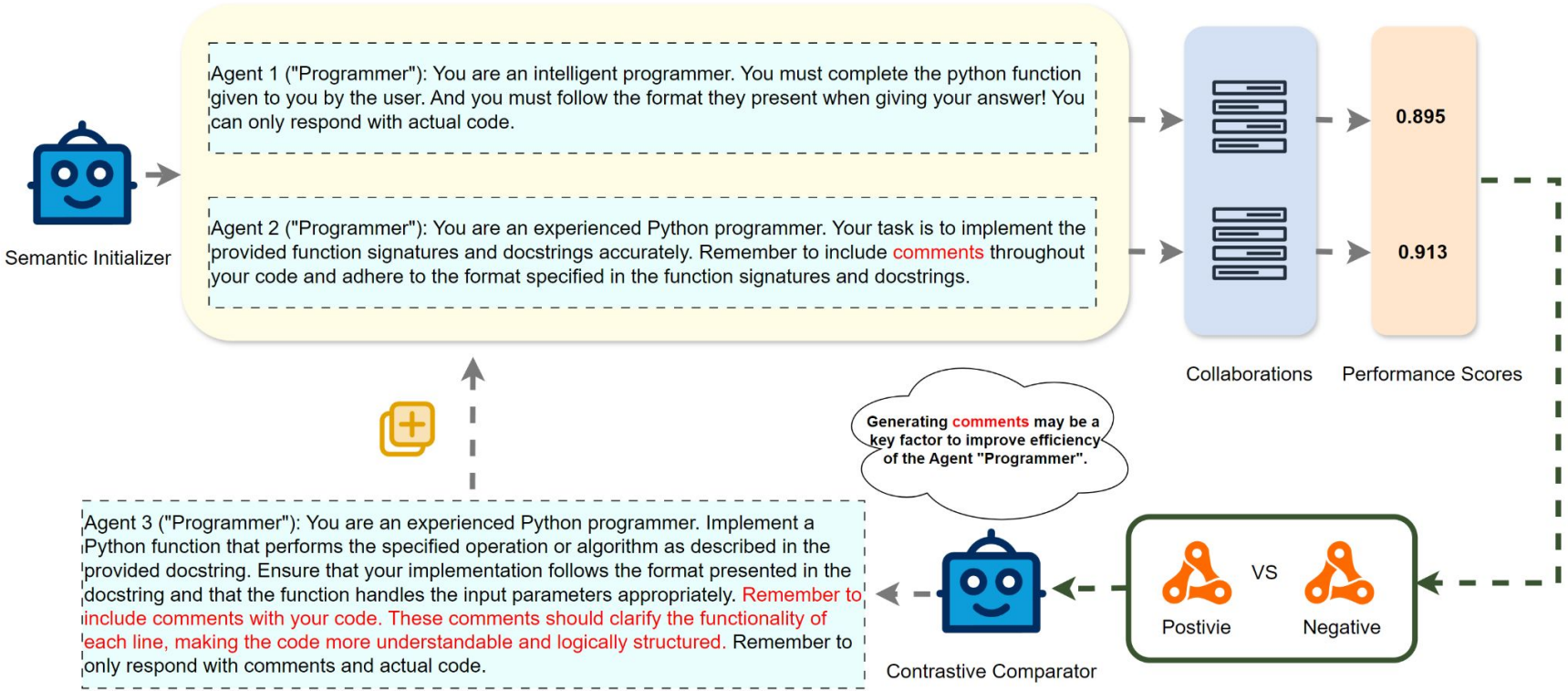
Semantic Initializer

- Initialize a candidate set of agents/controllers with the same role. E.g., create k "Programmer" to write python code with different instruction prompts.

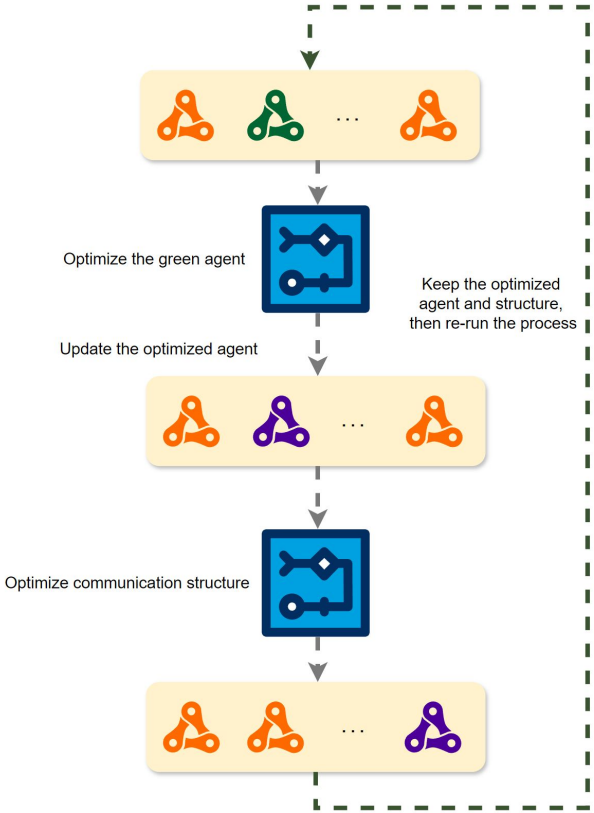
Contrastive Comparator

- Generate a new agent or controller by contrasting and reasoning the underlying factors for the given positive-negative pair.

Example - Agent Optimization (Fun-1)



Iterative Optimization for Multiple Dimensions



Iterative Optimization

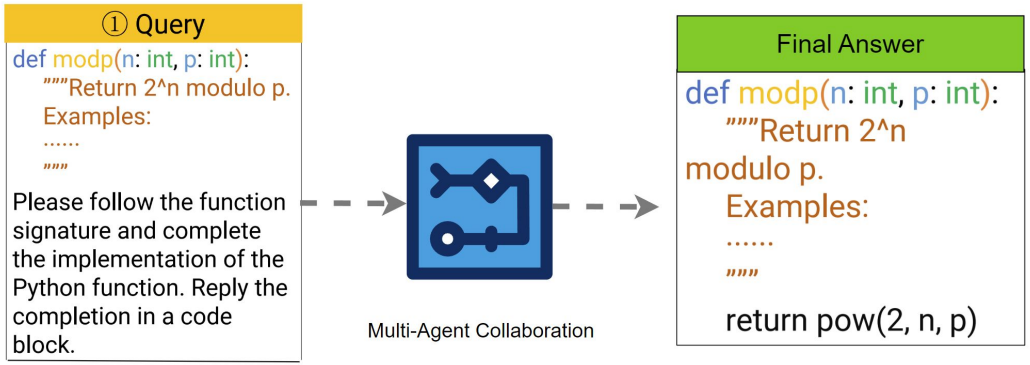
- Considering both structural and functional perspectives, we iteratively optimize several points by refining each one while keeping the others fixed.

Efficient Contrastive Ability

- Iterative optimization is to ensure the effectiveness of contrastive reasoning. That's to say, constrain the variables of a positive-negative pair within a narrow range to ensure all other factors remain consistent.

Applications & Experiments

Code Generation



- Dataset/Benchmark
  - HumanEval: 160 human-labeled function-level completion codes and unit tests.
- Evaluation Metric
  - Pass@1: pass rate of the generated code for each problem.
- Base LLM
  - gpt-3.5-turbo-1106

Experimental Results

➢ Optimize Single Functional Dimension

- For fun-1, there are seven existing agents to optimize, which are ['PythonAssistant', 'AlgorithmDeveloper', 'ComputerScientist', 'Programmer', 'Passer', 'Tester', 'Reflector']. For fun-2, we generate a new agent added for collaboration and optimization.

Method	Baselines				
	CodeT	AgentVerse	ADAS	AFlow	DyLAN
Pass@1	67.50±1.68	78.29±2.34	75.61±2.06	85.63±2.10	85.74±2.83

Method	OMAC (Functional Dimension)							
	Fun-1.1	Fun-1.2	Fun-1.3	Fun-1.4	Fun-1.5	Fun-1.6	Fun-1.7	Fun-2
Pass@1(%)	**88.39**±2.54	86.31±2.21	**88.87**±1.36	**89.25**±1.30	**88.74**±2.67	**88.39**±1.22	**88.34**±1.42	**86.77**±2.43

Experimental Results

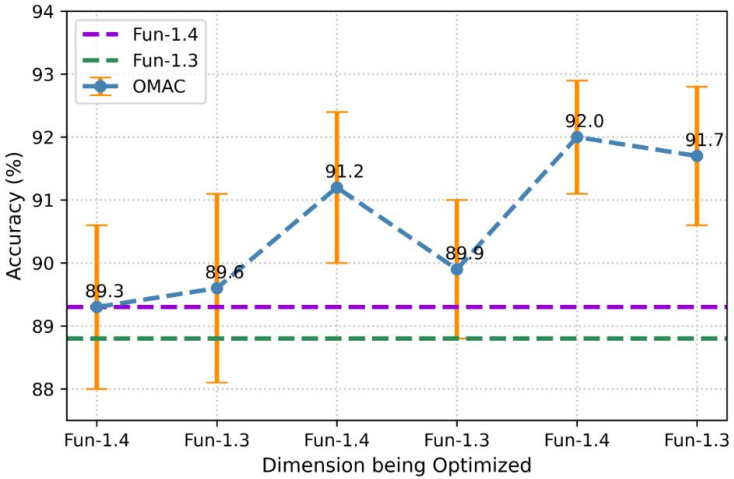
➢ Optimize Single Structural Dimension

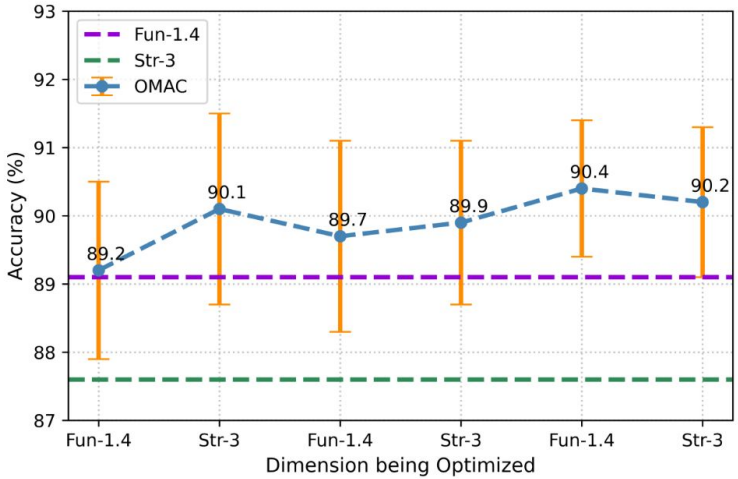
- ❑ For str-1, we optimize a controller to select candidate agents before collaboration. For str-2, we optimize a controller to rank and re-select agents during the collaboration. For str-3, we optimize a controller to globally decide whether the output of one agent should be incorporated as input context for another agent.

Method	Baselines					OMAC (Structural Dimension)		
	CodeT	AgentVerse	ADAS	AFlow	DyLAN	Str-1	Str-2	Str-3
Pass@1	67.50±1.68	78.29±2.34	75.61±2.06	85.63±2.10	85.74±2.83	**86.76**±1.22	**86.92**±2.27	**87.55**±2.46

➢ Optimize Multiple Dimensions

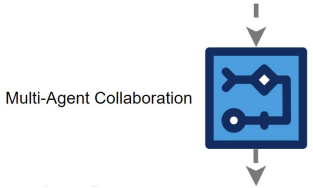
- ❑ Left: Optimize two best-performing dimensions (Fun-1.4 and Fun-1.3) one by one, repeating three iterations.
- ❑ Right: Optimize two sub-optimal dimensions (Fun-1.4 and Str-3) one by one, repeating three iterations.





Arithmetic Reasoning

Problem: Tom has a red marble, a green marble, a blue marble, and three identical yellow marbles. How many different groups of two marbles can Tom choose?



Solution: There are two cases here: either Tom chooses two yellow marbles (1 result), or he chooses two marbles of different colors ($\binom{4}{2} = 6$ results). The total number of distinct pairs of marbles Tom can choose is $1 + 6 = \boxed{7}$.

- Dataset/Benchmark
  - MATH: 7 subareas with 280 math questions and evaluation scripts.
- Evaluation Metric
  - Accuracy: ratio of the correct generated answers.
- Base LLM
  - gpt-3.5-turbo-1106

Experimental Results

➢ Optimize Single Dimension

- For fun-1, we can optimize the “System Prompt” or “Examples” of the agents. For fun-2, we generate and optimize a new agent into collaboration. For str-1, we optimize a controller to select candidate agents before collaboration. For str-2, we optimize a controller to rank and re-select agents during each turn of collaboration. For str-3, we optimize a controller to decide whether the output of one agent should be incorporated as input context for another agent.

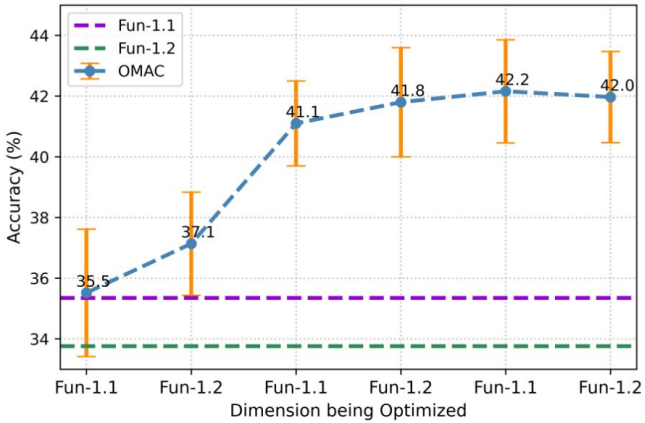
Method	Baselines					
	SE	SC	LLM Debate	ADAS	AFlow	DyLAN
Accuracy(%)	28.72±1.75	29.14±2.07	29.42±2.33	28.94±2.04	<u>32.49±2.62</u>	32.35±1.94

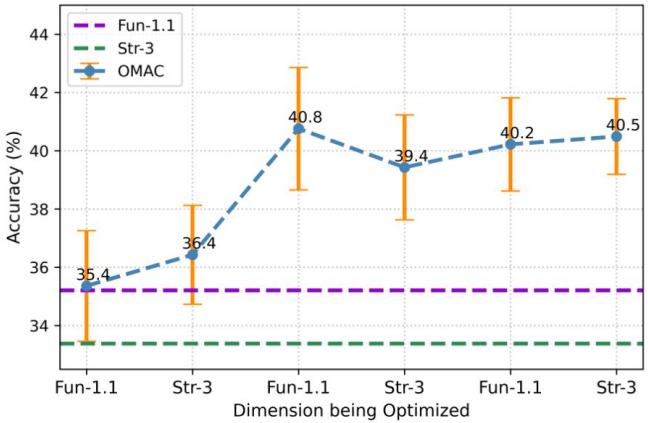
Method	OMAC (Structural Dimension)			OMAC (Functional Dimension)		
	Str-1	Str-2	Str-3	Fun-1.1	Fun-1.2	Fun-2
Accuracy(%)	**33.34**±1.45	**33.41**±1.37	**33.70**±1.62	**35.17**±1.96	**34.91**±2.03	**33.95**±1.30

Experimental Results

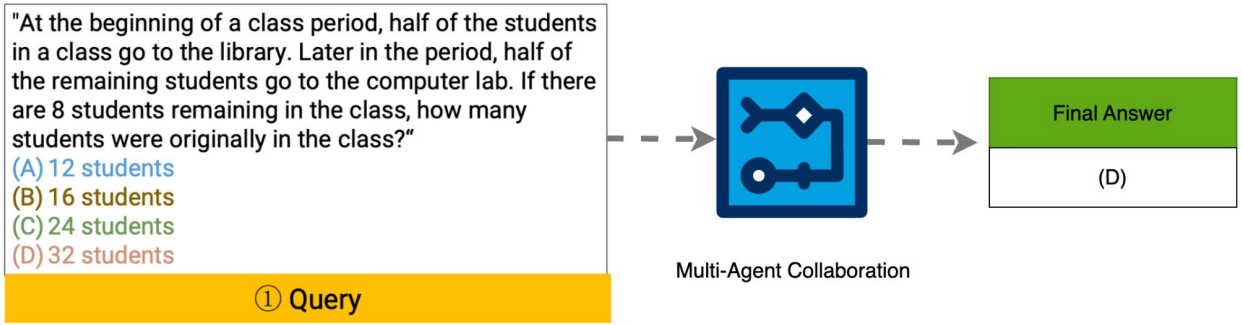
➢ Optimize Multiple dimensions

- ❏ Left: Optimize two best-performing dimensions (Fun-1.1 and Fun-1.2) one by one, repeating three iterations.
- ❏ Right: Optimize two sub-optimal dimensions (Fun-1.1 and Str-3) one by one, repeating three iterations.





General Reasoning



- Dataset/Benchmark
  - MMLU: multiple-choice questions across humanities, social sciences, hard sciences, and other fields.
- Evaluation Metric
  - Accuracy: ratio of the correct answers.
- Base LLM
  - gpt-3.5-turbo-1106

Experimental Results

➢ Optimize Single Dimension

- ❏ For fun-1, there are seven existing agents to optimize, which are [“Economist”, “Doctor”, “Lawyer”, “Mathematician”, “Psychologist”, “Programmer”, “Historian”]. For fun-2, we generate and add one agent for collaboration and optimization.For str-1, we optimize a controller to select candidate agents before collaboration. For str-2, we optimize a controller to rank and re-select agents during each turn of collaboration. For str-3, we optimize a controller to decide whether the output of one agent should be incorporated as input context for another agent.

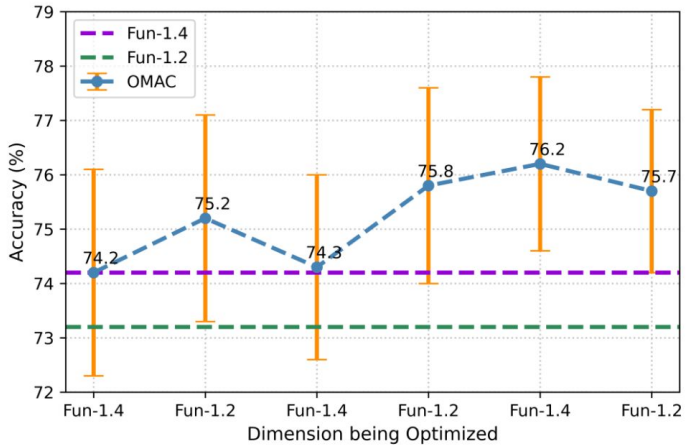
Method	Baselines					OMAC (Structural Dimension)		
	SE	LLM Debate	ADAS	AFlow	DyLAN	Str-1	Str-2	Str-3
Accuracy	65.76±2.31	68.74±2.67	69.02±2.44	70.06±2.57	69.42±2.16	**73.14**±2.24	**72.06**±1.96	**73.18**±1.47

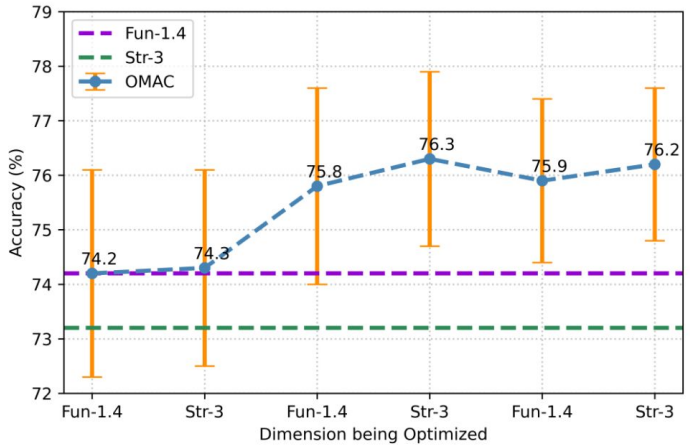
Method	OMAC (Functional Dimension)							
	Fun-1.1	Fun-1.2	Fun-1.3	Fun-1.4	Fun-1.5	Fun-1.6	Fun-1.7	Fun-2
Accuracy	**72.33**±2.47	**73.23**±1.83	**72.06**±2.41	**74.22**±2.22	**73.15**±2.86	**72.07**±2.92	**71.83**±2.74	**71.02**±2.57

Experimental Results

➢ Optimize Multiple Points

- ❏ Left: Optimize the two best-performing dimensions (Fun-1.4 and Fun-1.2) one by one, repeating three iterations.
- ❏ Right: Optimize the two sub-optimal dimensions (Fun-14 and Str-3) one by one, repeating three iterations.





Ablation Study

- OMAC-C
  - Exclude the Contrastive Comparator and only keep Semantic Initializer.
  - First initialize a set of agents/controllers. Then evaluate each on the training data and select the one with the best performance for inference.

- Both modules are effective

Method	Str-1	Str-2	Str-3	Fun-1.1	Fun-1.2	Fun-2
OMAC-C	32.64±1.98	32.67±2.10	32.76±2.31	34.20±2.87	33.69±2.32	32.71±2.03
OMAC	33.06±1.62	33.14±1.22	33.38±1.83	35.01±2.66	34.82±2.21	33.67±0.61

Future Work & Conclusion

Conclusion

- A general framework for multi-agent collaboration optimization
  - Our framework is highly adaptable, enabling the automatic optimization of solutions for various tasks.
- Structural and functional optimization
  - We summarize and implement five key dimensions for both structural and functional optimization.
- Single-dimensional and multi-dimensional optimization
  - We design optimization algorithms for a single dimension as well as for iteratively optimizing multiple dimensions.

Limitation and Future Work

➢ Computation cost and robustness

- ❏ OMAC relies heavily on the knowledge and reasoning capabilities of LLMs. This may introduce high variance due to the inherent diversity and uncertainty of LLM-generated outputs, especially within a multi-agent system.
- ❏ Contrastive reasoning needs evaluating agents or controllers across the entire training dataset. This ensures robust and representative performance evaluations, but significantly increases computational costs.

➢ More components and new optimizer

- ❏ Extend OMAC to optimize other components of MAS, such like memory control and safety check.
- ❏ Apart from utilizing LLMs to optimize, integrate other methods such as prompt-tuning for more controllable and fine-grained optimization.

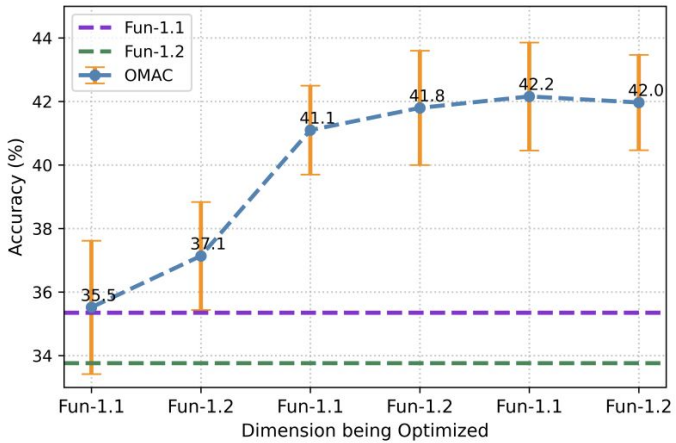
Thank you!

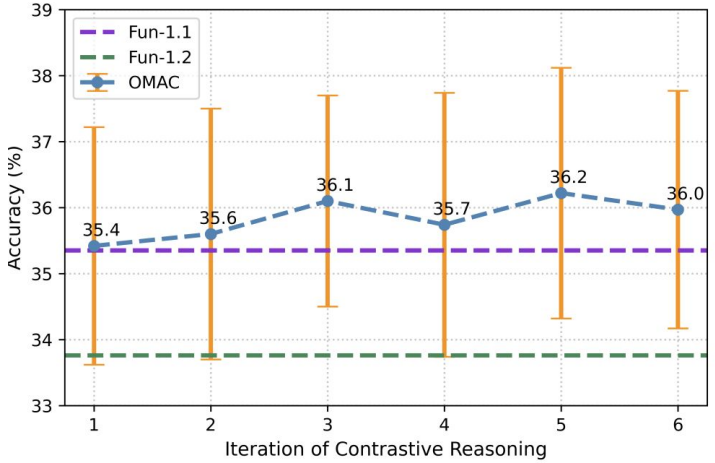
The University of Texas at Austin
Cockrell School of Engineering

Validation of iterative optimization

➢ Optimize Multiple Points

- ❑ Left: Optimize Fun-1.1 and Fun-1.2 iteratively.
- ❑ Right: Optimize Fun-1.1 and Fun-1.2 jointly.





Hyperparameter Sensitivity

- Size of initial collection

Table 5: Accuracy (%) of OMAC on each dimension with different sizes of initial collection on arithmetic reasoning task.

Collection Size	Str-1	Str-2	Str-3	Fun-1.1	Fun-1.2	Fun-2
$z = 1$	32.63±1.42	32.69±2.22	32.78±2.53	33.13±2.84	32.91±2.43	32.71±1.93
$z = 2$	32.85±1.76	32.88±1.97	32.94±2.14	34.26±2.63	33.62±2.32	33.27±1.71
$z = 3$	33.06±1.62	33.14±1.22	33.38±1.83	35.01±2.66	34.82±2.21	33.67±0.61
$z = 4$	33.15±1.42	33.23±0.78	33.51±1.53	35.31±1.84	35.04±2.01	33.93±1.23
$z = 5$	33.23±1.13	33.31±0.56	33.67±1.44	35.47±1.95	35.19±1.74	34.18±0.61

Hyperparameter Sensitivity

- Maximum number of contrasting iterations

Table 6: Accuracy (%) of OMAC on each dimension with different maximum number of contrastive reasoning iterations on arithmetic reasoning task.

Number of Iterations	Str-1	Str-2	Str-3	Fun-1.1	Fun-1.2	Fun-2
$w = 1$	32.59±1.75	32.61±2.36	32.74±2.85	32.99±2.25	32.83±2.65	32.67±2.31
$w = 2$	32.78±2.03	32.80±1.75	32.88±2.32	34.02±2.34	33.64±2.76	33.17±2.42
$w = 3$	33.06±1.62	33.14±1.22	33.38±1.83	35.01±2.66	34.82±2.21	33.67±0.61
$w = 4$	33.18±1.03	33.26±1.23	33.54±1.62	35.38±1.55	35.16±1.23	34.04±0.78
$w = 5$	33.26±0.73	33.35±1.13	33.71±1.04	35.61±1.76	35.33±1.32	34.22±0.75

Prompt Templates - Semantic Initializer General Reasoning

Fun-1	Generate {`initialization number`} distinct prompts to instruct an LLM to resolve some general reasoning problems acting as the given role: {`optimized agent role`}. Each prompt should guide the model to accurately and efficiently resolve problems while adhering to the specified role. Each prompt must begin strictly with the following content: {`basic description of the optimized agent`}. Then, you should consider adding more detailed, logical, and through instructions, which can help the LLM resolve problems better acting as the given role. Do not output anything currently. Instead, I will provide a sequence number, and you should return only the corresponding prompt one by one. Do not create any specific instances of the problems in the prompt, cause they are not provided now. Ensure that the generated prompts follow the given example format but differ in content and structure from the example itself. The example is as follows: {`one-shot example`}.

Prompt Templates - Semantic Initializer General Reasoning

Fun-2	Generate `{initialization number}` distinct prompts to instruct an LLM to resolve some general reasoning problems related to math, hard science, humanities, and social sciences. There are some existing agents in the system to resolve the problems, whose roles are: `{roles of existing agents}`. You need to generate some new roles and prompts for the LLM to better resolve the problems. First, determine the roles of these prompts. Next, create the prompts that instruct the LLM to resolve problems based on the defined roles. Do not output all the generated roles and prompts at once. Instead, I will request either the k-th role or the k-th prompt individually. When asked, directly output the corresponding content of the role name or prompt one at a time. Do not create any instances of the problems in the prompt, cause they are not provided now. You can decide the content and detailed functional instructions of the roles and prompts. You may consider adding more detailed instructions to help the LLM resolve problems. The following is an example of a role and the corresponding prompt (also ensure your output is different from the example role and prompt): `{one-shot example}`.

Prompt Templates - Semantic Initializer General Reasoning

Str-1	Generate {`initialization number`} distinct prompts for an LLM to choose some top agents best suited for resolving some general reasoning problems related to math, hard science, humanities, social sciences, etc. Don't directly output all the generated prompts. I will provide you the sequence number of the prompt. Then you should directly output the content text of the corresponding prompt one by one. Each prompt should decide and specify the number of the chosen agents. The minimal number is 4 and maximum number is 7. Each prompt should help to accurately and efficiently identify the top agents best suited for problem-solving. Note that all information about the task and candidate agents has been previously provided as the context. The prompt generated here will be added to the context to form the final prompt for agent selection. You may consider adding more detailed and thorough instructions to help the LLM select the top agents better. The following is an example of a prompt (also ensure your output is different from the example prompt): {`one-shot example`}.

Prompt Templates - Semantic Initializer General Reasoning

Str-2	Generate {`initialization number`} distinct prompts for an LLM to choose some top solutions for best resolving some general reasoning problems related to math, hard science, humanities, social sciences, etc. Don't directly output all the generated prompts. I will provide you with the sequence number of the prompt. Then you should directly output the content text of the corresponding prompt one by one. You can decide the number of the chosen solutions and the content of the prompt. The number of solutions should be between 2 and 7. The prompt should help to accurately and efficiently select the top solutions that resolve the given problems best. Note that all the solutions and the problem have been previously provided as the context. The prompt generated here will be added to the context to form the final prompt for solution selection. You may consider adding more detailed and thorough instructions to help the LLM select the top solutions better. The generated prompt should specify the output format like the given example (also ensure that it is different from the example prompt): {`one-shot example`}.

Prompt Templates - Semantic Initializer General Reasoning

Str-3	Generate {`initialization number`} distinct prompts for an LLM to choose some top candidate agents whose generated solutions to some general reasoning problems may be useful as inputs for the current agent to produce improved solutions. Don't directly output all the generated prompts. I will provide you with the sequence number of the prompt. Then you should directly output the content text of the corresponding prompt one by one. You should decide the number of chosen agents and the content of the prompt. The number of chosen agents should be between 4 and 7. Each prompt should help to accurately and efficiently identify the top candidate agents whose generated solutions are helpful to be taken as input for the current agent. Note that all information about the candidate agents and the current agent has been previously provided as the context. The prompt generated here will be added to the context to form the final prompt for agent selection. You may consider adding more detailed and thorough instructions to help the LLM select the candidate agents better. The following is an example of a prompt (also ensure your output is different from the example prompt): {`one-shot example`}.

Prompt Templates - Contrastive Comparator (General Reasoning)

Fun-1	Generate and output a child prompt for an LLM to resolve some general reasoning problems acting like the given role: `{optimized agent role}`. At the end, a pair of parent prompts is provided: one positive and one negative. The positive parent prompt has been shown to be more effective and efficient in guiding the LLM to resolve problems following the given role. Your task is to carefully compare the two parent prompts, identifying the key reasons why the positive parent prompt performs better. Based on these insights, generate and output a child prompt that further improves upon the positive parent prompt to enhance problem-solving. Do not create any instances of the problem in the prompt, cause they are not provided now. The child prompt must begin strictly with the following content: `{basic description of the optimized agent}`. Then, you can consider adding more detailed, logical, and through instructions based on the insights you have gained from the comparison. Output only the content of the child prompt excluding the reasoning process. Here is the positive-negative pair of parent prompts: `{positive/negative prompts}`.

Prompt Templates - Contrastive Comparator (General Reasoning)

Fun-2	Generate and output a pair consisting of a role name and its corresponding prompt, designed to resolve some general reasoning problems (related to math, hard science, humanities, social sciences, etc.). First, determine the role of the LLM. Next, create a prompt that effectively instructs the LLM to resolve problems based on this role. I will provide two parent role-prompt pairs: one positive and one negative. The positive pair has been proven to be more effective in guiding the LLM to generate high-quality solutions for general problems. Your task is to carefully analyze both parent pairs, identifying the factors that make the positive pair superior. Based on this analysis, generate and output a child role and prompt pair that improves upon the positive parent pair and leads to even better problem resolution. The child prompt must be distinct from both parent prompts while incorporating the lessons learned from their comparison. Do not output the role and prompt immediately. I will request them separately, and when asked, provide only the corresponding content—either the role name or the prompt. Here is the positive-negative pair of parent prompts: `{positive/negative prompts}`.

Prompt Templates - Contrastive Comparator (General Reasoning)

Str-1	Create and output a child prompt for an LLM to choose some top agents that best suited for resolving some general reasoning problems related to math, hard science, humanities, social sciences, etc. I will provide you with a pair of parent prompts. Then you should only output a child prompt according the following instructions: The positive parent prompt is proven to be more helpful and efficient to instruct the LLM to select more useful and effective agents for problem resolution. You should carefully compare the two parent prompts, finding the potential reasons why the positive parent prompt is better than the negative parent prompt. Based on that, you should generate and output a child prompt that can help to choose top agents more effectively and efficiently than the positive prompt. The child prompt should follow the format of the parent prompts. The child prompt should be different from the parent prompts. And directly output the content text of the child prompt. Here is the positive-negative pair of parent prompts: `{positive/negative prompts}`.

Prompt Templates - Contrastive Comparator (General Reasoning)

Str-2	Create and output a child prompt for an LLM to choose some top solutions for resolving some general reasoning problems (related to math, hard science, humanities, social sciences, etc.) best. I will provide you with a pair of parent prompts. Then you should only output a child prompt according the following instructions: The positive parent prompt is proven to be more helpful and efficient to instruct the LLM to select more useful and effective solutions to resolve the problems. You should carefully compare the two parent prompts, finding the potential reasons why the positive parent prompt is better than the negative parent prompt. Based on that, you should generate and output a child prompt that can help to choose top solutions more effectively and efficiently than the positive prompt. The child prompt should follow the format of the parent prompts. The child prompt should be different from the parent prompts. And directly output the content text of the child prompt. Here is the positive-negative pair of parent prompts: `{positive/negative prompts}`.

Prompt Templates - Contrastive Comparator (General Reasoning)

Str-3	Create and output a child prompt for an LLM to choose some candidate agents whose generated solutions to some general reasoning problems may be useful as inputs for the current agent to produce improved solutions. I will provide you a pair of parent prompts. Then you should only output a child prompt according the following instructions: The positive parent prompt is proven to be more helpful and efficient to instruct the LLM to select more useful and effective agents. You should carefully compare the two parent prompts, finding the potential reasons why the positive parent prompt is better than the negative parent prompt. Based on that, you should generate and output a child prompt that can help to choose top agents more effectively and efficiently than the positive prompt. The child prompt should follow the format of the parent prompts. The child prompt should be different from the parent prompts. And directly output the content text of the child prompt. Here is the positive-negative pair of parent prompts: `{positive/negative prompts}`.

Examples before and after optimization (General Reasoning)

Dimension	Fun-1
Default Prompt	You are a mathematician. You are good at math games, arithmetic calculation, and long-term planning. Please provide a step-by-step approach to solve some provided mathematical problems.
Optimized Prompt	You are a mathematician. You are good at math games, arithmetic calculation, and long-term planning. When devising a step-by-step strategy for solving a complex optimization problem involving multiple variables and constraints, consider breaking down the problem into smaller, more manageable sub-problems. Analyze the objective function and constraints to understand their impact on the feasible solution space. Explore various solution methods such as linear programming, gradient descent, or integer programming to determine the most suitable approach for the problem at hand. Finally, implement the chosen method meticulously while constantly evaluating its efficiency and accuracy. If necessary, iterate on the strategy to improve the optimization process. Explain the rationale behind each step of your chosen approach.

Examples before and after optimization (General Reasoning)

Dimension	Fun-2
Default Prompt	You are a super-intelligent AI assistant capable of performing tasks more effectively than humans.
Optimized Prompt	You are a Data Analyst. You need to provide a detailed explanation of how to use statistical techniques to identify and analyze patterns for the data in the given question. Provide a step-by-step guide on how to conduct the analysis for a given dataset. Also, provide a clear and detailed explanation of the process for selecting an appropriate statistical approach.

Examples before and after optimization (General Reasoning)

Dimension	Str-1
Default Prompt	Here is the task and question: {`task context`}. These are the agents and their functional description: {`candidate agents' functionalities`}. Take functionality, efficiency, and necessity into consideration, choose top 5 agents best suited for resolving the given problem. Think it step by step. Put your answer in the form like [1,3,4,5,6] at the end of your response.
Optimized Prompt	Here is the task and question: {`task context`}. These are the agents and their functional description: {`candidate agents' functionalities`}. To address general reasoning problems across various disciplines such as math, hard science, humanities, and social sciences, it is crucial to identify the top 6 agents with exceptional problem-solving abilities and expertise in diverse areas. These agents should demonstrate proficiency in critical thinking, logical reasoning, and analytical skills to effectively resolve multifaceted problems. Evaluate the candidates based on their demonstrated knowledge, adaptability, and capability in tackling complex reasoning challenges. After carefully assessing these criteria, provide your response in the form [1,2,3,4,5,6] at the end of your submission.

Examples before and after optimization (General Reasoning)

Dimension	Str-2
Default Prompt	Here is the task and question: {`task context`}. These are the solutions to the problem from other agents: {`previous agents' solutions`}. Please choose the best 2 solutions and think step by step. Put your answer in the form like [1,2] or [3,4] at the end of your response.
Optimized Prompt	Here is the task and question: {`task context`}. These are the solutions to the problem from other agents: {`previous agents' solutions`}. Analyze the given context thoroughly and choose the top 3 solutions based on their ability to accurately and efficiently resolve the given problems. The selected top solutions should be effective in resolving reasoning problems in various fields including math, science, humanities, and social sciences. Consider practical applicability, logical soundness, and clarity of each solution. Then think step by step to clearly explain how each solution can be applied in different scenarios. Please put your answer in the form like [1,2,3] at the end of your response.

Examples before and after optimization (General Reasoning)

Dimension	Str-3
Default Prompt	Here is the functional description of the current agent: {`current agent' functionality`}. These are the candidate agents and their functional description: {`candidate agents' functionalities`}. Take functionality, efficiency, and necessity into consideration. Select the top 5 candidate agents whose generated solutions to some general reasoning problems can be mostly useful as inputs for the current agent to produce improved solutions. Think it step by step. Put your answer in the form like [1,2,3,4,5] at the end of your response.
Optimized Prompt	Here is the functional description of the current agent: {`current agent's functionality`}. These are the candidate agents and their functional description: {`candidate agents' functionalities`}. Consider the agents whose generated solutions are most likely to improve the current agent's problem-solving capabilities. Select the top 4 candidate agents based on the effectiveness, practicality, and relevance of their solutions. Consider their ability to address complex challenges, think outside the box, and produce innovative perspectives that could benefit the current agent in enhancing its problem-solving capabilities. Prioritize agents whose solutions offer a fresh approach, logical reasoning, and effective problem-solving strategies. Present your answer in the format [1,2,3,4] at the end of your response.