Multi-Agent Design:

Optimizing Agents with Better Prompts and Topologies

Han Zhou, Xingchen Wan, Ruoxi Sun, Hamid Palangi, Shariq Iqbal, Ivan Vulić, Anna Korhonen, and Sercan Ö. Arık

Presenters: Han Zhou

Mar, 2026

Google Research

ICLR
International Conference On Learning Representations

Agenda

01 Introduction

02 Analysis

03 Multi-Agent System Search

04 Experiments

05 Results

06 Conclusion

Google Research

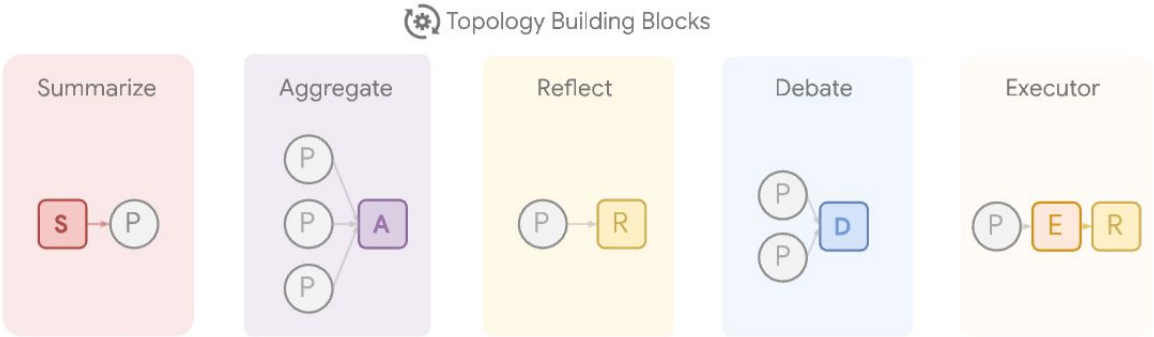
01

Introduction

Multi-Agent System (MAS)

What components form a MAS in the era of LLMs?

- Role of agents
  - Prompts that declare the functionality of agents.
- Interaction across agents
  - Topologies that orchestrate the interaction across agents.



Google Research

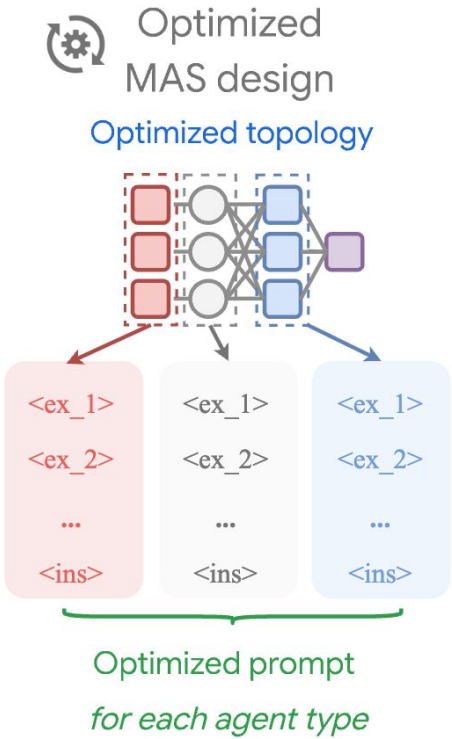
Automating the Design of MAS

Can we automate the design process?

- The problem complexity grows exponentially!
- What component matters most?
  - **Prompt** vs. **Topology**

Existing efforts in automatic agent design:

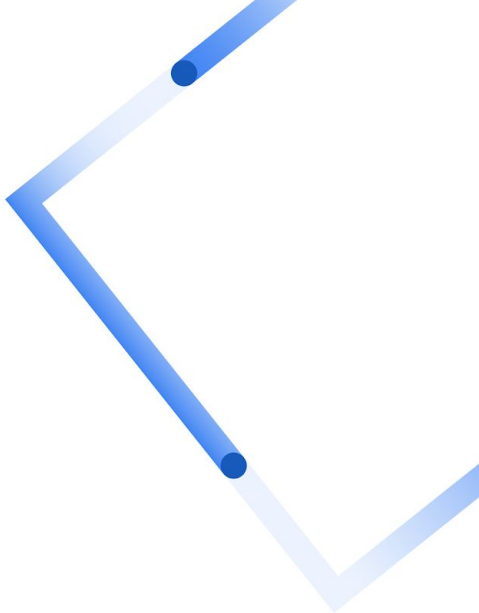
- DSPy [Oct 2023]: automatic prompt design
- GPTSwarm [Feb 2024]: graph optimization
- ADAS [Aug 2024]: LLMs as optimizers for agent design
- AFlow [Oct 2024]: MCTS for agentic operator search



Google Research

02

Analysis



Google Research

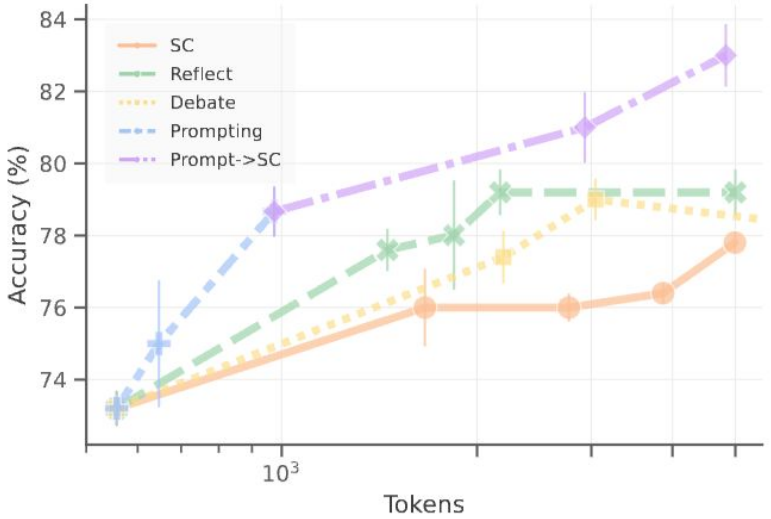
Prompt Design for Agents

Automatic prompt design (e.g., APO) optimizes both instructions and exemplars for agents.

The effect of PO compared to common operators in MAS:

- Multi-agent operators saturate quickly.
- PO leads to better token-effectiveness.
- `Prompt -> SC' shows the best scaling.

Design principle: *optimize agents locally before scaling their topology.*



Accuracy vs. the total token counts for prompt-optimized agents per question on MATH by Gemini 1.5 Pro compared to scaling agents with self-consistency (SC), self-refine (reflect), and multi-agent debate (debate).

Google Research

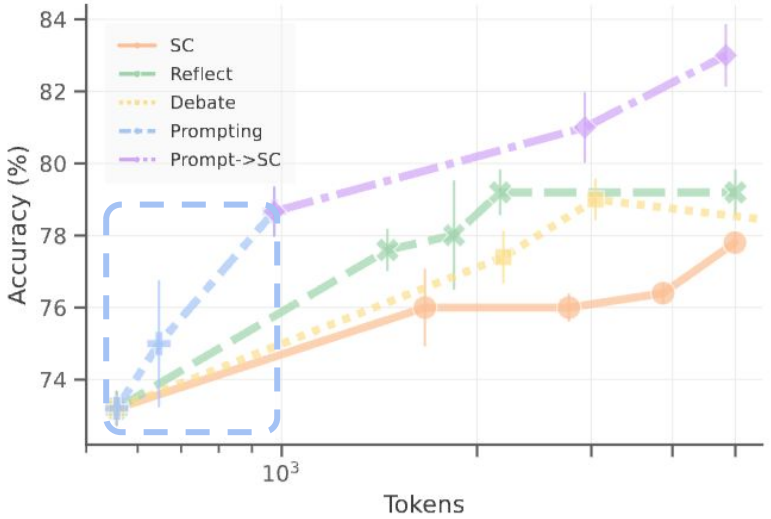
Prompt Design for Agents

Automatic prompt design (e.g., APO) optimizes both instructions and exemplars for agents.

The effect of PO compared to common operators in MAS:

- Multi-agent operators saturate quickly.
- PO leads to better token-effectiveness.
- `Prompt -> SC' shows the best scaling.

Design principle: *optimize agents locally before scaling their topology.*



Accuracy vs. the total token counts for prompt-optimized agents per question on MATH by Gemini 1.5 Pro compared to scaling agents with self-consistency (SC), self-refine (reflect), and multi-agent debate (debate).

Google Research

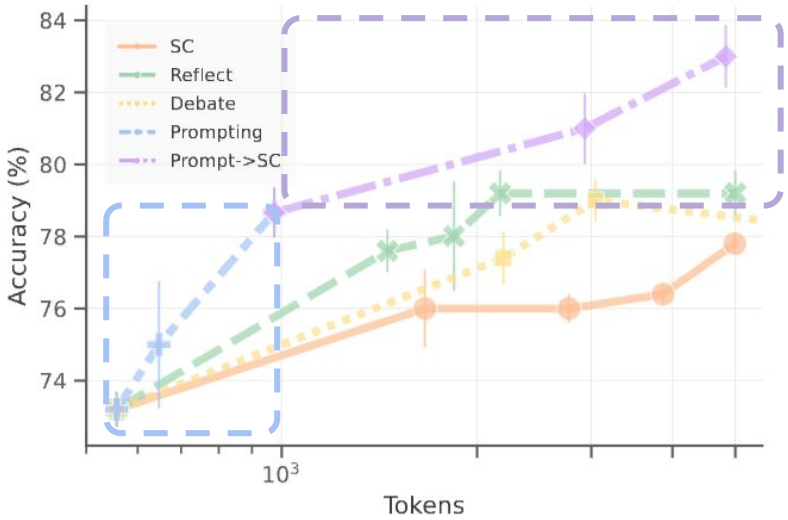
Prompt Design for Agents

Automatic prompt design (e.g., APO) optimizes both instructions and exemplars for agents.

The effect of PO compared to common operators in MAS:

- Multi-agent operators saturate quickly.
- PO leads to better token-effectiveness.
- `Prompt -> SC' shows the best scaling.

Design principle: *optimize agents locally before scaling their topology.*



Accuracy vs. the total token counts for prompt-optimized agents per question on MATH by Gemini 1.5 Pro compared to scaling agents with self-consistency (SC), self-refine (reflect), and multi-agent debate (debate).

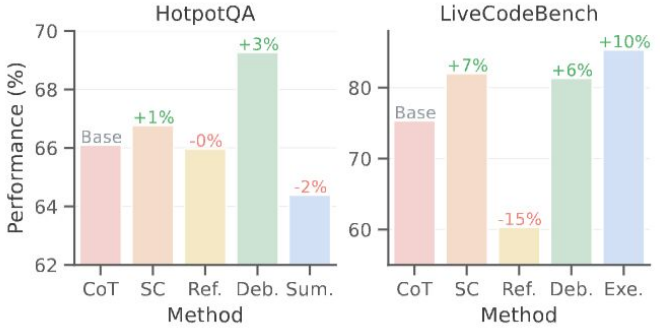
Google Research

Topology Design Space

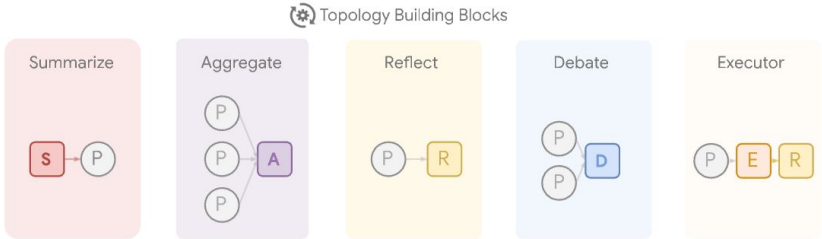
Representative Blocks:

- *Aggregate*: majority vote, self-consistency.
- *Reflect*: self-refine, reflexion.
- *Debate*: multi-agent debate.
- *Custom Agents*: user-defined agents.
- *Tool-use*: plugable tool-use operations.

Observation: *not all topologies exert a positive influence on the MAS design.*



The performance of different topologies with Gemini 1.5 Pro compared to the base agent with each topology being optimized with APO



Google Research

03

Multi-Agent System Search

Google Research

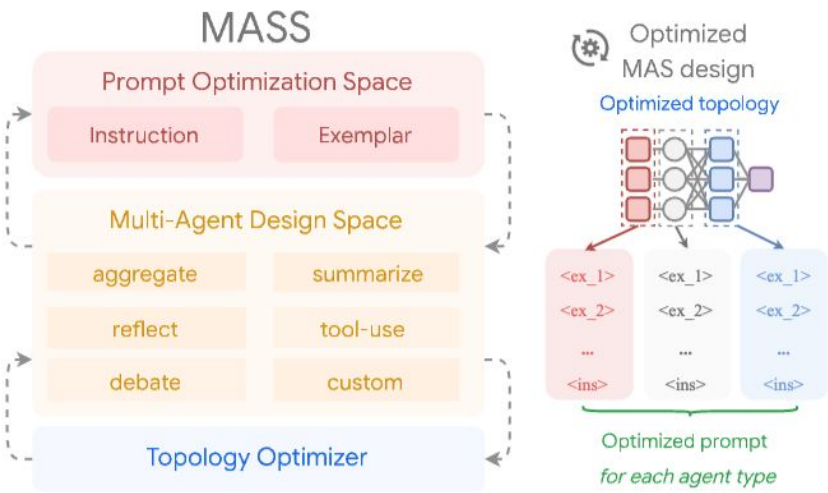
MASS: Multi-Agent System Search

MASS involves interleaved optimization stages,

from ***local*** to ***global***, from ***prompts*** to ***topologies***,

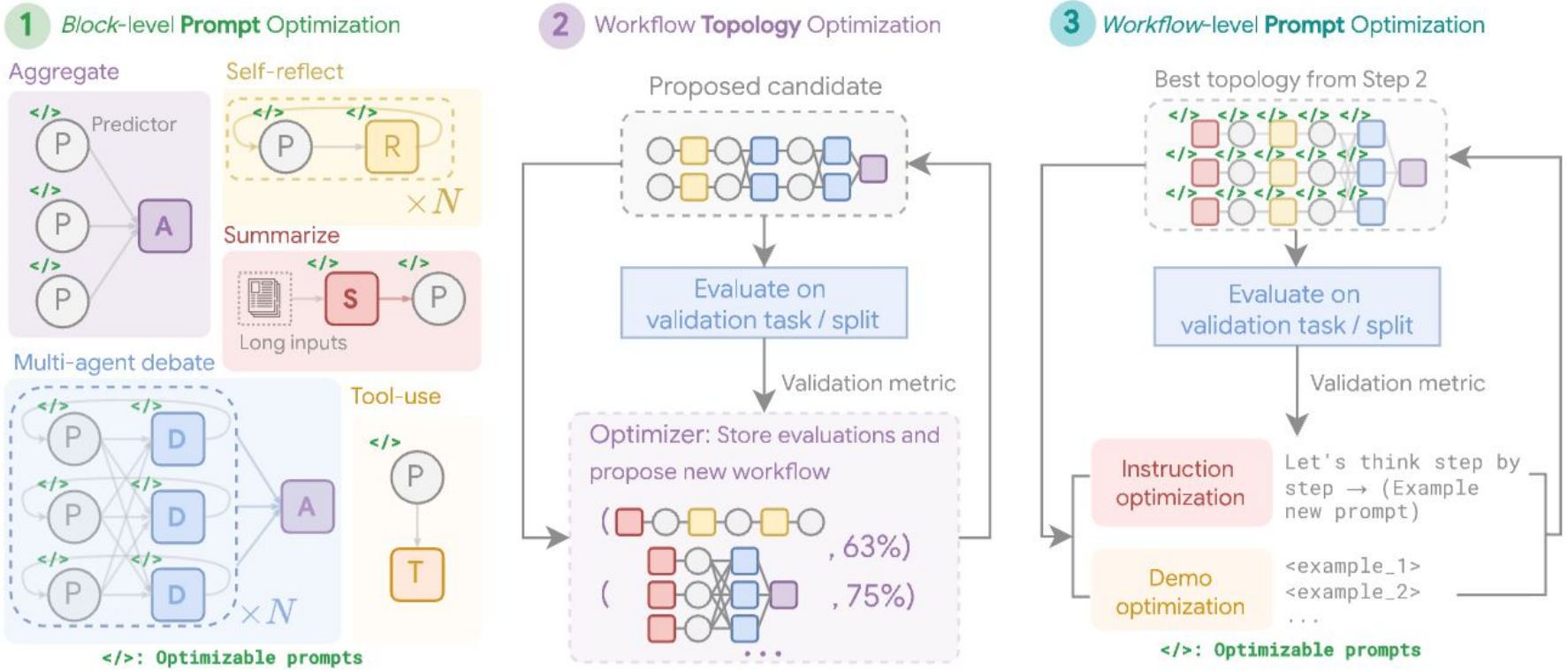
over three stages:

1. *Block-level* **prompt** optimization.
2. Workflow **topology** optimization
3. *Workflow*-level **prompt** optimization



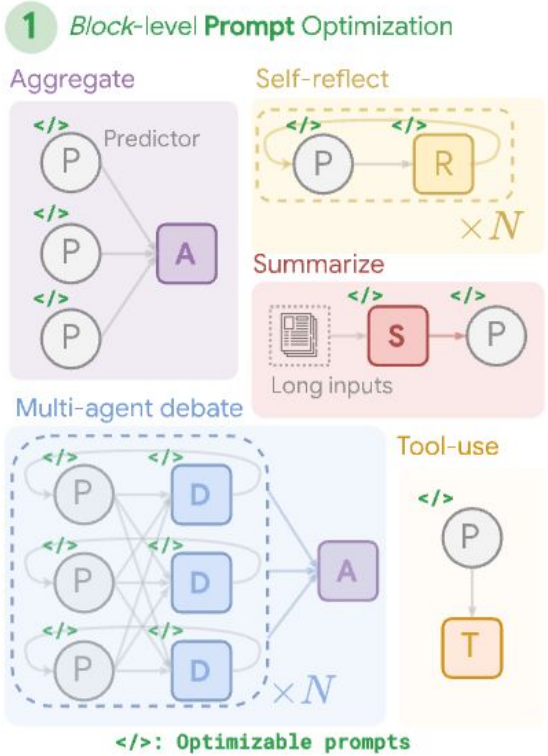
Google Research

MASS: Multi-Agent System Search



Google Research

MASS: Multi-Agent System Search



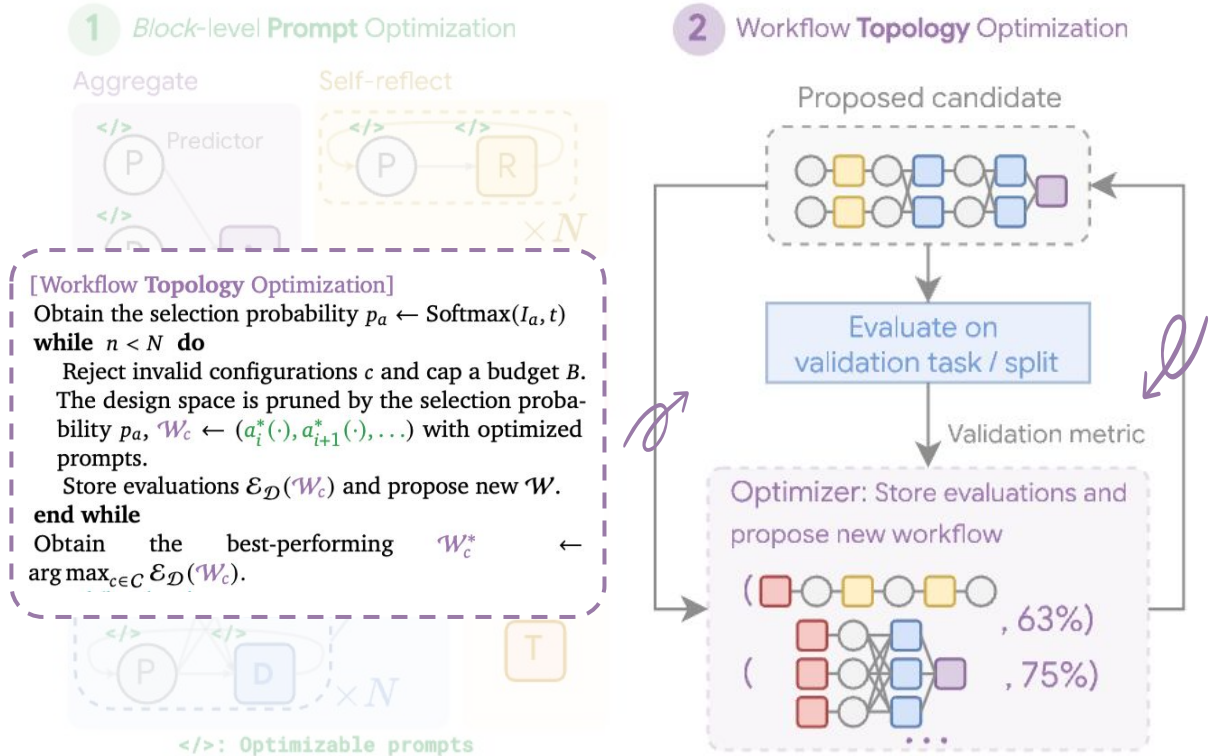
[*Block*-level **Prompt** Optimization]
Prompt optimization for the initial agent $a_0^* \leftarrow \mathcal{O}_{\mathcal{D}}(a_0)$.
for a_i in $\mathcal{A} \setminus \{a_0\}$ **do**
  Local prompt optimization for each building block in the design space: $a_i^* \leftarrow \mathcal{O}_{\mathcal{D}}(a_i|a_0^*)$
  Obtain incremental Influence $I_{a_i} \leftarrow \mathcal{E}(a_i^*)/\mathcal{E}(a_0^*)$.
end for

Stage (1) serves as the ***warm-up*** stage per building block:

- Each agent is primed for its role.
- Minimum topology for saving the cost.
- Guarantee an effective space for follow-up search, preventing the *compounding* impact from any ill-formed agents.

Google Research

MASS: Multi-Agent System Search



Stage (2) optimizes the topology in a influence-weighted pruned space:

- Beneficial topologies only represent a *small* fraction of the full design space.
- Including decremental blocks may result in higher complexity and degraded performance.
- We propose a selection probability for each search dimension weighted by the influence of the building blocks.

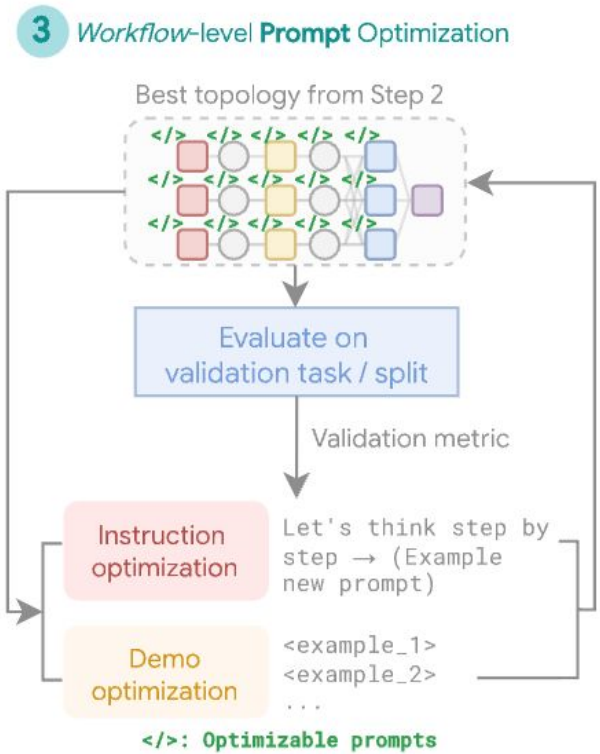
Google Research

MASS: Multi-Agent System Search

[*Workflow*-level **Prompt** Optimization]
Workflow-level prompt optimization for the best-performing topology: $\mathcal{W}^* \leftarrow O_{\mathcal{D}}(\mathcal{W}_c^*)$.
Return optimized multi-agent system $\mathcal{W}^*$.

Stage (3) serves as the ***fine-adaptation*** stage for workflow-level prompt design:

- Conditioned on the best-found topology, we optimize the prompts further, modelling the interdependence across agents.
- We show this stage often yields practical improvements.



Google Research

04

Experiments

Google Research

Experiments

Models

- Gemini-1.5-pro-002, Gemini-1.5-flash-002, Claude-3.5-Sonnet, Mistral-Nemo-12B

Tasks

- Reasoning: MATH, DROP.
- Long-context multi-hop understanding: HotpotQA, MuSiQue, 2WikiMQA (Longbench)
- Coding: MBPP, HumanEval, LiveCodeBench-test output prediction

Baselines

- Single agent: Chain-of-Thought (CoT)
- Manually crafted: Self-Consistency (SC), Self-Refine, Multi-Agent Debate
- Automatic agent design: ADAS, AFlow

05

Results

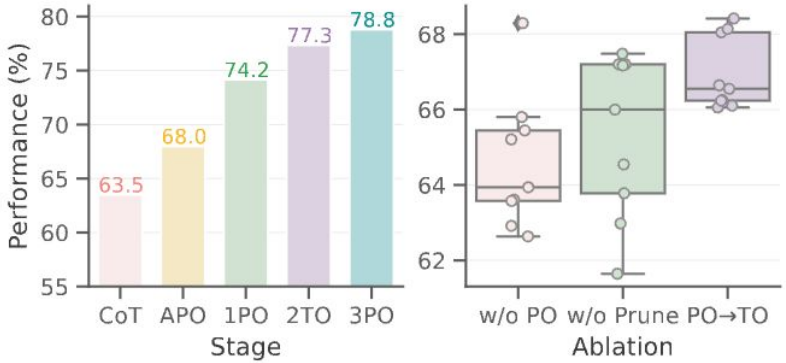
Main results

Gemini-1.5-pro-002									
Task	**Reasoning**		**Multi-hop Long-context**			**Coding**			
Method	MATH	DROP	HotpotQA	MuSiQue	2WikiMQA	MBPP	HumanEval	LCB	Avg.
CoT	$71.67_{3.30}$	$70.59_{1.67}$	$57.43_{0.52}$	$37.81_{1.43}$	$63.39_{1.12}$	$68.33_{0.47}$	$86.67_{0.94}$	$66.33_{0.62}$	65.28
Self-Consistency	$77.33_{1.25}$	$74.06_{0.90}$	$58.60_{2.19}$	$41.81_{1.00}$	$67.79_{1.19}$	$69.50_{0.71}$	$86.00_{0.82}$	$70.33_{0.94}$	68.18
Self-Refine	$79.67_{2.36}$	$71.03_{1.31}$	$60.62_{3.33}$	$42.15_{1.34}$	$66.74_{2.43}$	$63.67_{0.24}$	$84.00_{1.63}$	$67.33_{1.31}$	66.90
Multi-Agent Debate	$78.67_{0.94}$	$71.78_{0.71}$	$64.87_{0.23}$	$46.00_{0.80}$	$71.78_{0.63}$	$68.67_{0.85}$	$86.67_{1.25}$	$73.67_{1.65}$	70.26
ADAS	$80.00_{0.82}$	$72.96_{0.90}$	$65.88_{1.29}$	$41.95_{1.24}$	$71.14_{0.66}$	$73.00_{1.08}$	$87.67_{1.70}$	$65.17_{1.25}$	69.72
AFlow*	$76.00_{0.82}$	$88.92_{0.63}$	$68.62_{0.47}$	$32.05_{1.29}$	**$76.51_{1.05}$**	-	$88.00_{0.00}$	-	-
MASS (Ours)	**$84.67_{0.47}$**	**$90.52_{0.64}$**	**$69.91_{1.11}$**	**$51.40_{0.42}$**	$73.34_{0.67}$	**$86.50_{0.41}$**	**$91.67_{0.47}$**	**$82.33_{0.85}$**	**78.79**
Gemini-1.5-flash-002									
CoT	$66.67_{2.36}$	$71.79_{0.69}$	$57.82_{1.10}$	$37.10_{1.35}$	$63.40_{0.68}$	$63.33_{1.25}$	$75.67_{1.89}$	$51.17_{0.24}$	60.87
Self-Consistency	$69.33_{1.25}$	$73.42_{0.19}$	$60.19_{1.01}$	$41.94_{0.93}$	$67.98_{0.72}$	$63.67_{0.62}$	$77.67_{1.89}$	$53.83_{1.18}$	63.50
Self-Refine	$71.33_{0.94}$	$73.71_{1.09}$	$58.84_{3.04}$	$41.21_{1.99}$	$65.56_{1.57}$	$63.33_{1.25}$	$81.67_{1.89}$	$52.00_{1.41}$	63.46
Multi-Agent Debate	$71.67_{0.94}$	$74.79_{0.87}$	$64.17_{1.69}$	$46.27_{1.33}$	$72.19_{0.54}$	$63.00_{0.71}$	$79.67_{1.25}$	$55.50_{0.41}$	65.91
ADAS	$68.00_{1.41}$	$75.95_{1.18}$	$61.36_{2.89}$	**$48.81_{1.03}$**	$66.90_{1.00}$	$65.83_{0.24}$	$80.67_{2.49}$	$50.50_{1.63}$	64.75
MASS (Ours)	**$81.00_{2.45}$**	**$91.68_{0.14}$**	**$66.53_{0.38}$**	$43.67_{1.21}$	**$76.69_{0.50}$**	**$78.00_{0.82}$**	**$84.67_{0.47}$**	**$72.17_{0.85}$**	**74.30**

Ablation

Left: ablating over MASS optimization stages:

- 1PO: MAS benefits substantially from optimizing agents within the building blocks.
- 2TO: Composing influential topologies yield further improvements.
- 3PO: Modelling the interdependence of agents is beneficial



Left: *average performance per optimization stage of Mass over 8 evaluation tasks on Gemini 1.5 Pro. We compare Mass with a single agent (CoT) starting point as the reference and an APO baseline that optimizes over the single agent by MIPROv2.*

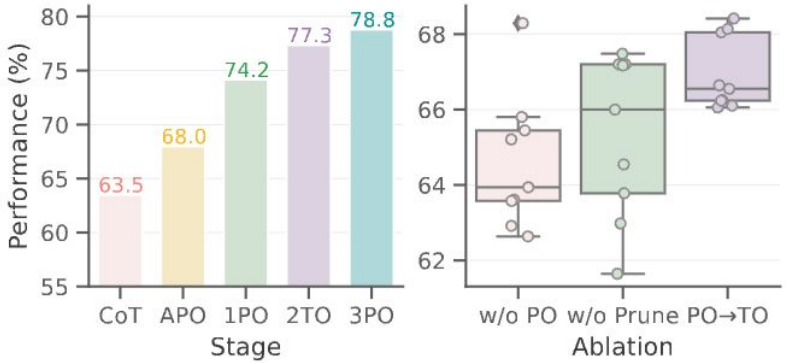
Right: *a comparative ablation study on topology optimization (2TO) without pruning and without the former stage of prompt optimization (1PO) evaluated on HotpotQA*

Google Research

Ablation

Right: the influence of the *optimization order* and the *search space pruning*

- **w/o PO**: conducting stage 2 (2TO) without 1PO beforehand results in ineffective configuration of topologies.
- **w/o Prune:** even with 1PO, without an influence-pruned design space, the variance is large, and search is inefficient.
- **PO->TO (1PO -> 2TO):** MASS exploits in the most effective design space, and discovers the optimal topology in a minimum budget.



Left: *average performance per optimization stage of Mass over 8 evaluation tasks on Gemini 1.5 Pro. We compare Mass with a single agent (CoT) starting point as the reference and an APO baseline that optimizes over the single agent by MIPROv2.*

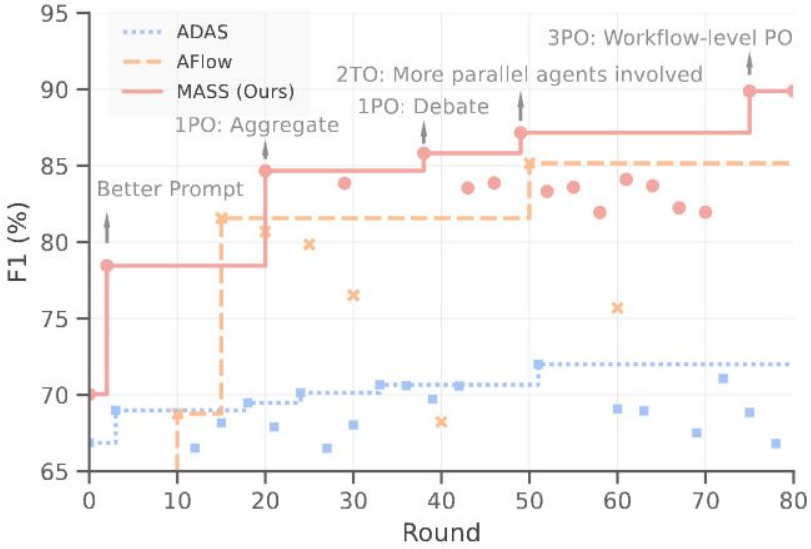
Right: *a comparative ablation study on topology optimization (2TO) without pruning and without the former stage of prompt optimization (1PO) evaluated on HotpotQA*

Google Research

Optimization trajectory

MASS demonstrates a *steady* trend of optimization:

- By interleaving the optimization stages, **MASS** efficiently searches towards better prompts and topologies.
- **ADAS** was trapped in discovering over-complex topologies in an ineffective design space.
- **AFlow** searches within a set of predefined operators. However, MCTS results in high variance and inefficient search.



The optimization trajectories of Mass compared to automatic agent design baselines per validation round on DROP. We note that, as a distinct advantage of Mass, the optimization within stages (1) & (2) of Mass can be completely parallelized, whereas ADAS and AFlow are iterative algorithms that have to wait to propose new agents until finishing earlier trajectories.

Google Research

Case Study

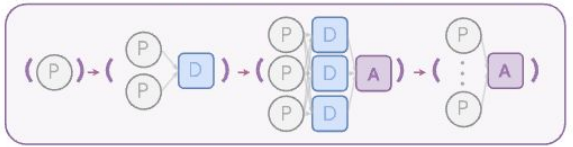
1 *Block*-level **Prompt** Optimization (62% → 79%)

```
Debator:
You are a seasoned math professor specializing in clear and concise
explanations.  You are reviewing student solutions to math problems.
Below, you will find the problem, followed by solutions from several
students.  Carefully examine each student's solution, identifying any
errors in their logic or calculations. Provide a comprehensive
rationale explaining your analysis of each student's work, clearly
stating whether their final answer is correct or incorrect and why.
Finally, provide your own definitive and simplified solution to the
problem, ensuring its accuracy and clarity. Present your final answer
bracketed between <answer> and </answer> at the end.

Question: Compute $17^{-1}\\pmod{83}$.
Solutions: Agent 0: 44\nAgent 1: 74
Rationale: <Rationale>
Answer: 44

<Task Demo: Exemplar_2>
<Task Demo: Examplar_3>
```

2 Workflow **Topology** Optimization (79% → 83%)



3 *Workflow*-level **Prompt** Optimization (83% → 85%)

```
Predictor:
Let's think step by step to solve the
given problem. Clearly explain your
reasoning process, showing all
intermediate calculations and
justifications.  Express your final
answer as a single numerical value or
simplified expression enclosed within
<answer></answer> tags.  Avoid
extraneous text or explanations outside
of the core reasoning and final answer.
<Task Demo: Exemplar_1>
```

Figure 7 | A demonstration of the optimization trajectory of MASS on MATH. In (1) block-level optimization: multi-agent debate serves as the best-performing topology. In (2) workflow **topology** optimization, aggregating with more parallel agents outweighs the performance of agents in debate. Lastly, (3) workflow-level optimization discovers the optimal prompt conditioned on the best topology.

Google Research

06

Conclusion

Multi-Agent Design Guidelines:

- Optimizing individual agents properly is important before composing them into an MAS;
- More effective MAS can be built by composing influential topologies;
- Modeling the interdependence between agents is beneficial, and can be achieved via workflow-level joint optimization.

Google Research

Thank You

Han Zhou

Student Researcher

Authors: Han Zhou, Xingchen Wan, Ruoxi Sun, Hamid Palangi, Shariq Iqbal, Ivan Vulić, Anna Korhonen, and Sercan Ö. Arık

Google Research