INTERNATIONAL UNIVERSITY OF APPLIED SCIENCES

ICNLP 2025
The 7th International Conference on Natural Language Processing

BASKHAD IDRISOV, ESTHER EISENACHER AND TIM SCHLIPPE

PROGRAM CODE GENERATION: SINGLE LLMS VS. MULTI-AGENT SYSTEMS

Guangzhou, China
March 23, 2025

AGENDA

1

INTRODUCTION

MOTIVATION: Program Code Generation Using LLMs

LLMs help
with coding

But coding solutions are
not always

- *correct*
- *efficient*
- *maintainable*

Image Sources: Image supplied by Microsoft 365. Text Sources: Bluemke & Malanowska (2021); Butt et al. (2021); Kaur et al. (2022); Zhang et al. (2023).

MOTIVATION: Multi-Agent Systems for Code Generation

First **multi-agent systems** to address limitations.

But only analyzed

- *code correctness* (pass@k)
- AI-generated code (no comparison to human-generated code)

Image Sources: Image supplied by Microsoft 365. Text Sources: Chen et al. (2021); Chen et al. (2023); Liu et al. (2023); Yetistiren et al. (2022); Yetistiren et al. (2023); Wu et al. (2024); Hong et al. (2024); Qian et al. (2024); Tao et alk. (2024).

MOTIVATION: Multi-Agent Systems for Code Generation

➔ MULTI-AGENT SYSTEMS VS SINGLE LLMS
➔ COMPARISON TO HUMAN-GENERATED CODES
➔ CORRECTNESS, EFFICIENCY, MAINTAINABILITY

First **multi-agent systems** to address limitations.

But only analyzed *correctness* (pass@k)

Image Sources: Image supplied by Microsoft 365. Text Sources: Chen et al. (2021); Chen et al. (2023); Liu et al. (2023); Yetistiren et al. (2022); Yetistiren et al. (2023); Wu et al. (2024); Hong et al. (2024); Qian et al. (2024); Tao et alk. (2024).

2

RELATED WORK

RELATED WORK

- Various studies on program code generation with LLMs have been carried out with single-agent setups.

RELATED WORK

- Various studies on program code generation with LLMs have been carried out with single-agent setups.

- Single-agent LLMs like Codex, GitHub Copilot, and ChatGPT show varying code generation performance, with ChatGPT (65.2%) and GPT-4 (76.2%) achieving the highest correctness rates.
  (Chen et al., 2021; Liu at al., 2023; Yetistiren et al., 2023)

RELATED WORK

- Various studies on program code generation with LLMs have been carried out with single-agent setups.

- Single-agent LLMs like Codex, GitHub Copilot, and ChatGPT show varying code generation performance, with ChatGPT (65.2%) and GPT-4 (76.2%) achieving the highest correctness rates.
  (Chen et al., 2021; Liu at al., 2023; Yetistiren et al., 2023)

- AutoGen enables customizable multi-agent scenarios with code execution.
  (Wu et al., 2024)

RELATED WORK

- Various studies on program code generation with LLMs have been carried out with single-agent setups.

- Single-agent LLMs like Codex, GitHub Copilot, and ChatGPT show varying code generation performance, with ChatGPT (65.2%) and GPT-4 (76.2%) achieving the highest correctness rates.
(Chen et al., 2021; Liu at al., 2023; Yetistiren et al., 2023)

- AutoGen enables customizable multi-agent scenarios with code execution.
(Wu et al., 2024)

- Multi-agent systems like MetaGPT, GameGPT, ChatDev, and MAGIS use specialized LLM agents to improve code generation, collaboration, and quality across various task.
(Chen at al., 2023; Hong et al., 2024; Qian et al., 2024; Tao et al., 2024)

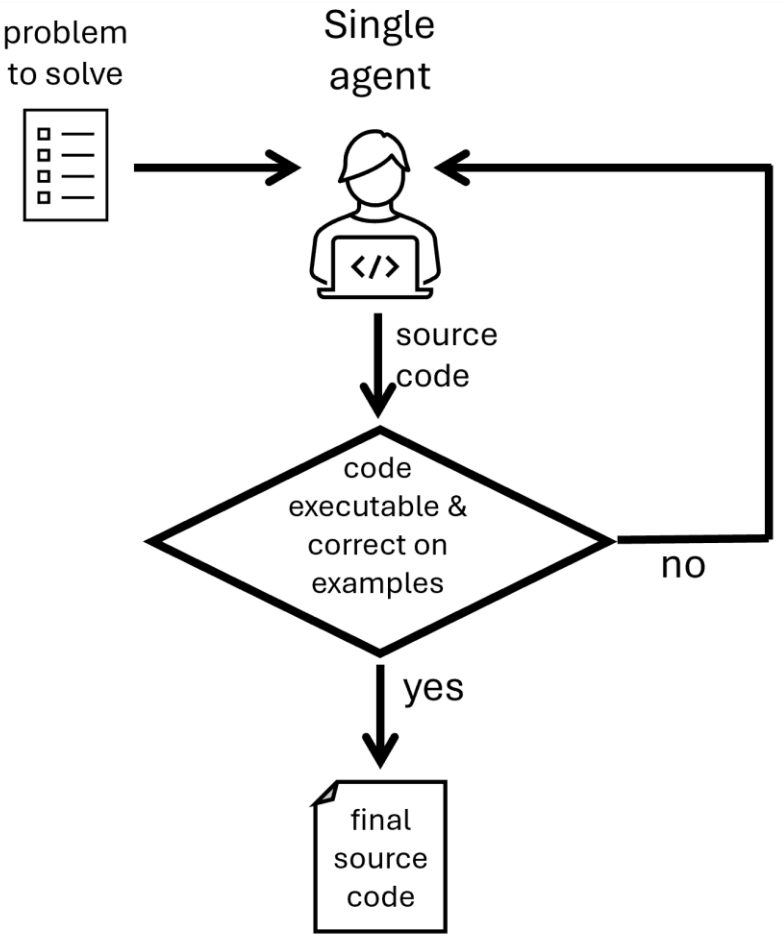
RELATED WORK

- Various studies on program code generation with LLMs have been carried out with si[illegible] agent setups.
- S[illegible] GitHub Copilot, and ChatGPT show varying code generation [illegible] GPT-4 (76.2%) achieving the highest correctness rates.
- [illegible] ution.
  (Wu et al., 2[illegible])
- Multi-agent systems like Meta[illegible] to improve code generation, collaborati[illegible]
  (Chen at al., 2023; Hong et al., 2024; Qian et al., 2024; Tao et al., 2024)

OUR APPROACH OFFERS FLEXIBLE, TASK-BASED AGENT ROLES BEYOND THE WATERFALL MODEL.

EXPERIMENTAL SETUP

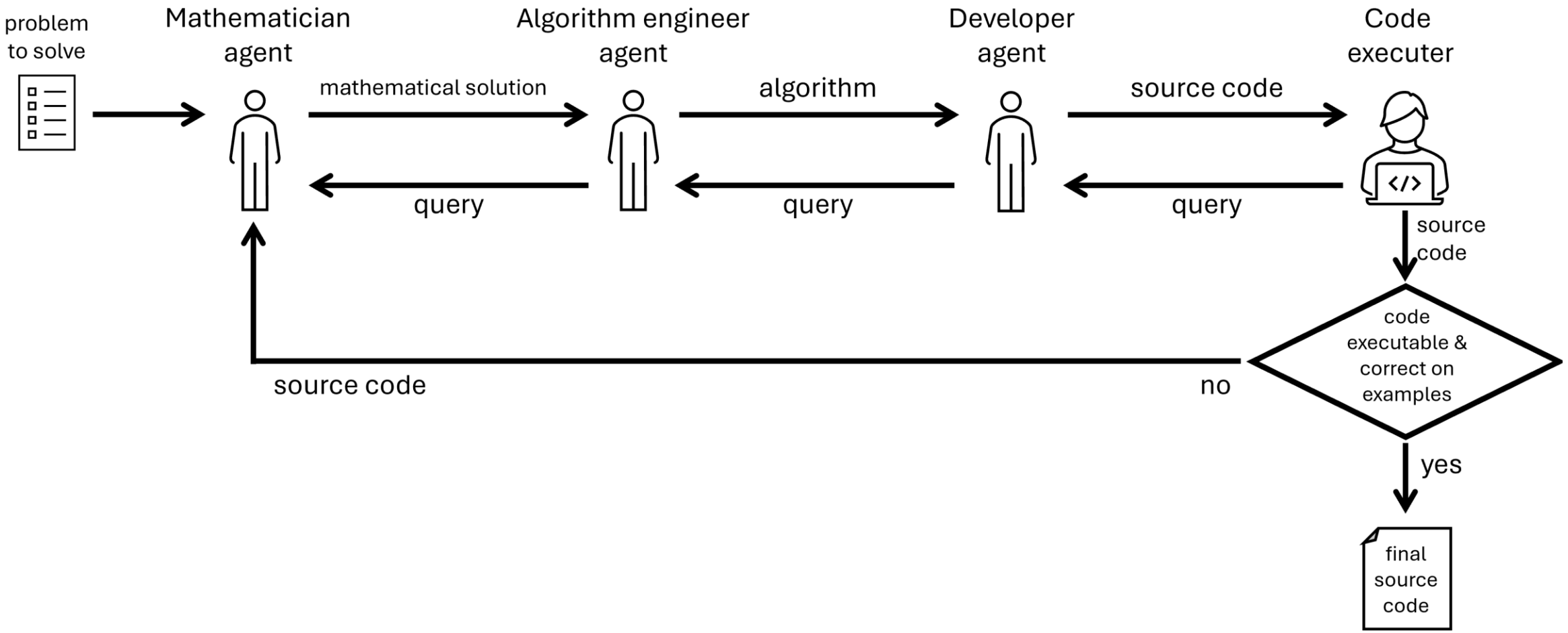
EXPERIMENTAL SETUP



SINGLE-AGENT SCENARIO

Image Source: Idrisov et al. (2025).

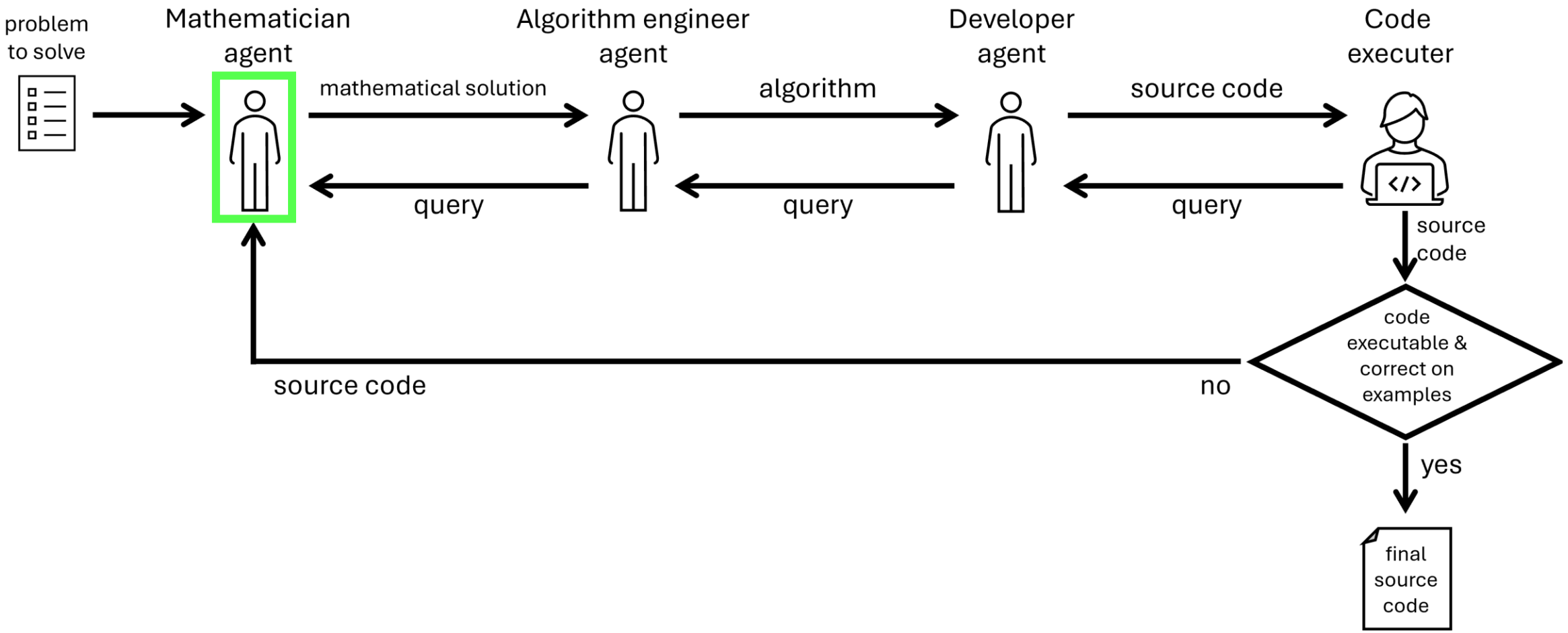
EXPERIMENTAL SETUP



MULTI-AGENT SCENARIO

Image Source: Idrisov et al. (2025).

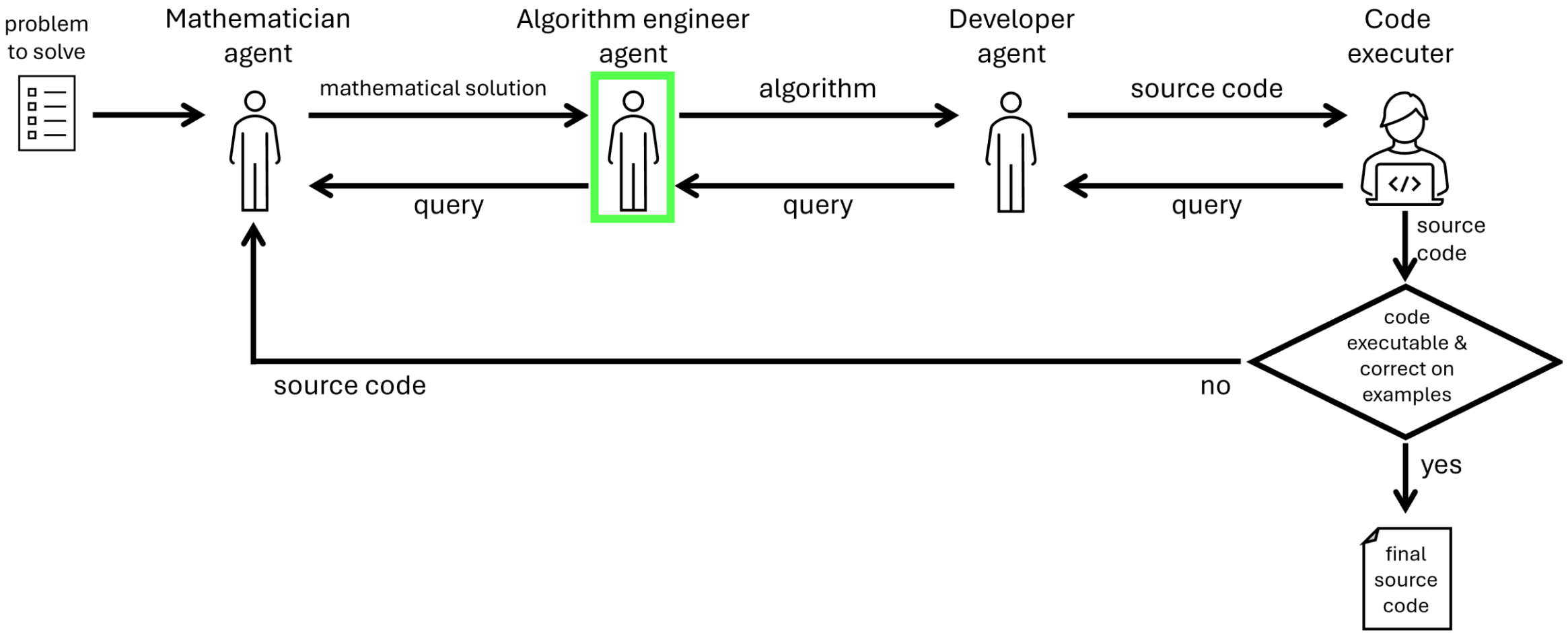
EXPERIMENTAL SETUP



MULTI-AGENT SCENARIO

Image Source: Idrisov et al. (2025).

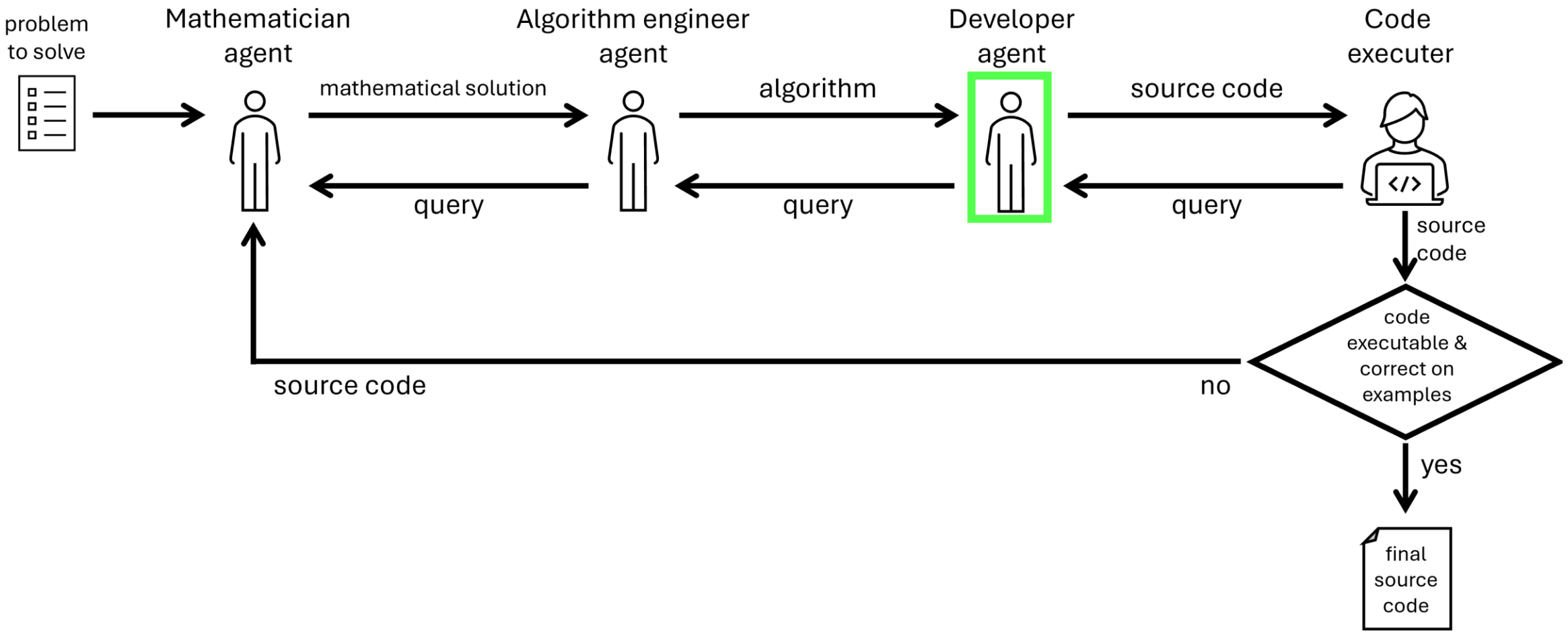
EXPERIMENTAL SETUP



MULTI-AGENT SCENARIO

Image Source: Idrisov et al. (2025).

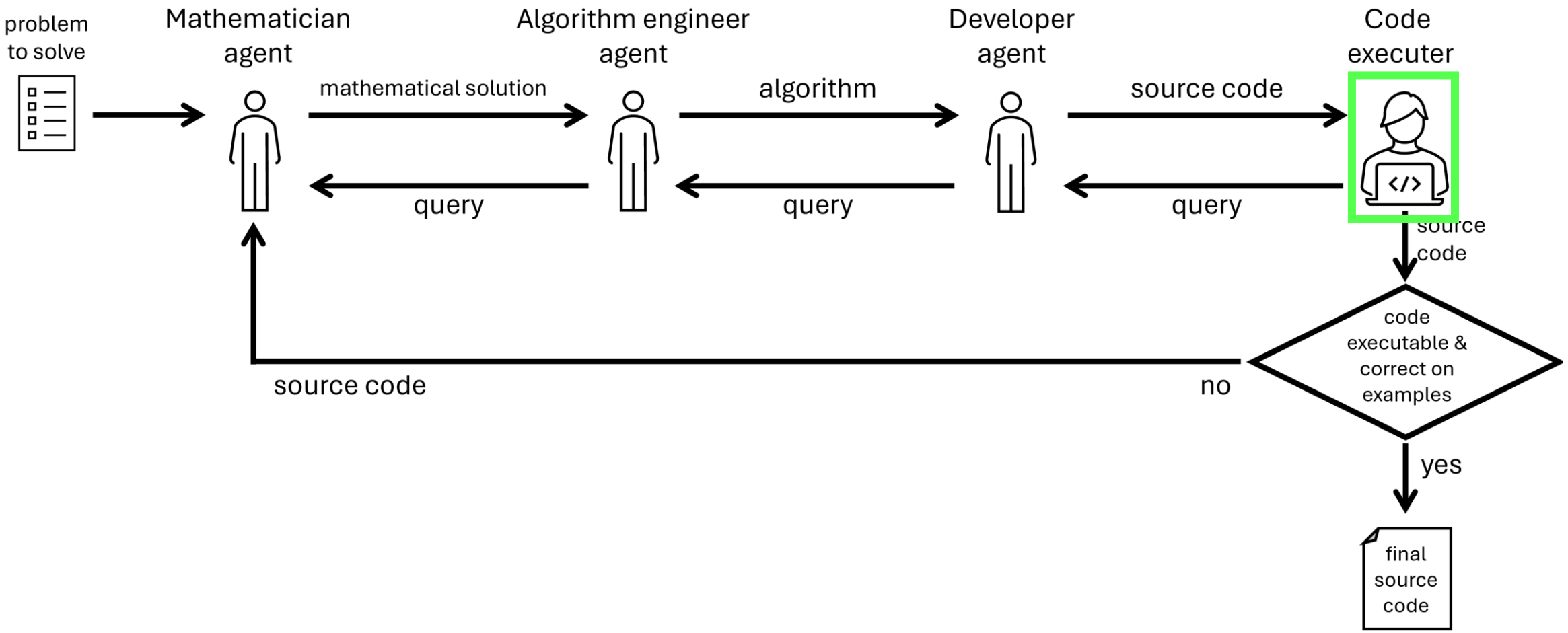
EXPERIMENTAL SETUP



MULTI-AGENT SCENARIO

Image Source: Idrisov et al. (2025).

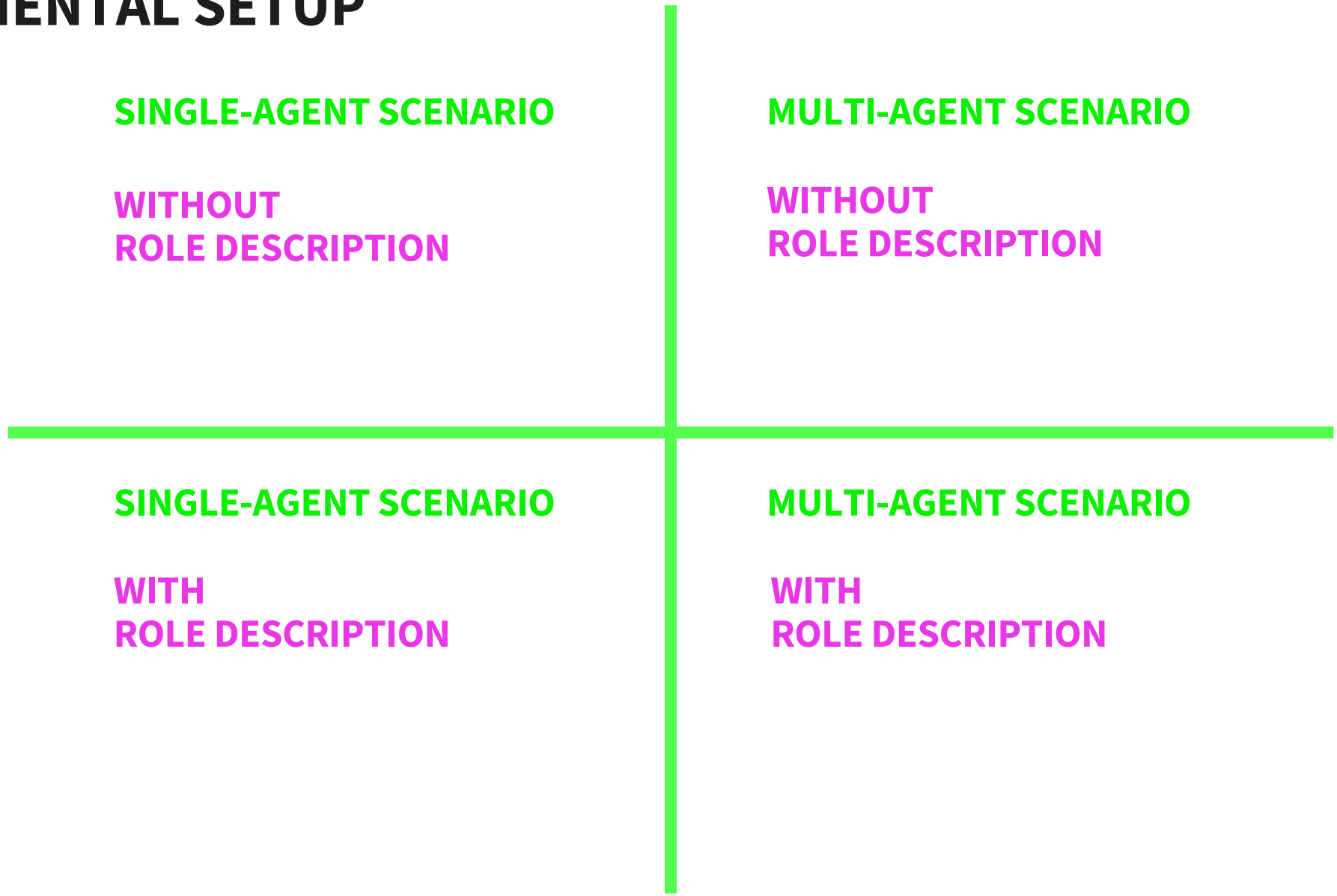
EXPERIMENTAL SETUP



MULTI-AGENT SCENARIO

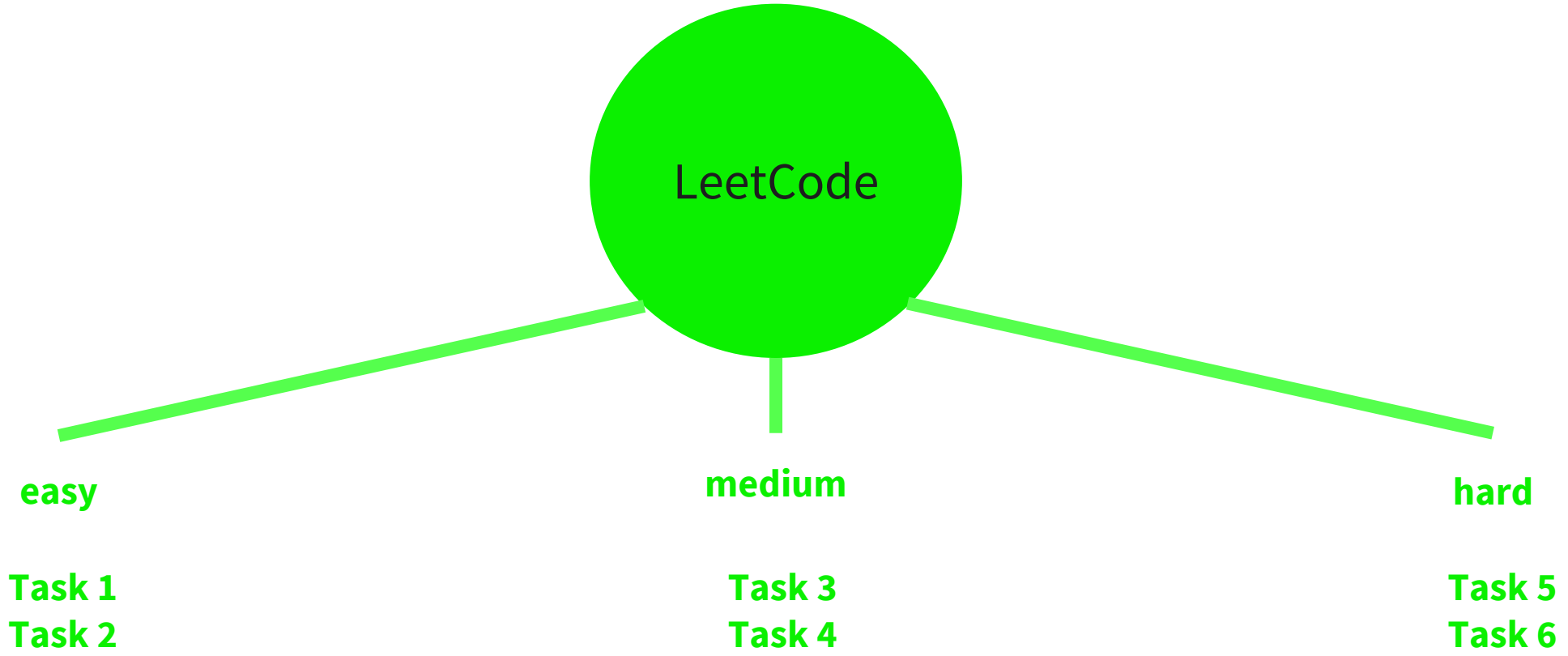
Image Source: Idrisov et al. (2025).

EXPERIMENTAL SETUP



Graphic Source: Custom Depiction.

DATA



Graphic Source: Custom Depiction.

DATA

LeetCode

2818. Apply Operations to Maximize Score

You are given an array nums of n positive integers and an integer k.
Initially, you start with a score of 1. You have to maximize your score by applying the following operation at most k times:
•Choose any **non-empty** subarray nums[l, ..., r] that you haven't chosen previously.
•Choose an element x of nums[l, ..., r] with the highest **prime score**. If multiple such elements exist, choose the one with the smallest index.
•Multiply your score by x.
Here, nums[l, ..., r] denotes the subarray of nums starting at index l and ending at the index r, both ends being inclusive.
The **prime score** of an integer x is equal to the number of distinct prime factors of x. For example, the prime score of 300 is 3 since 300 = 2 * 2 * 3 * 5 * 5.
Return *the **maximum possible score** after applying at most* k *operations*.
Since the answer may be large, return it modulo $10^9 + 7$.

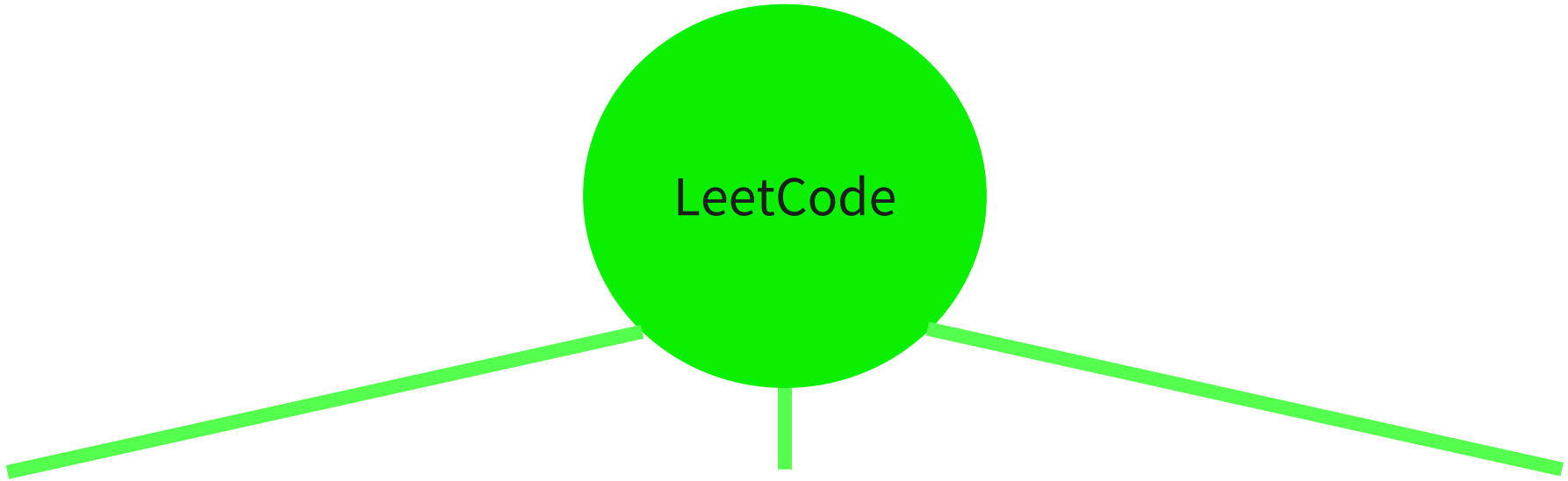
hard

Example 1:
Input: nums = [8,3,9,3,8], k = 2 **Output:** 81 **Explanation:** To get a score of 81, we can apply the following operations: - Choose subarray nums[2, ..., 2]. nums[2] is the only element in this subarray. Hence, we multiply the score by nums[2]. The score becomes 1 * 9 = 9. - Choose subarray nums[2, ..., 3]. Both nums[2] and nums[3] have a prime score of 1, but nums[2] has the smaller index. Hence, we multiply the score by nums[2]. The score becomes 9 * 9 = 81. It can be proven that 81 is the highest score one can obtain.
Example 2:
Input: nums = [19,12,14,6,10,18], k = 3 **Output:** 4788 **Explanation:** To get a score of 4788, we can apply the following operations: - Choose subarray nums[0, ..., 0]. nums[0] is the only element in this subarray. Hence, we multiply the score by nums[0]. The score becomes 1 * 19 = 19. - Choose subarray nums[5, ..., 5]. nums[5] is the only element in this subarray. Hence, we multiply the score by nums[5]. The score becomes 19 * 18 = 342. - Choose subarray nums[2, ..., 3]. Both nums[2] and nums[3] have a prime score of 2, but nums[2] has the smaller index. Hence, we multipy the score by nums[2]. The score becomes 342 * 14 = 4788. It can be proven that 4788 is the highest score one can obtain.

Constraints:
•$1 <= nums.length == n <= 10^5$
•$1 <= nums[i] <= 10^5$
•$1 <= k <= min(n * (n + 1) / 2, 10^9)$

Graphic Source: Custom Depiction.

DATA



Graphic Source: Custom Depiction.

EVALUATION METRICS

CORRECT SOLUTIONS

LINES OF CODE

CYCLOMATIC COMPLEXITY

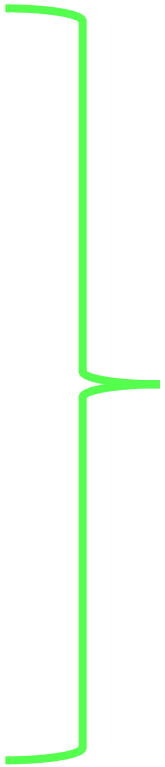
TIME COMPLEXITY

SPACE COMPLEXITY

RUNTIME

MEMORY USAGE

MAINTAINABILITY INDEX



EFFICIENCY & MAINTAINABILITY

Graphic Source: Custom Depiction.

EVALUATION METRICS

CORRECT SOLUTIONS

LINES OF CODE

CYCLOMATIC COMPLEXITY

TIME COMPLEXITY

SPACE COMPLEXITY

RUNTIME

MEMORY USAGE

MAINTAINABILITY INDEX

EFFICIENCY & MAINTAINABILITY

POTENTIAL OF INCORRECT LLM-GENERATED PROGRAM CODE

Graphic Source: Custom Depiction.

EXPERIMENTS AND RESULTS

RESULTS: CORRECT SOLUTIONS

	easy		medium		hard		total
	#1	#2	#3	#4	#5	#6	
single (no roles)	✓	✓	—	—	—	—	2
single (roles)	✓	✓	✓	—	—	—	3
multi (no roles)	✓	✓	✓	—	—	—	3
multi (roles)	✓	✓	—	✓	—	—	3
human	✓	✓	✓	✓	✓	✓	6

EXECUTABLE & CORRECT

Image Source: Idrisov et al. (2025).

RESULTS: CORRECT SOLUTIONS

	easy		medium		hard		total
	#1	#2	#3	#4	#5	#6	
single (no roles)	✓	✓	—	—	—	—	2
single (roles)	✓	✓	✓	—	—	—	3
multi (no roles)	✓	✓	✓	—	—	—	3
multi (roles)	✓	✓	—	✓	—	—	3
human	✓	✓	✓	✓	✓	✓	6

EXECUTABLE & CORRECT: 11 (46%)

Image Source: Idrisov et al. (2025).

RESULTS: CORRECT SOLUTIONS

	easy		medium		hard		total
	#1	#2	#3	#4	#5	#6	
single (no roles)	✓	✓	—	—	—	—	2
single (roles)	✓	✓	✓	—	—	—	3
multi (no roles)	✓	✓	✓	—	—	—	3
multi (roles)	✓	✓	—	✓	—	—	3
human	✓	✓	✓	✓	✓	✓	6

50%

EXECUTABLE & CORRECT

Image Source: Idrisov et al. (2025).

RESULTS: RUNTIME

	easy		medium		hard	
	#1	#2	#3	#4	#5	#6
single (no roles)	118	36	—	—	—	—
single (roles)	**107**	41	984	—	—	—
multi (no roles)	123	**33**	**924**	—	—	—
multi (roles)	120	45	—	309	—	—
human	114	47	961	**235**	**5k**	**1k**
best vs. *human* (Δ in %)	+6	+30	+4	-21	—	—

IN MILLISECONDS

Image Source: Idrisov et al. (2025).

RESULTS: RUNTIME

	easy		medium		hard	
	#1	#2	#3	#4	#5	#6
single (no roles)	118	36	—	—	—	—
single (roles)	**107**	41	984	—	—	—
multi (no roles)	123	**33**	**924**	—	—	—
multi (roles)	120	45	—	309	—	—
human	114	47	961	**235**	**5k**	**1k**
best vs. *human* (Δ in %)	+6	+30	+4	-21	—	—

IN MILLISECONDS

Image Source: Idrisov et al. (2025).

RESULTS: MAINTAINABILITY INDEX (MI)

	easy		medium		hard	
	#1	#2	#3	#4	#5	#6
single (no roles)	54	**64**	—	—	—	—
single (roles)	56	**64**	55	—	—	—
multi (no roles)	55	63	**56**	—	—	—
multi (roles)	48	63	—	42	—	—
human	**61**	**64**	**56**	**63**	**39**	**54**
best vs. *human* (Δ in %)	-8	—	0	-33	—	—

MAINTAINABILITY INDEX
= 171 - 5.2 * LN(HALSTEAD VOLUME) - 0.23 * (CYCLOMATIC COMPLEXITY) - 16.2 * LN(LINES OF CODE)

Image Source: Idrisov et al. (2025).

RESULTS: MAINTAINABILITY INDEX (MI)

	easy		medium		hard	
	#1	#2	#3	#4	#5	#6
single (no roles)	54	**64**	—	—	—	—
single (roles)	56	**64**	55	—	—	—
multi (no roles)	55	63	**56**	—	—	—
multi (roles)	48	63	—	42	—	—
human	**61**	**64**	**56**	**63**	**39**	**54**
best vs. *human* (Δ in %)	-8	—	0	-33	—	—

MAINTAINABILITY INDEX
= 171 - 5.2 * LN(HALSTEAD VOLUME) - 0.23 * (CYCLOMATIC COMPLEXITY) - 16.2 * LN(LINES OF CODE)

Image Source: Idrisov et al. (2025). Text Source: Halstead (1977).

RESULTS: POTENTIAL OF INCORRECT PROGRAM CODE

Time **T**o **C**orrect
the incorrect code
in seconds

	#	MI	$T_{incorrect}$ | $T_{correct}$	TTC	Δ $T_{correct}$–TTC (%)
multi (no roles)	3	55	1,054 | 881	1,046	–15.74
single (no roles)	4	55	669 | 685	581	**+17.93**
single (roles)	4	55	646 | 663	539	**+23.11**
multi (no roles)	4	55	699 | 714	585	**+22.04**
single (no roles)	5	48	1,694 | 6,270	6,384	–1.79
single (roles)	5	54	1,296 | 6,270	6,082	**+3.10**
multi (no roles)	5	54	1,245 | 6,270	6,092	**+2.93**
multi (roles)	5	44	2,885 | 6,270	7,129	–12.05
single (no roles)	6	49	2,204 | 1,341	3,179	–57.82
single (roles)	6	46	3,320 | 3,425	3,954	–13.38
multi (no roles)	6	48	1,685 | 1,341	2,158	–37.85
multi (roles)	6	52	1,849 | 1,341	2,244	–40.23

$$T_{maintain} = \frac{T_{incorrect}}{MI/100} - T_{incorrect} \qquad TTC = |T_{correct} - T_{incorrect}| + T_{maintain}$$

BASED ON HALSTEAD'S ESTIMATES

Image Source: Idrisov et al. (2025). Text Source: Halstead (1977).

RESULTS: POTENTIAL OF INCORRECT PROGRAM CODE

Time **T**o **C**orrect the incorrect code in seconds

	#	MI	$T_{incorrect} \mid T_{correct}$	TTC	Δ $T_{correct}$–TTC (%)
multi (no roles)	3	55	1,054 | 881	1,046	–15.74
single (no roles)	4	55	669 | 685	581	**+17.93**
single (roles)	4	55	646 | 663	539	**+23.11**
multi (no roles)	4	55	699 | 714	585	**+22.04**
single (no roles)	5	48	1,694 | 6,270	6,384	–1.79
single (roles)	5	54	1,296 | 6,270	6,082	**+3.10**
multi (no roles)	5	54	1,245 | 6,270	6,092	**+2.93**
multi (roles)	5	44	2,885 | 6,270	7,129	–12.05
single (no roles)	6	49	2,204 | 1,341	3,179	–57.82
single (roles)	6	46	3,320 | 3,425	3,954	–13.38
multi (no roles)	6	48	1,685 | 1,341	2,158	–37.85
multi (roles)	6	52	1,849 | 1,341	2,244	–40.23

FOR 5 PROGRAM CODES
TTC **<** $T_{correct}$

$$T_{maintain} = \frac{T_{incorrect}}{MI/100} - T_{incorrect}$$

$$TTC = |T_{correct} - T_{incorrect}| + T_{maintain}$$

Image Source: Idrisov et al. (2025). Text Source: Halstead (1977).

5

CONCLUSION AND FUTURE WORK

CONCLUSION AND FUTURE WORK

Conclusion

- LLMs improve coding efficiency for experts and novices.

CONCLUSION AND FUTURE WORK

Conclusion

- LLMs improve coding efficiency for experts and novices.
- *Single LLMs with role descriptions* and *multi-agent systems* achieved 50% correctness, outperforming *single LLMs without role descriptions* (33%).

CONCLUSION AND FUTURE WORK

Conclusion

- LLMs improve coding efficiency for experts and novices.
- *Single LLMs with role descriptions* and *multi-agent systems* achieved 50% correctness, outperforming *single LLMs without role descriptions* (33%).
- All setups solved *easy* tasks, but none solved *hard* problems.

CONCLUSION AND FUTURE WORK

Conclusion

- LLMs improve coding efficiency for experts and novices.
- *Single LLMs with role descriptions* and *multi-agent systems* achieved 50% correctness, outperforming *single LLMs without role descriptions* (33%).
- All setups solved *easy* tasks, but none solved *hard* problems.
- Multi-agent systems showed better *runtime* performance but did not outperform human-written code in *maintainability*.

CONCLUSION AND FUTURE WORK

Conclusion

- LLMs improve coding efficiency for experts and novices.
- *Single LLMs with role descriptions* and *multi-agent systems* achieved 50% correctness, outperforming *single LLMs without role descriptions* (33%).
- All setups solved *easy* tasks, but none solved *hard* problems.
- Multi-agent systems showed better *runtime* performance but did not outperform human-written code in *maintainability*.

Future Work

- Refine LLM frameworks for better code quality and maintainability.
- Balance predefined roles and flexibility in multi-agent systems.

INTERNATIONAL UNIVERSITY OF APPLIED SCIENCES

THANK YOU

Tim Schlippe
tim.schlippe@iu.org