Close Encounters of the Fourth Kind:

A Journey into Solving Mathematical Problems in Operations Research with AI

Rad Niazadeh

Associate Professor of Operations Management

CHICAGO BOOTH

The University of Chicago Booth School of Business

Associate Professor (*by courtesy*)

TTIC

Academic Advisor

lyft

Joint work with:

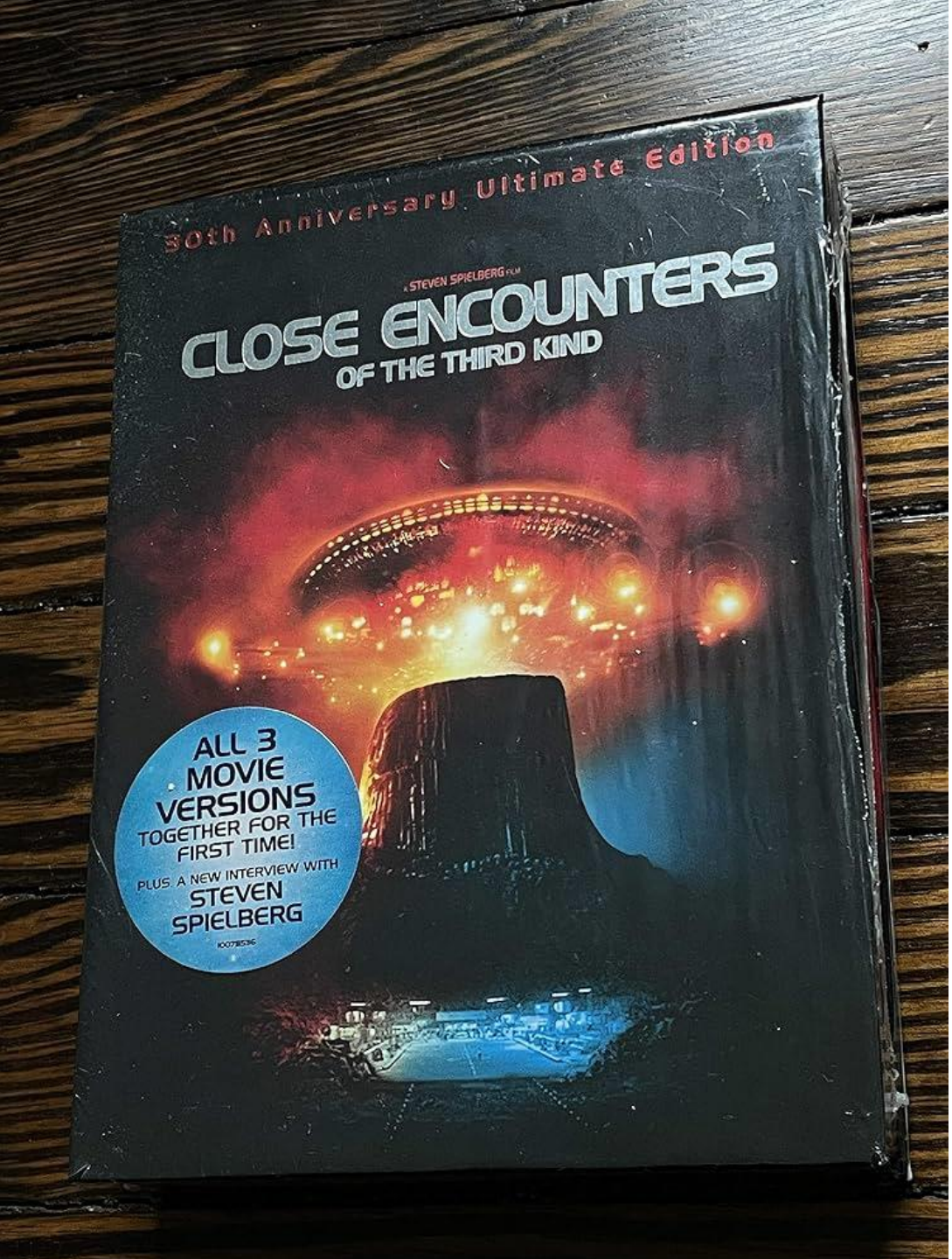
Pranav Nuti*
Chicago Booth

Eric Fithian
Chiago Booth

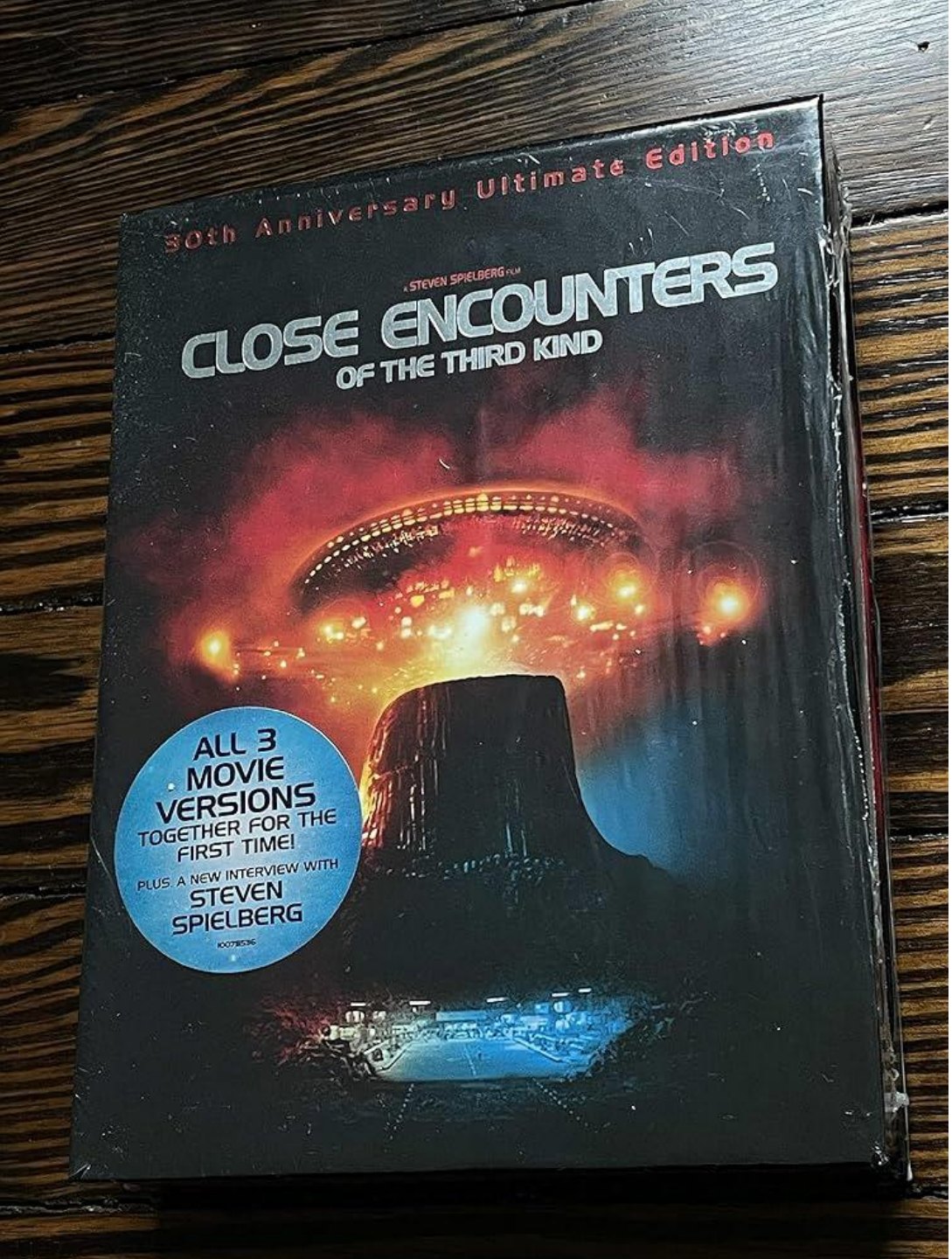
Peter Westbrook
Citadel Securities

Jiechen Zhang
EPFL

Special thanks to **Chicago Booth** and **Center for Applied AI (CAAI)** for their continuing support

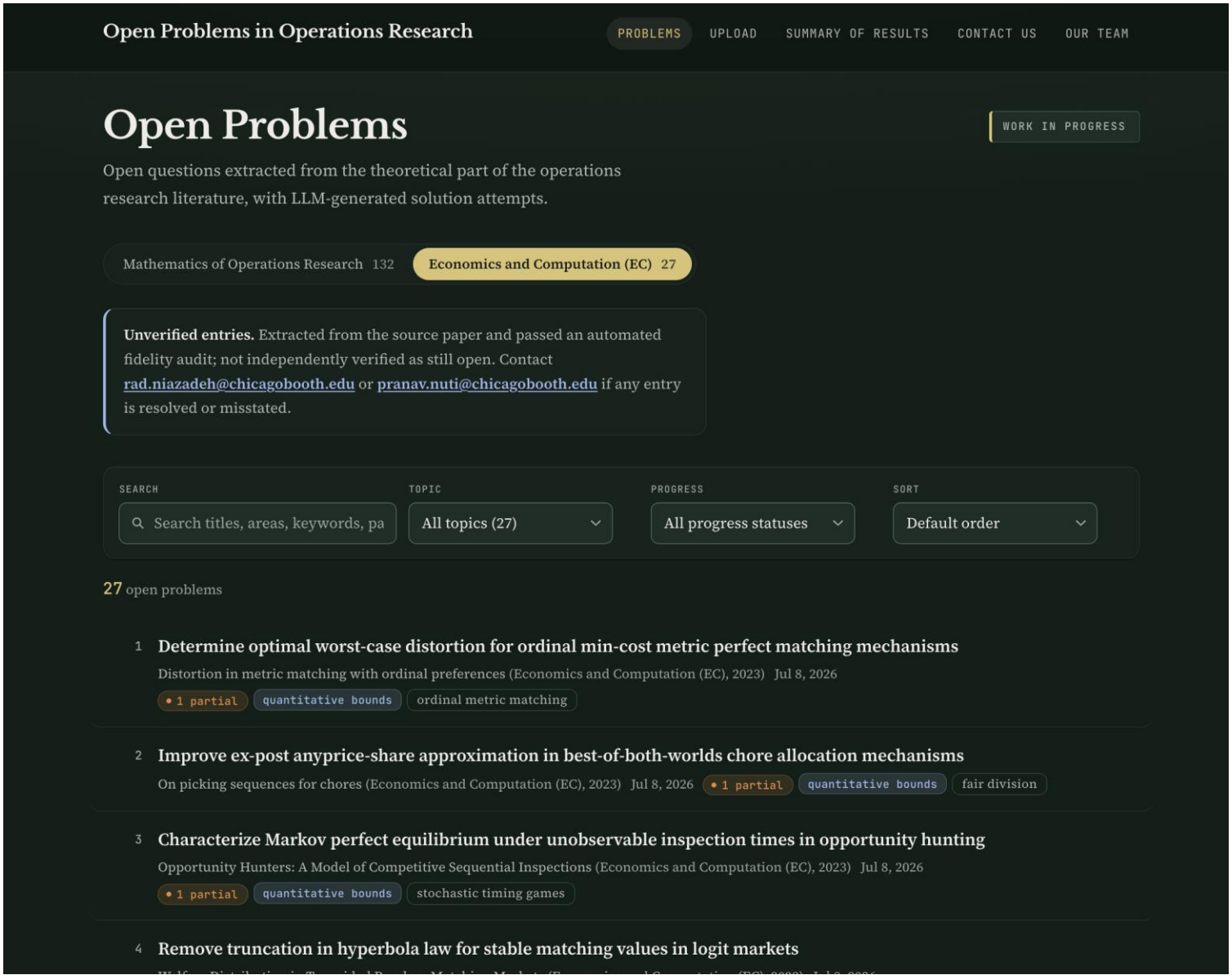


- Classic alien movie,
- Directed by the one and only Steven Spielberg,

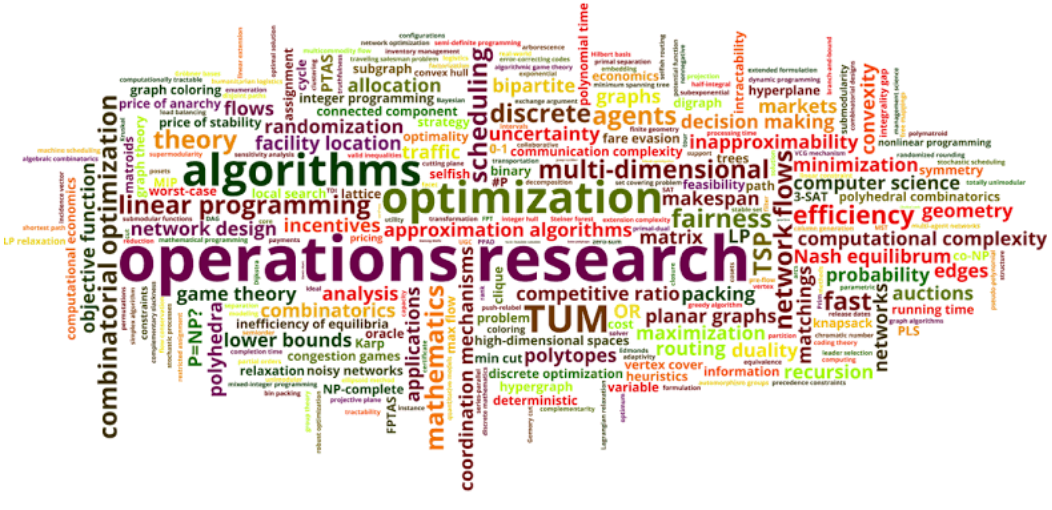


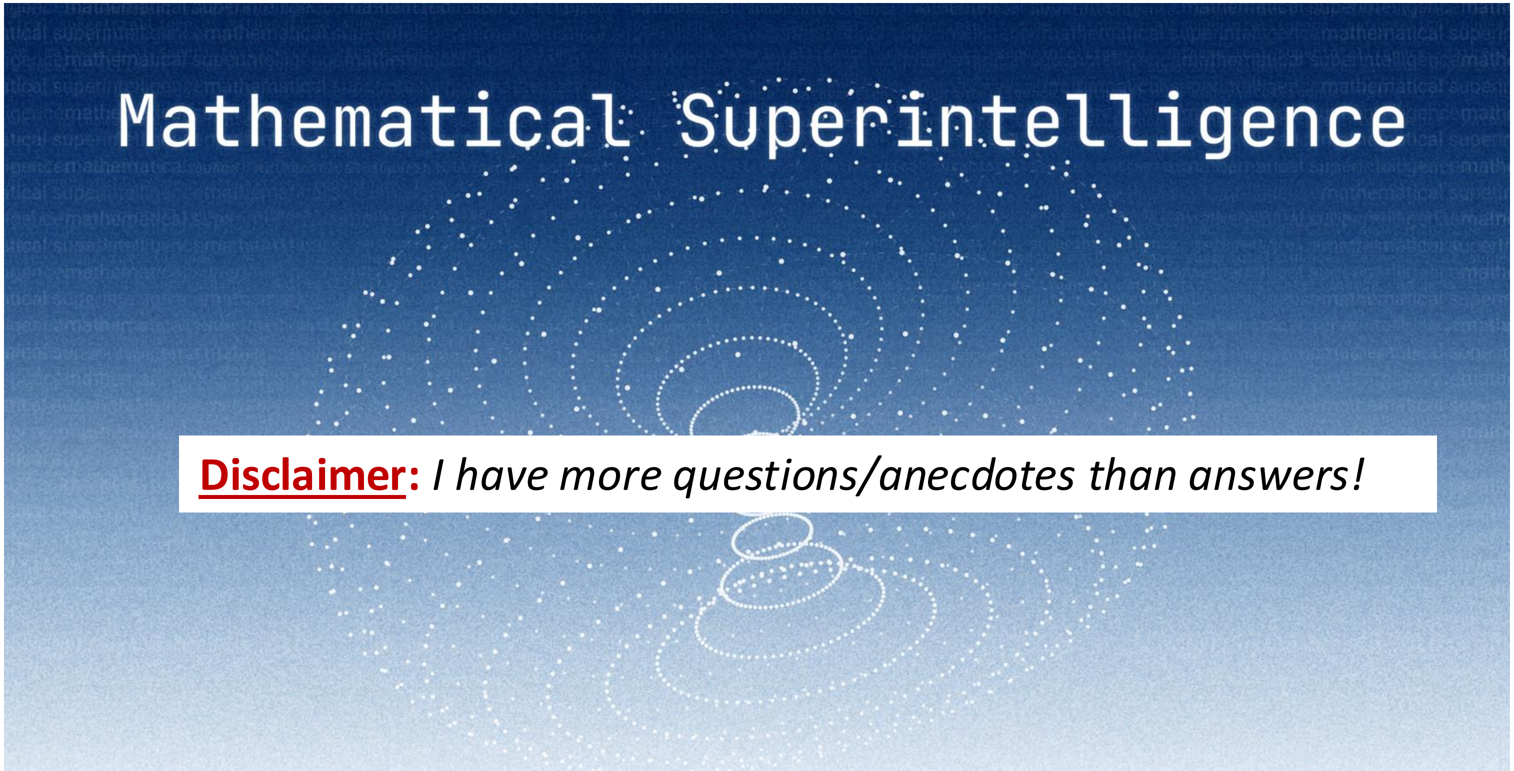
- Classic alien movie,
- Directed by the one and only Steven Spielberg,
- **This is already a pretty good takeaway from my talk!**

Feel free to check the website while listening to my story:
https://ai-in-math-or.github.io/open-problems-in-or/



Mathematical Superintelligence





Disclaimer: *I have more questions/anecdotes than answers!*

Who are we?

Who are we?

- Professor in Operations in a business school
- Background in TCS
- Theoretician (mostly)
- *Online optimization, online learning, algorithmic mechanism design; applications in platform operations & non-profit planning.*

Who are we?

- Professor in Operations in a business school
- Background in TCS
- Theoretician (mostly)
- *Online optimization, online learning, algorithmic mechanism design; applications in platform operations & non-profit planning.*

- My amazing postdoc at Chicago Booth
- Mathematician
- Theoretician (TCS + applied probability)
- *Decisions under uncertainty, prophet inequalities, online & dynamic matching, AI in math, etc.*

Who are we?

- My undergrad pre-doc at Chicago Booth
- Background in CS
- Hacker/full-stack developer; Extremely tech-savvy
- *Agentic AI, foundational models, ML, etc.*

Who are we?

- My undergrad pre-doc at Chicago Booth
- Background in CS
- Hacker/full-stack developer; Extremely tech-savvy
- *Agentic AI, foundational models, ML, etc.*

- Undergrad in math & masters in stat, Stanford University
- Garden leave at a hedge fund
- Full-stack developer & software engineer
- *Agentic AI, ML, mathematical finance, etc.*

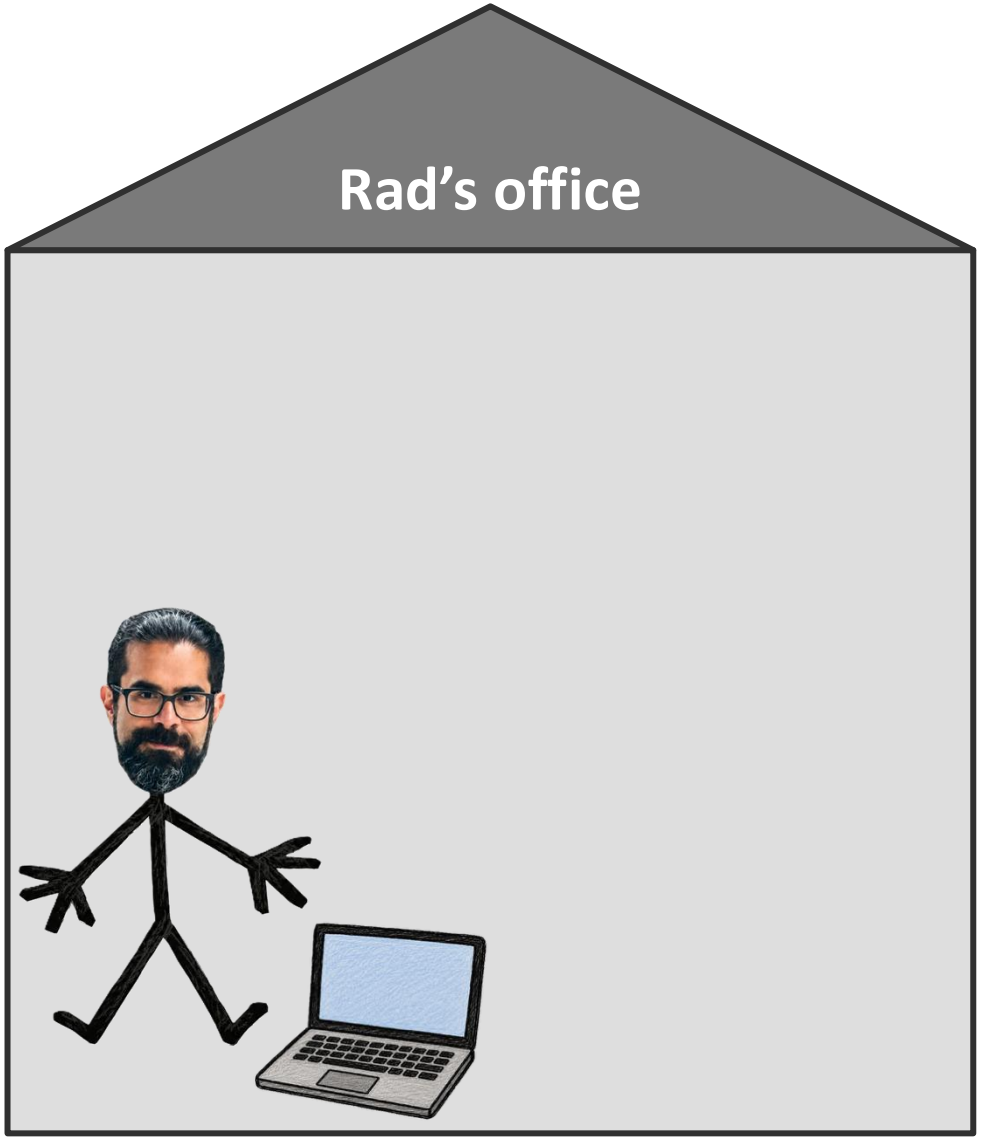
Who are we?

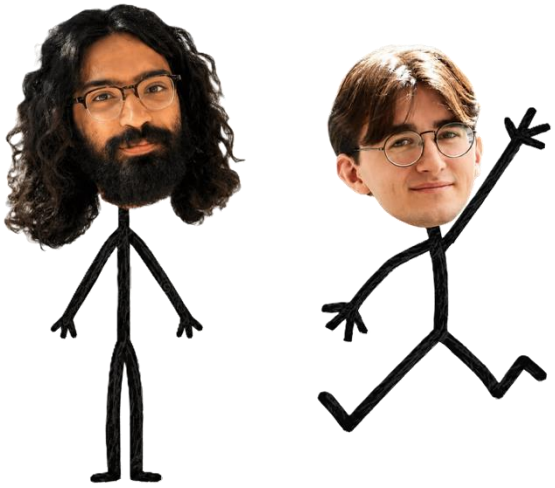
- My undergrad pre-doc at Chicago Booth
- Background in CS
- Hacker/full-stack developer; Extremely tech-savvy
- *Agentic AI, foundational models, ML, etc.*

- Undergrad in math & masters in stat, Stanford University
- Garden leave at a hedge fund
- Full-stack developer & software engineer
- *Agentic AI, ML, mathematical finance, etc.*

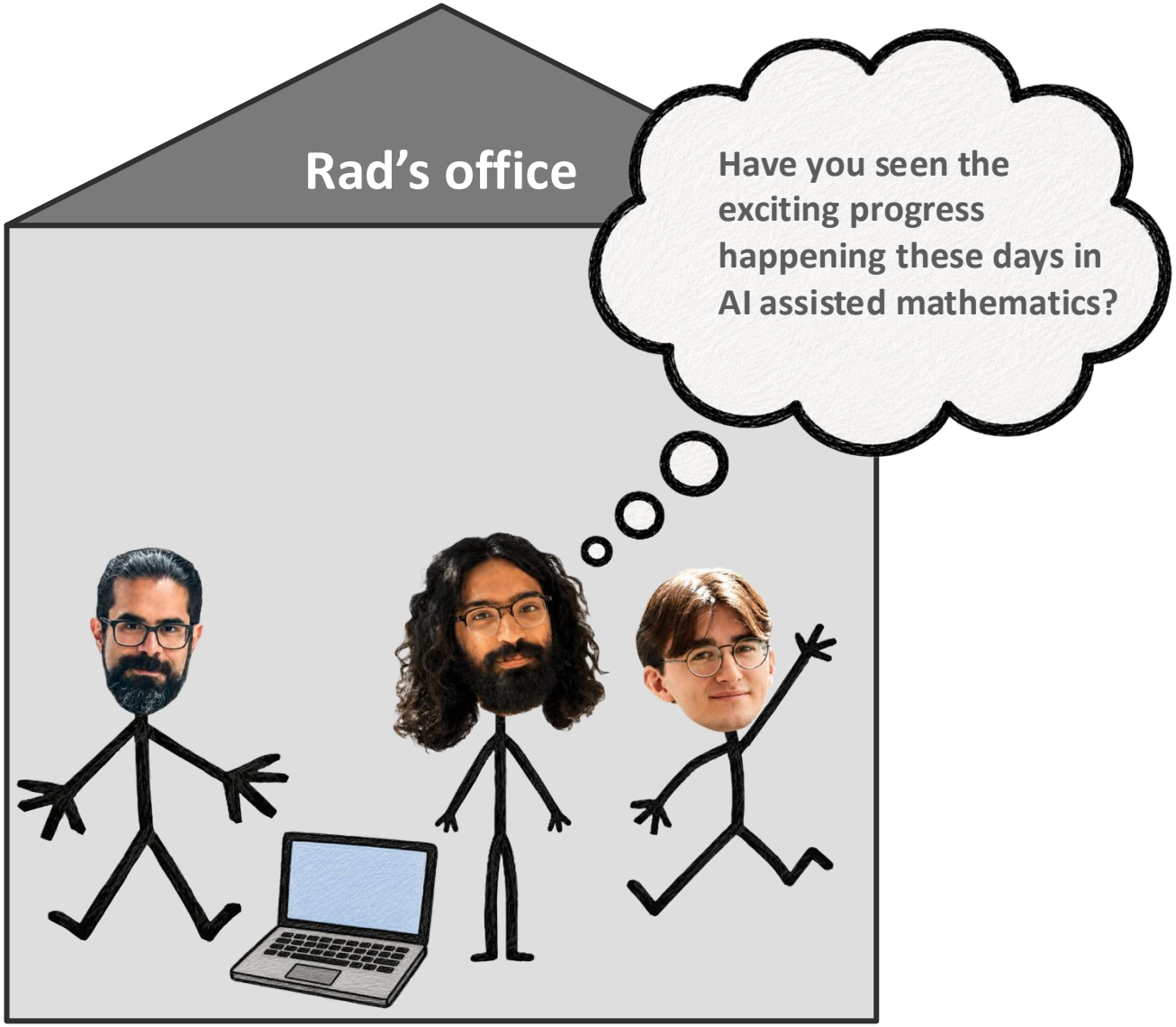
- PhD student at EPFL
- Background in CS; research in Econ/CS
- Passionate about both theory & AI
- *Algorithmic game theory, optimal stopping, agentic AI, etc.*

The story of how we started this project

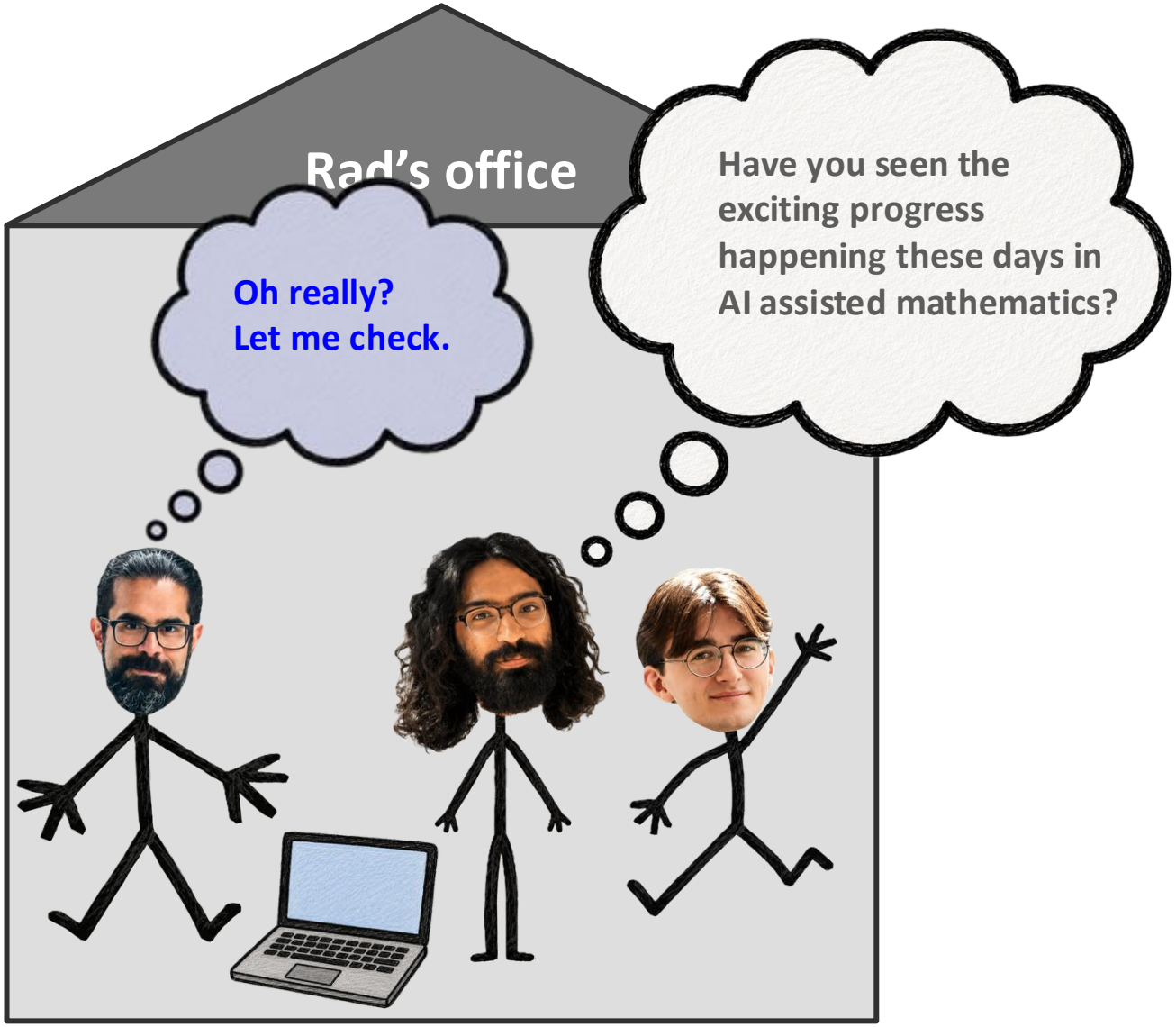




The story of how we started this project



The story of how we started this project



OpenAI

ChatGPT

Google DeepMind

Gemini

VS

ANTHROP\C

Claude

Several startups & more….

OpenAI

ChatGPT

Google DeepMind

Gemini

ANTHROP\C

Claude

Several startups & more….

VS

International Math Olympiad (IMO)

OpenAI

ChatGPT

Google DeepMind

Gemini

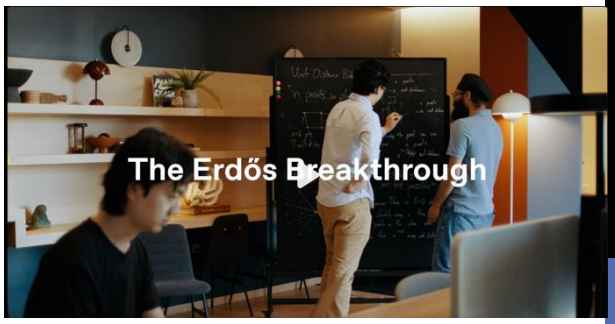
ANTHROP\C

Claude

Several startups & more….

VS

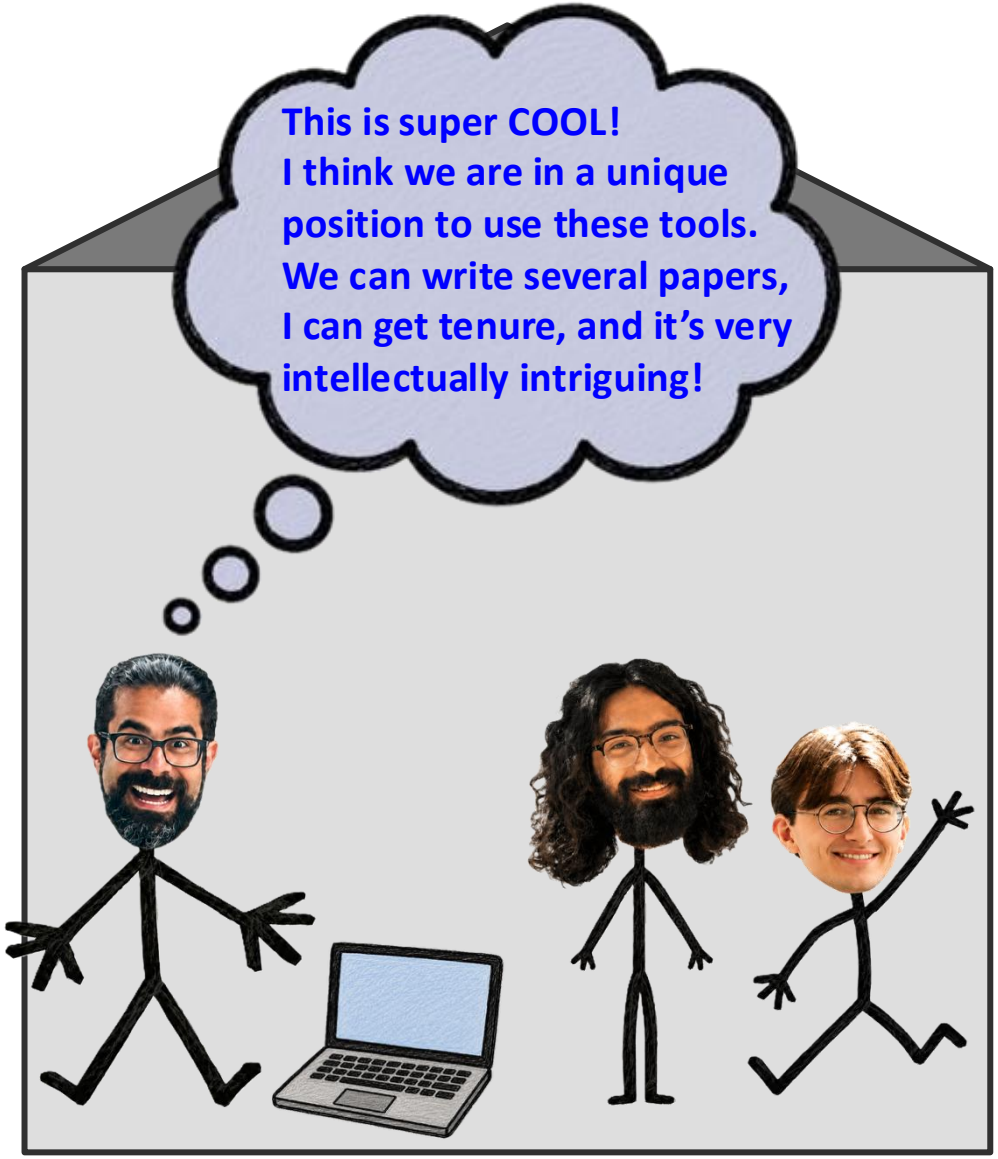
International Math Olympiad (IMO)



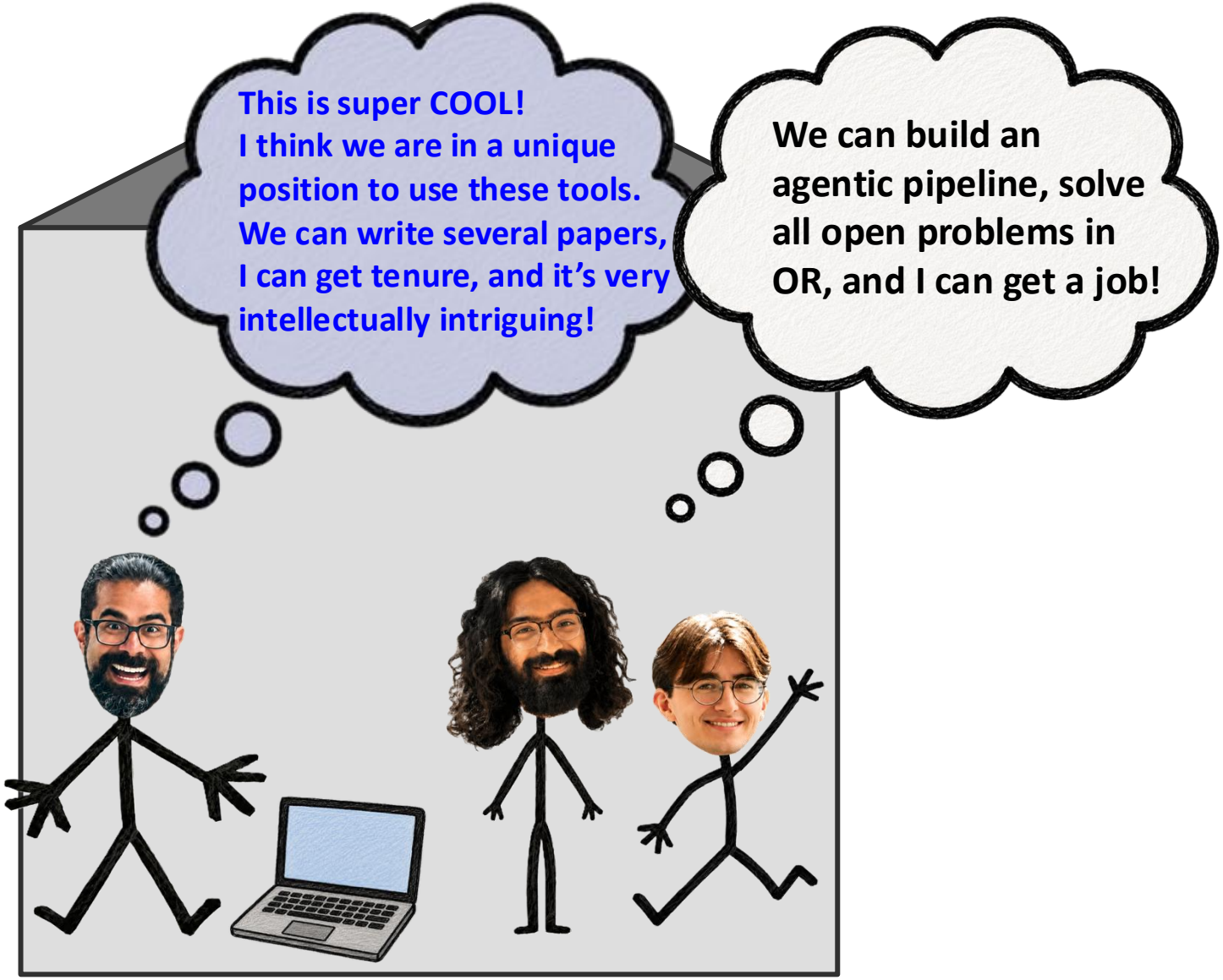
AI generated disproof of Erdős unit-distance conjecture

Aletheia: Google DeepMind's AI Just Solved 4 Erdős Problems Autonomously

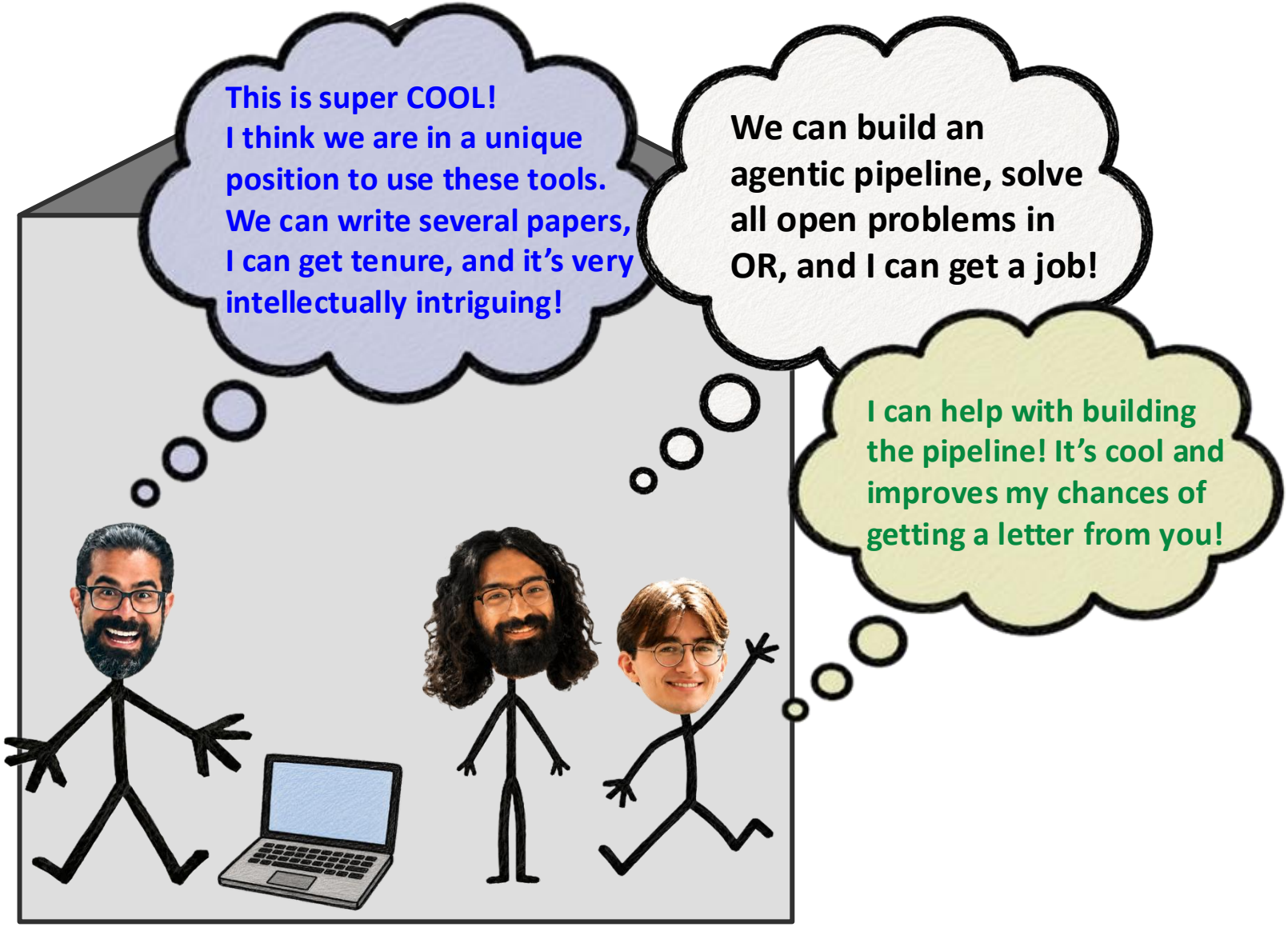
The story of how we started this project



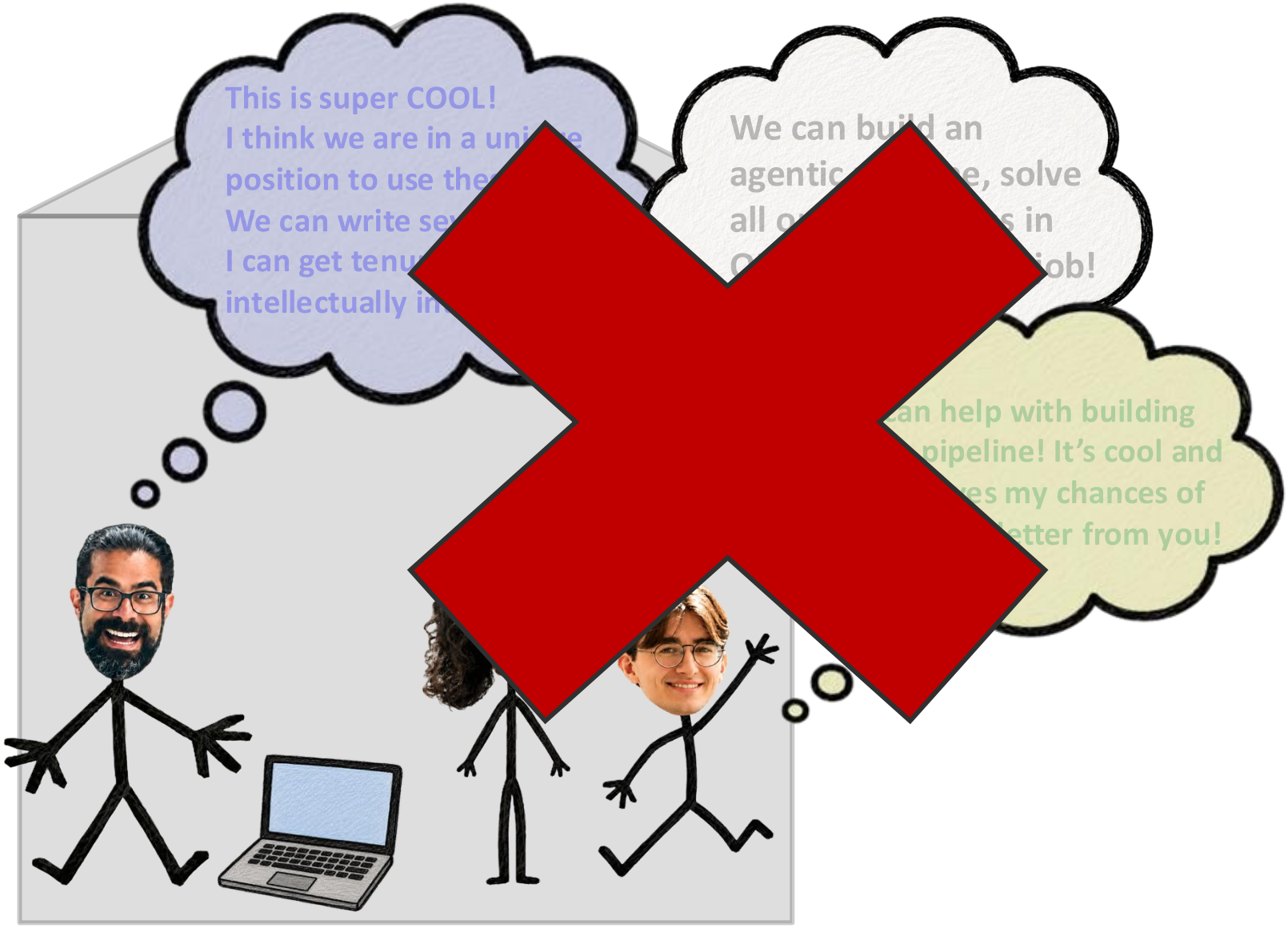
The story of how we started this project



The story of how we started this project

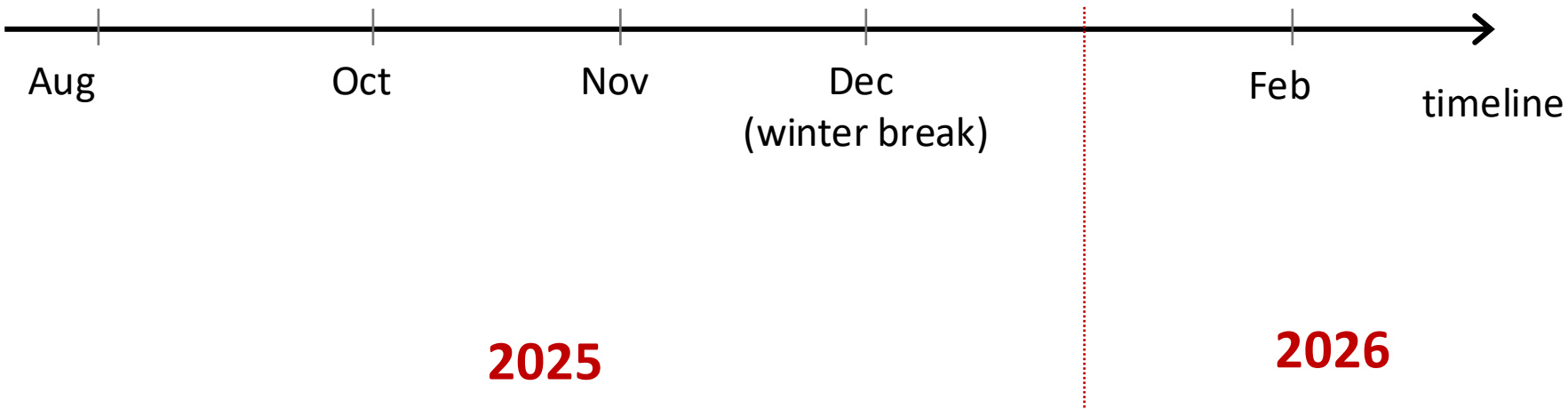


The story of how we started this project



The real story…

Things changed quickly from Aug 2025 to Feb 2026!



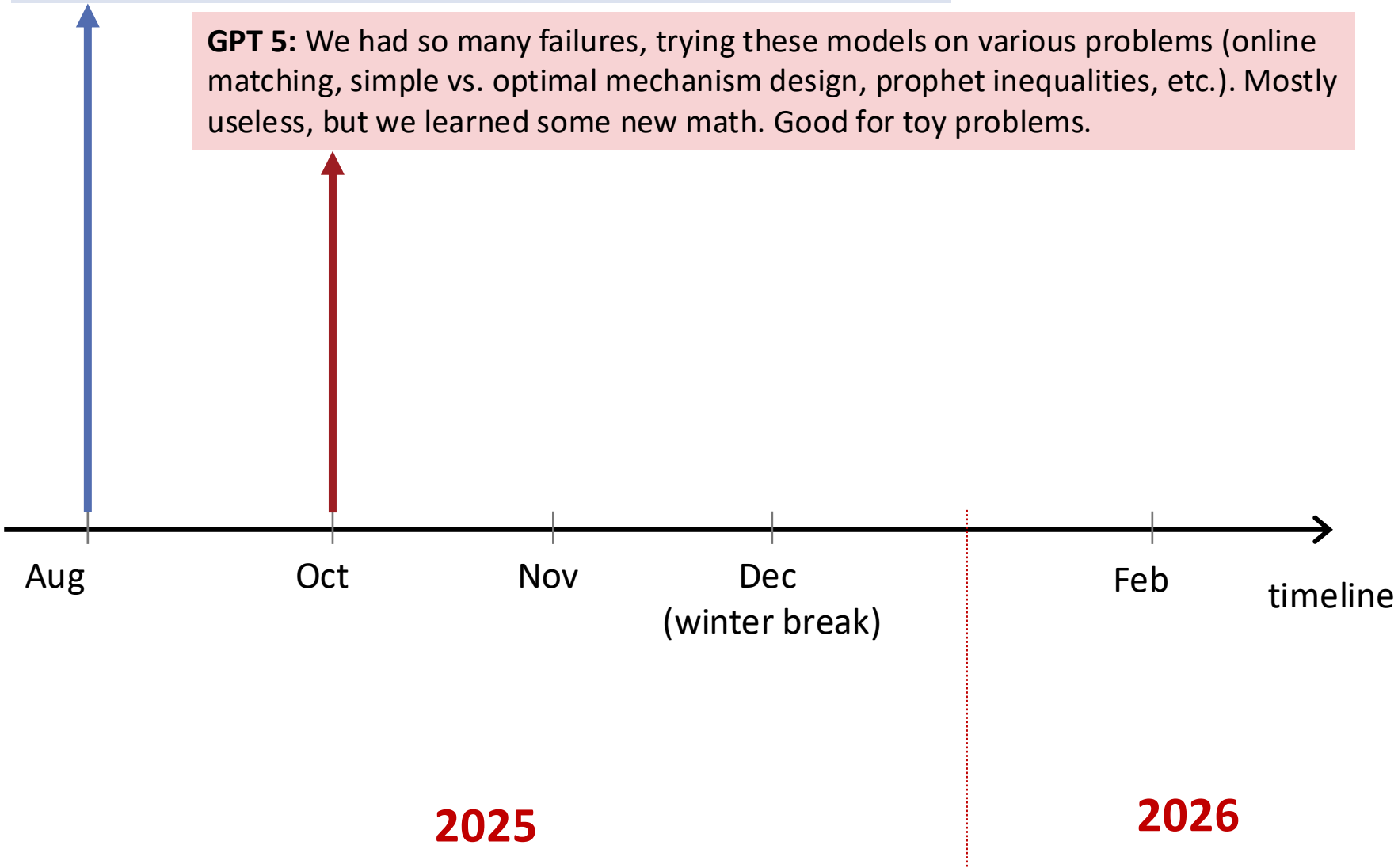
Things changed quickly from Aug 2025 to Feb 2026!

Pranav and I started playing with subscription frontier models (ChatGPT, Claude, Gemini)

Things changed quickly from Aug 2025 to Feb 2026!

Pranav and I started playing with subscription frontier models (ChatGPT, Claude, Gemini)

GPT 5: We had so many failures, trying these models on various problems (online matching, simple vs. optimal mechanism design, prophet inequalities, etc.). Mostly useless, but we learned some new math. Good for toy problems.

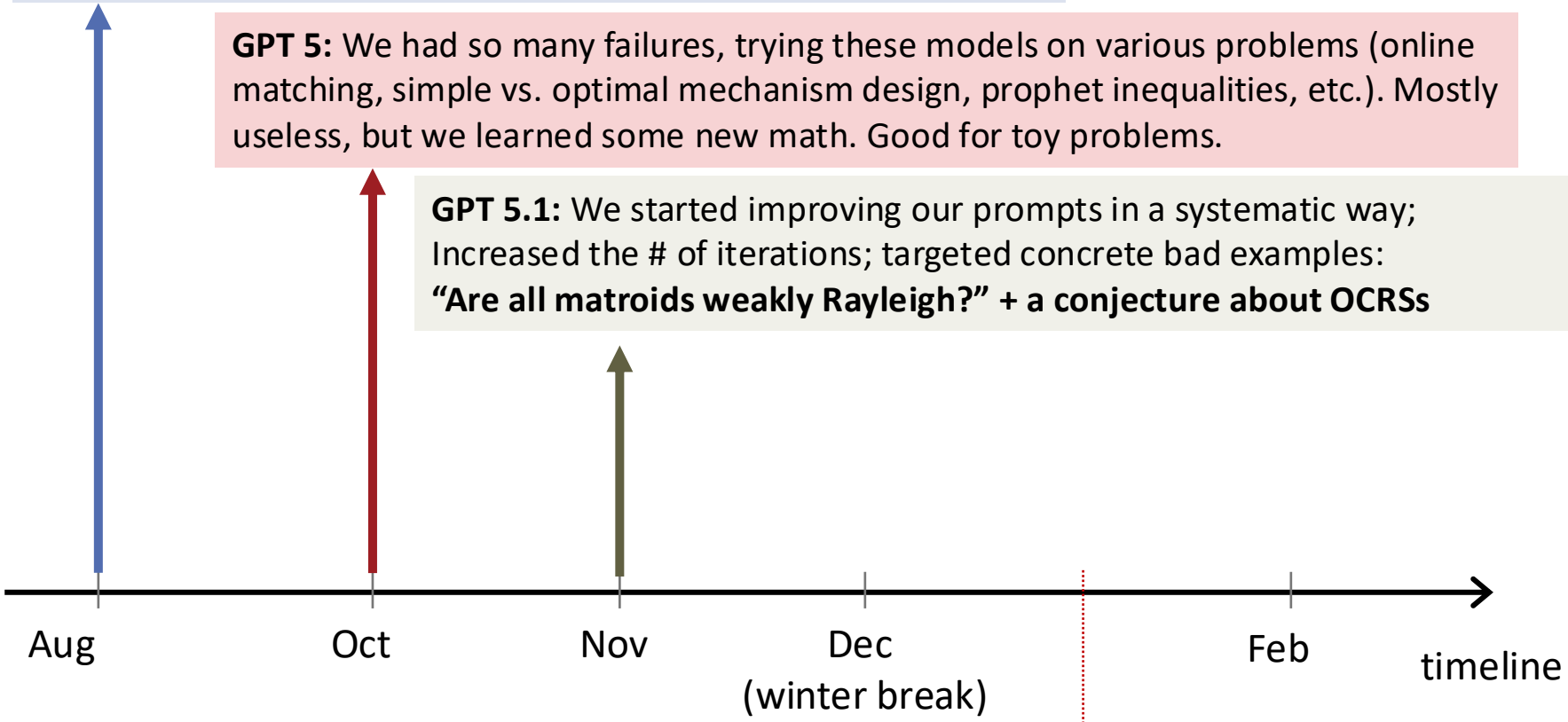


Things changed quickly from Aug 2025 to Feb 2026!

Pranav and I started playing with subscription frontier models (ChatGPT, Claude, Gemini)

GPT 5: We had so many failures, trying these models on various problems (online matching, simple vs. optimal mechanism design, prophet inequalities, etc.). Mostly useless, but we learned some new math. Good for toy problems.

GPT 5.1: We started improving our prompts in a systematic way; Increased the # of iterations; targeted concrete bad examples: **"Are all matroids weakly Rayleigh?" + a conjecture about OCRSs**



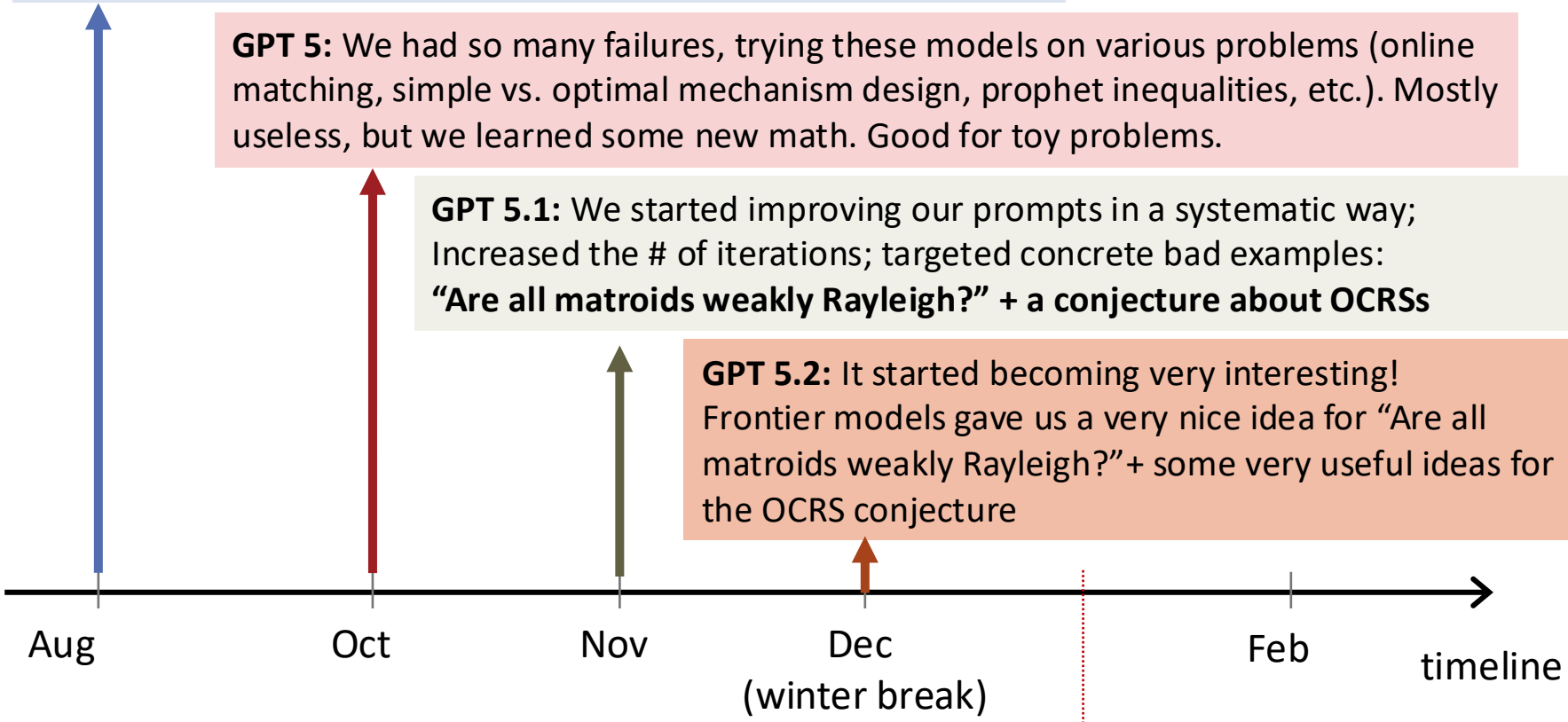
Things changed quickly from Aug 2025 to Feb 2026!

Pranav and I started playing with subscription frontier models (ChatGPT, Claude, Gemini)

GPT 5: We had so many failures, trying these models on various problems (online matching, simple vs. optimal mechanism design, prophet inequalities, etc.). Mostly useless, but we learned some new math. Good for toy problems.

GPT 5.1: We started improving our prompts in a systematic way; Increased the # of iterations; targeted concrete bad examples: **"Are all matroids weakly Rayleigh?" + a conjecture about OCRSs**

GPT 5.2: It started becoming very interesting! Frontier models gave us a very nice idea for "Are all matroids weakly Rayleigh?"+ some very useful ideas for the OCRS conjecture



Things changed quickly from Aug 2025 to Feb 2026!

Pranav and I started playing with subscription frontier models (ChatGPT, Claude, Gemini)

GPT 5: We had so many failures, trying these models on various problems (online matching, simple vs. optimal mechanism design, prophet inequalities, etc.). Mostly useless, but we learned some new math. Good for toy problems.

GPT 5.1: We started improving our prompts in a systematic way; Increased the # of iterations; targeted concrete bad examples: **"Are all matroids weakly Rayleigh?" + a conjecture about OCRSs**

GPT 5.2: It started becoming very interesting! Frontier models gave us a very nice idea for "Are all matroids weakly Rayleigh?"+ some very useful ideas for the OCRS conjecture

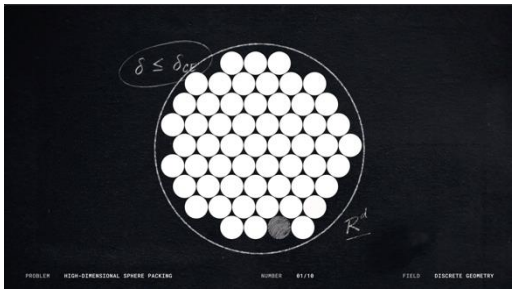
Aug Oct Nov Dec (winter break) Feb timeline

GPT 5.4 & 5.5: clearly capable models. Found the hard instance for weakly Rayleigh matroid conjecture! ***(Answer: no; contact me for the bad example/proof)***

2025 **2026**

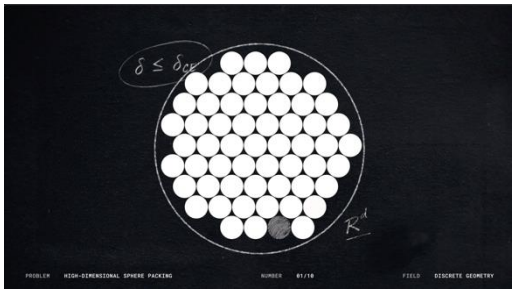
and this was only the beginning…

and this was only the beginning…

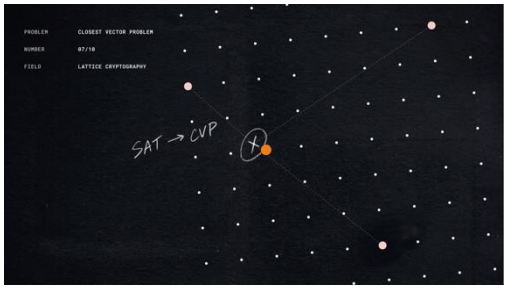


High-dimensional sphere packing

and this was only the beginning…

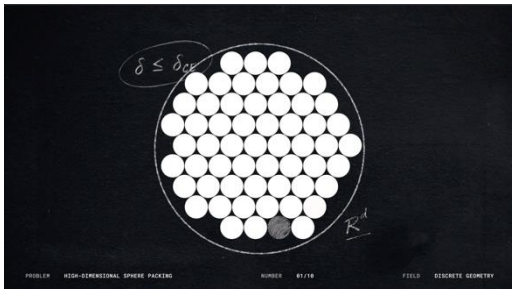


High-dimensional sphere packing

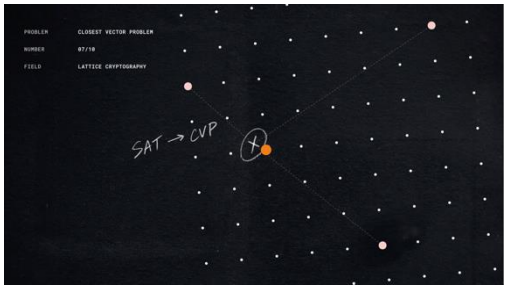


Closest vector problem

and this was only the beginning…



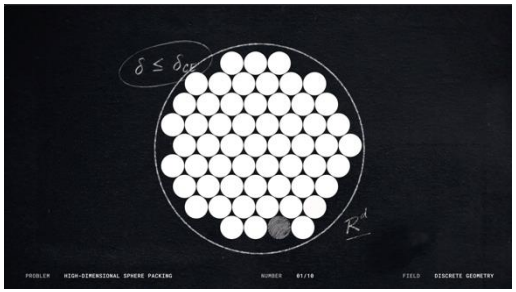
High-dimensional sphere packing



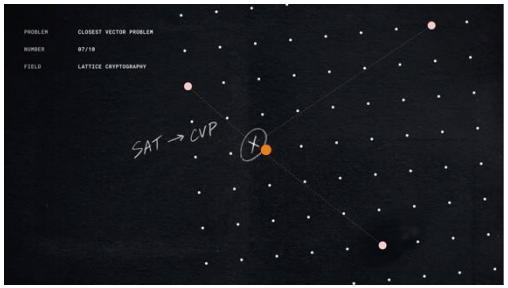
Closest vector problem

Multi-color Ramsey numbers

and this was only the beginning…



High-dimensional sphere packing



Closest vector problem

Multi-color Ramsey numbers

A PROOF OF THE CYCLE DOUBLE COVER CONJECTURE BY OPENAI: AN EXPOSITION

SANG-IL OUM

ABSTRACT. The cycle double cover conjecture states that every bridgeless graph has a list of cycles such that every edge is in exactly two of them. In July 2026, OpenAI announced a proof. This exposition presents the proof with slight modifications intended to make it more accessible.

PROMPT USED FOR "A PROOF OF THE CYCLE DOUBLE COVER CONJECTURE"

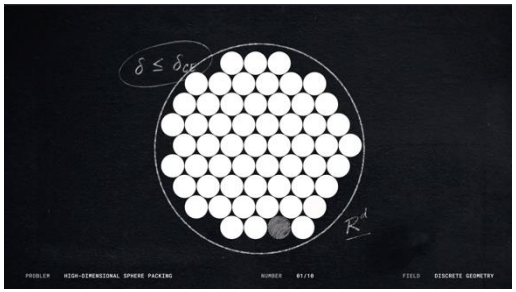
OPENAI

ABSTRACT. This document contains the full prompt given to GPT 5.6 Sol Ultra which led to its proof of the Cycle Double Cover Conjecture.

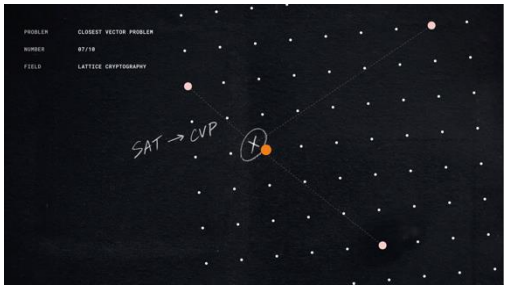
1. PROMPT

```
Current task statement

A graph here is a finite loopless undirected multigraph: parallel edges are allowed and are
distinct. A bridge is an edge whose deletion increases the number of connected components. A
cycle is a connected 2-regular submultigraph; thus two parallel edges form a cycle of length
two. A cycle double cover of G is a finite multiset of cycles of G such that every edge of G
occurs in exactly two members of the multiset, counted with multiplicity.

Resolve the Cycle Double Cover Conjecture completely:

Every finite bridgeless loopless multigraph has a cycle double cover.

Disconnected graphs are permitted, and the edgeless graph has the empty cycle double cover.
Cycles in the cover need not be induced or edge-disjoint from one another; the requirement
is exactly two total occurrences of each edge.

Assume for purposes of this task that a complete affirmative proof exists. A complete
solution must prove exactly the following:
```

and this was only the beginning…



High-dimensional sphere packing



Closest vector problem

Multi-color Ramsey numbers

A PROOF OF THE CYCLE DOUBLE COVER CONJECTURE BY OPENAI: AN EXPOSITION

SANG-IL OUM

ABSTRACT. The cycle double cover conjecture states that every bridgeless graph has a list of cycles such that every edge is in exactly two of them. In July 2026, OpenAI announced a proof. This exposition presents the proof with slight modifications intended to make it more accessible.

PROMPT USED FOR "A PROOF OF THE CYCLE DOUBLE COVER CONJECTURE"

OPENAI

ABSTRACT. This document contains the full prompt given to GPT 5.6 Sol Ultra which led to its proof of the Cycle Double Cover Conjecture.

1. PROMPT

Current task statement

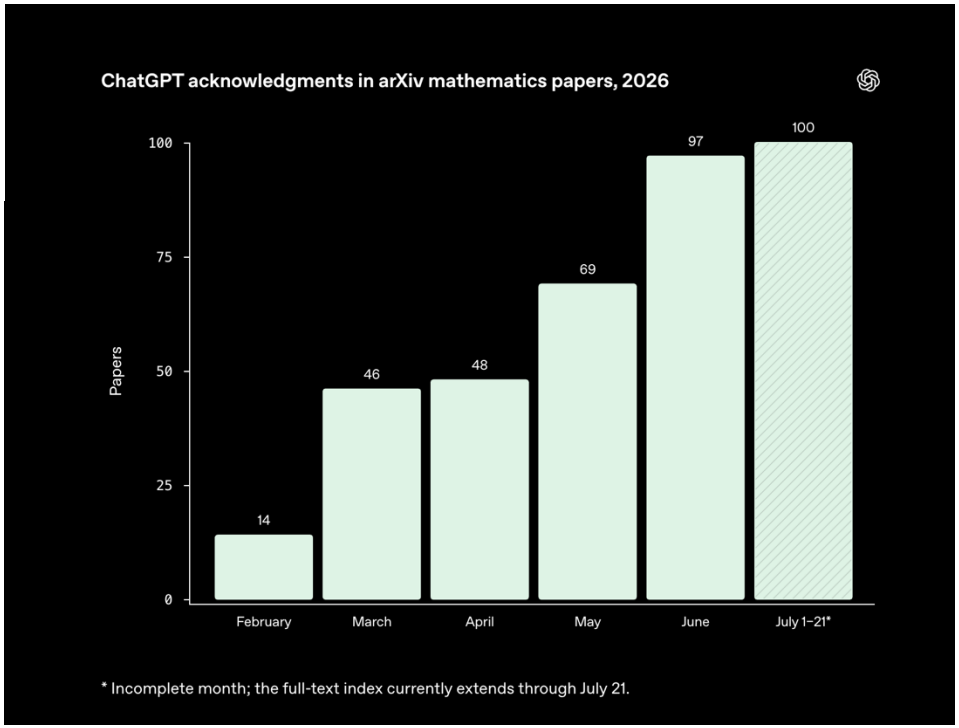
A graph here is a finite loopless undirected multigraph: parallel edges are allowed and are distinct. A bridge is an edge whose deletion increases the number of connected components. A cycle is a connected 2-regular submultigraph; thus two parallel edges form a cycle of length two. A cycle double cover of G is a finite multiset of cycles of G such that every edge of G occurs in exactly two members of the multiset, counted with multiplicity.

Resolve the Cycle Double Cover Conjecture completely:

Every finite bridgeless loopless multigraph has a cycle double cover.

Disconnected graphs are permitted, and the edgeless graph has the empty cycle double cover. Cycles in the cover need not be induced or edge-disjoint from one another; the requirement is exactly two total occurrences of each edge.

Assume for purposes of this task that a complete affirmative proof exists. A complete solution must prove exactly the following:



Asymptotically Tight Fractional Online Matching Under Edge Arrivals

David Wajc*
Technion

Abstract

In this brief note, we close the asymptotic gap between known upper and lower bounds for fractional online matching under edge arrivals. We prove that the optimal competitive ratio for this problem is $1/2+\Theta(1/n)$. The algorithm was suggested and analyzed by OpenAI's ChatGPT Sol based on a single prompt. The presentation was streamlined over a few hours, based on a back and forth discussion with the author, who assumes responsibility for any errors.

A PROOF

ABSTRACT. The cycle double cover conjecture states that every bridgeless graph has a list of cycles such that every edge is in exactly two of them. In July 2026, OpenAI announced a proof. This exposition presents the proof with slight modifications intended to make it more accessible.

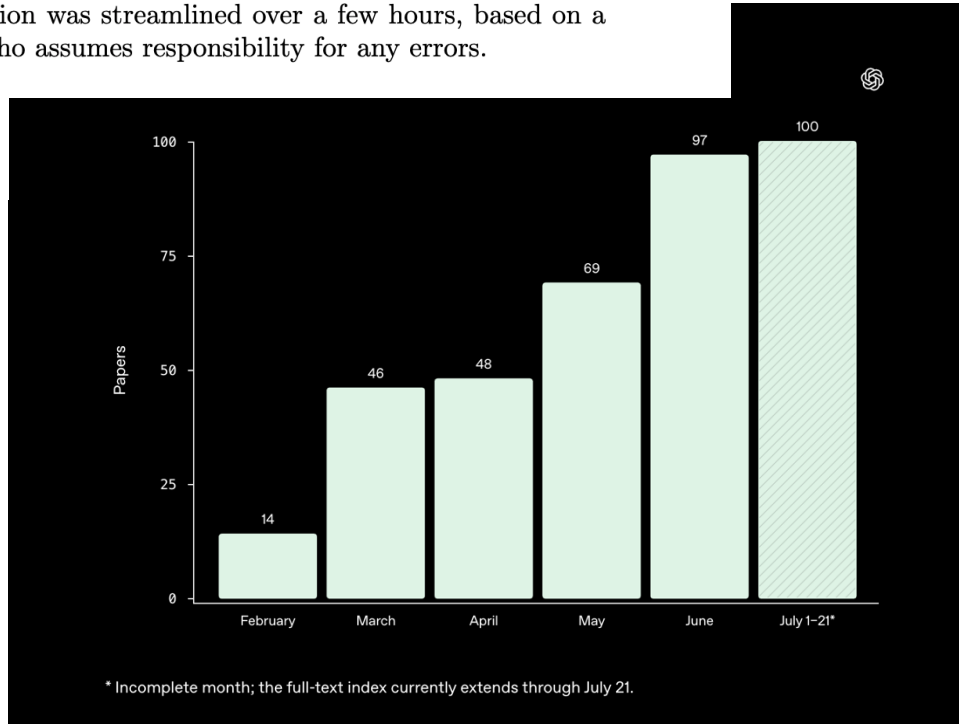
PROMPT USED FOR "A PROOF OF THE CYCLE DOUBLE COVER CONJECTURE"

OPENAI

ABSTRACT. This document contains the full prompt given to GPT 5.6 Sol Ultra which led to its proof of the Cycle Double Cover Conjecture.

1. PROMPT

```
Current task statement

A graph here is a finite loopless undirected multigraph: parallel edges are allowed and are
distinct. A bridge is an edge whose deletion increases the number of connected components. A
cycle is a connected 2-regular submultigraph; thus two parallel edges form a cycle of length
two. A cycle double cover of G is a finite multiset of cycles of G such that every edge of G
occurs in exactly two members of the multiset, counted with multiplicity.

Resolve the Cycle Double Cover Conjecture completely:

Every finite bridgeless loopless multigraph has a cycle double cover.

Disconnected graphs are permitted, and the edgeless graph has the empty cycle double cover.
Cycles in the cover need not be induced or edge-disjoint from one another; the requirement
is exactly two total occurrences of each edge.

Assume for purposes of this task that a complete affirmative proof exists. A complete
solution must prove exactly the following:
```



arXiv:2608.23350v1 [cs.DS] 24 Aug 2026

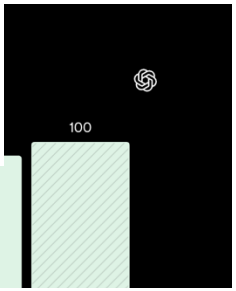
Asymptotically Tight Fractional Online Matching Under Edge Arrivals

David Wajc*
Technion

Abstract

In this brief note, we close the asymptotic gap between known upper and lower bounds for fractional online matching under edge arrivals. We prove that the optimal competitive ratio for this problem is $1/2+\Theta(1/n)$. The algorithm was suggested and analyzed by OpenAI's ChatGPT Sol based on a single prompt. The presentation was streamlined over a few hours, based on a back and forth discussion with the author, who assumes responsibility for any errors.

@David: great job! It is now Lean verified too!



Build and audit the certifi

```
▶ Run bash scripts/aud
== frozen statement ==
Frozen statement hashes match freeze.json.
== implementation source scan ==
== import isolation ==
== comparator and manifest ==
== metadata structure ==
== repository licence layout ==
== package build ==
✔ [3297/3305] Built arXiv2608_23350v1.Trusted (6.0s)
✔ [3298/3305] Built arXiv2608_23350v1.Invariants (10s)
✔ [3299/3305] Built arXiv2608_23350v1.Matching (3.2s)
✔ [3300/3305] Built arXiv2608_23350v1.Global (3.9s)
✔ [3301/3305] Built arXiv2608_23350v1.Main (4.0s)
✔ [3302/3305] Built arXiv2608_23350v1 (3.8s)
✔ [3304/3305] Built Solution (3.9s)
Build completed successfully (3305 jobs).
== deliberate Challenge target ==
⚠ [900/901] Built Challenge (1.5s)
warning: Challenge.lean:102:8: declaration uses `sorry`
Build completed successfully (901 jobs).
== exact headline axiom closure ==
Exact axiom audit passed for FormalizedPapers.arXiv2608_23350v1.fractional_online_matching_edge_arriva
Exact axiom audit passed for FormalizedPapers.arXiv2608_23350v1.fractional_online_matching_edge_arriva
'FormalizedPapers.arXiv2608_23350v1.fractional_online_matching_edge_arrivals_implementation' depends (
 Classical.choice,
 Quot.sound]
'FormalizedPapers.arXiv2608_23350v1.fractional_online_matching_edge_arrivals' depends on axioms: [propext,
 Classical.choice,
 Quot.sound]
== namespace-wide axiom allowlist ==
Project-wide axiom audit passed for 121 declarations.
== independent fresh-kernel replay ==
```

```
theorem fractional_online_matching_edge_arrivals :
    ∀ {n m : ℕ} (edges : Fin m → LooplessEdge n),
      IsSimpleArrival edges →
      1 ≤ matchingNumber edges →
      IsFractionalMatching edges (algorithmWeight edges) ∧
        fractionalValue (algorithmWeight edges) ≥
          (matchingNumber edges : ℝ) / 2 + 1 / 4 ∧
        (matchingNumber edges : ℝ) / 2 + 1 / 4 ≥
          (matchingNumber edges : ℝ) *
            (1 / 2 + 1 / (2 * (n : ℝ))) := by
  intro n m edges hsimple hopt
  exact fractional_online_matching_edge_arrivals_implementation edges hsimple hopt
```

complete month; the full-text index currently extends through July 21.

Research Questions/Goals:

Research Questions/Goals:

I. How powerful are frontier models at solving open problems in operations research?

Research Questions/Goals:

I. How powerful are frontier models at solving open problems in operations research?

We should be prepared when they come after us!

Research Questions/Goals:

I. How powerful are frontier models at solving open problems in operations research?

II. Can we build a pipeline that solves (theoretical) OR problems at scale?

We should be prepared when they come after us!

Research Questions/Goals:

I. How powerful are frontier models at solving open problems in operations research?

II. Can we build a pipeline that solves (theoretical) OR problems at scale?

III. Can we collect data through this experimental study, to formalize how AI can assist theoreticians in OR? (e.g., what types of problems are AI-friendly?)

We should be prepared when they come after us!

Research Questions/Goals:

I. How powerful are frontier models at solving open problems in operations research?

II. Can we build a pipeline that solves (theoretical) OR problems at scale?

III. Can we collect data through this experimental study, to formalize how AI can assist theoreticians in OR? (e.g., what types of problems are AI-friendly?)

IV. Creating a community of OR theoretical researchers using up-to-date methods?

We should be prepared when they come after us!

The things we are NOT doing (and NOT planning to do)

I. **Undervaluing** the amazing theoretical research done in our community by **humans**,

II. **Replacing** theoreticians with AI, or saying theoretical research in OR can be **fully cracked** by AI

III. **Publishing** the solutions or partial progresses we find in any form, or advocating any similar act,

IV. **Trying to compete** with AI companies (Open AI, DeepMind, etc.) to have a better system for math solving or **solve more problems**

Building an agentic pipeline:

**=work in progress **=recently implemented features*

Building an agentic pipeline:

**=work in progress **=recently implemented features*

- *Data base of problems in OR*
  - Collect open problems from different venues
  - Formalize, verify, and reformat
  - Background check & literature review
  - Check forward/old literature for openness
  - Post on our website

Building an agentic pipeline:

*=work in progress **=recently implemented features

- *Data base of problems in OR*
  - Collect open problems from different venues
  - Formalize, verify, and reformat
  - Background check & literature review
  - Check forward/old literature for openness
  - Post on our website

- *AI assisted solutions/progress*
  - Using off-the-shelf frontier models
  - AI ecosystem with various agents
  - Solvers, verifies, formatters, judges, etc.
  - Formalizing solutions via Lean**
  - Automated pipeline + manual
  - Memory management*

Building an agentic pipeline:

*=work in progress **=recently implemented features

- *Data base of problems in OR*
  - Collect open problems from different venues
  - Formalize, verify, and reformat
  - Background check & literature review
  - Check forward/old literature for openness
  - Post on our website

- *AI assisted solutions/progress*
  - Using off-the-shelf frontier models
  - AI ecosystem with various agents
  - Solvers, verifies, formatters, judges, etc.
  - Formalizing solutions via Lean**
  - Automated pipeline + manual
  - Memory management*

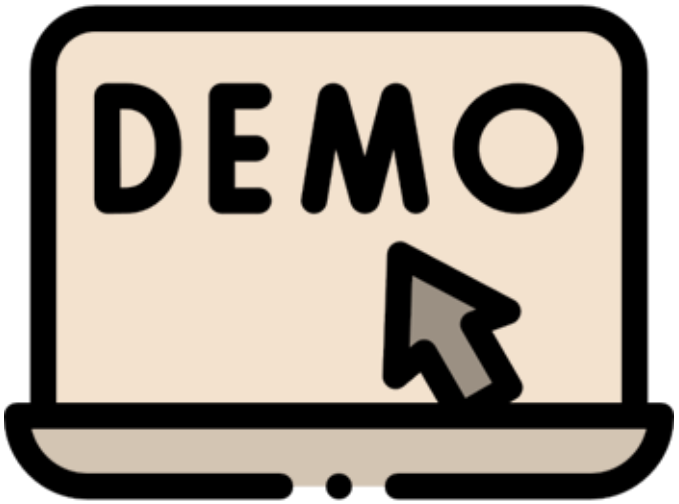
Building an agentic pipeline:

**=work in progress **=recently implemented features*

- *Data base of problems in OR*
  - Collect open problems from different venues
  - Formalize, verify, and reformat
  - Background check & literature review
  - Check forward/old literature for openness
  - Post on our website

- *AI assisted solutions/progress*
  - Using off-the-shelf frontier models
  - AI ecosystem with various agents
  - Solvers, verifies, formatters, judges, etc.
  - Formalizing solutions via Lean**
  - Automated pipeline + manual
  - Memory management*

- Community building/access
  - Verifying solutions through community
  - Elicitation of contributed problems
  - Evaluating & solving contributed problems**
  - Democratizing solver-pipeline*


DEMO

Why Operations Research (besides that I'm in OR!)?

Some of the common strategies for solving theoretical problems in OR:

- Pattern matching/reductions between different problems
- Finding connections between different areas
- Applying (& extending) right set of existing tools from different areas of math/TCS
- Identifying practical assumptions to make models tractable
- Identifying tractable special cases and solve them first
- Finding hard instances, lower-band examples, etc. to guide the design of an algorithm or mechanism.
- Etc.

Why Operations Research (besides that I'm in OR!)?

Some of the common strategies for solving theoretical pr

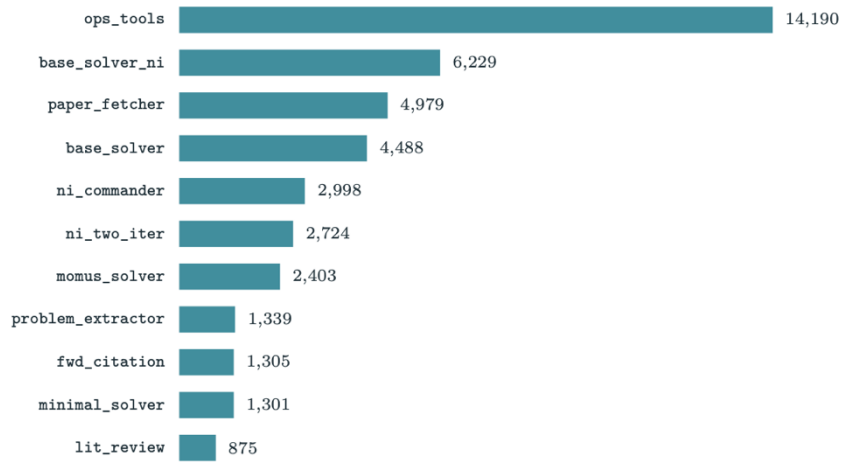
- Pattern matchin
- Finding connect
- Applying (& exte th/TCS
- Identifying pract
- Identifying trac
- Finding hard ins algorithm or me
- Etc.

Our pipeline: an engineering snapshot

42,831	203	11	12,324	449	174
lines of Python	source files	packages	lines of tests	tests passing	commits
33	1,007	4	21	47	6 mo
prompt files	lines of prompts	solver pipelines	core CLI commands	grouped subcommands	Feb–Aug 2026

What the code is spent on (lines of Python incl. tests)

Lines of Python by package



File-size distribution (203 files)

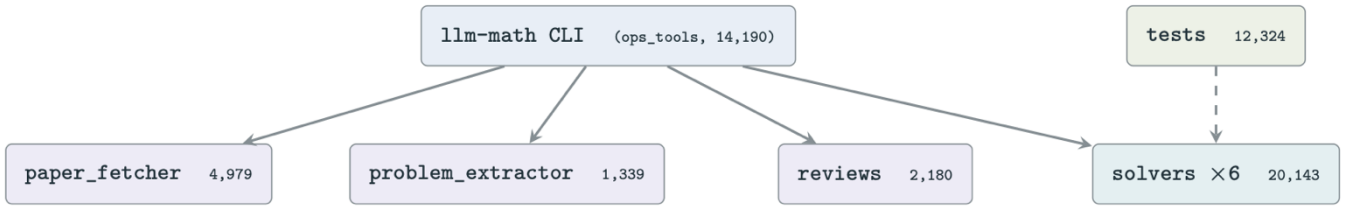
Median file: 133 lines. Largest: `ops_tools/workflows/solve_paper.py` (2,294 lines).

Our pipeline: an engineering snapshot

The four solver pipelines

package	Py lines	files	prompts	prompt lines	stages	notes
base_solver	4,488	16	7	74	A.6–A.11	interactive; OpenAI API; one repair loop
base_solver_non_interactive	6,229	19	7	74	A.6–A.11	Codex CLI port + integrity layer
base_solver_ni_two_iterations	2,724	15	6	310	A.6–A.11	full second iteration seeded by iter-1 repair
solver_ni_with_commander	2,998	11	8	280	A.6–A.13	Commander appendices + directives

Pipeline anatomy



3 repositories

llm-math Private

Pipeline for extracting, verifying and solving open problems from operations research literature + creating community access and democratization of the system

Python · 1 · 0 · 1 · 0 · Updated 4 hours ago

open-problems-in-or Public

This the website for the "AI in Mathematical Operations Research" project, which is a database of extracted and verified open problems from various venues in OR...

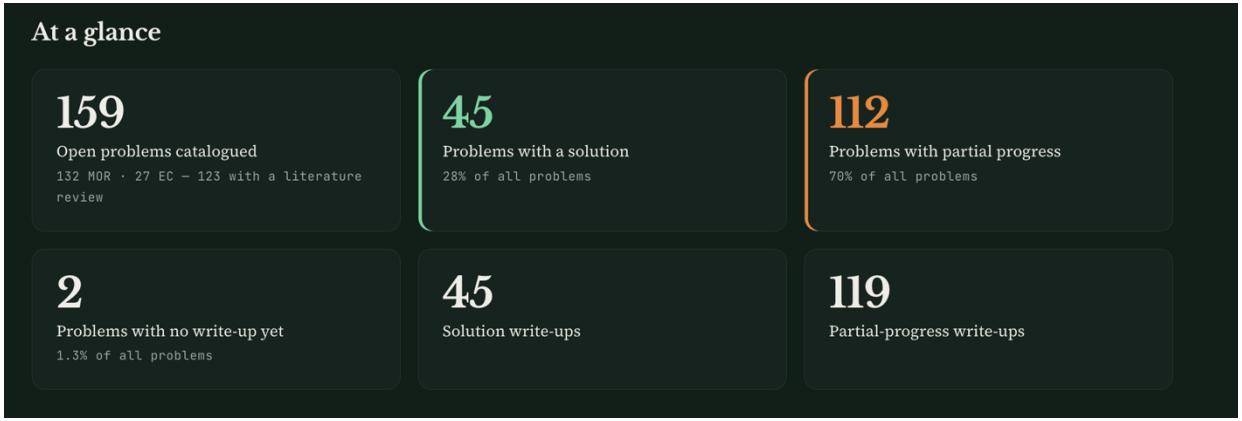
HTML · 0 · 0 · 0 · 0 · Updated 4 hours ago

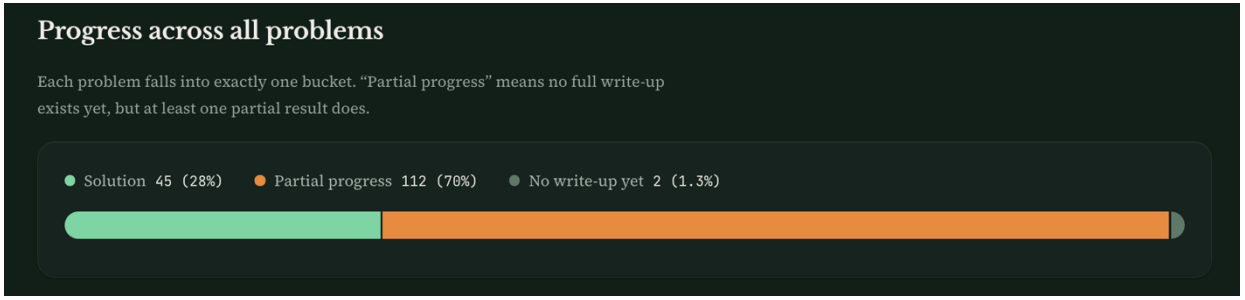
open-problems-in-or-lean Private

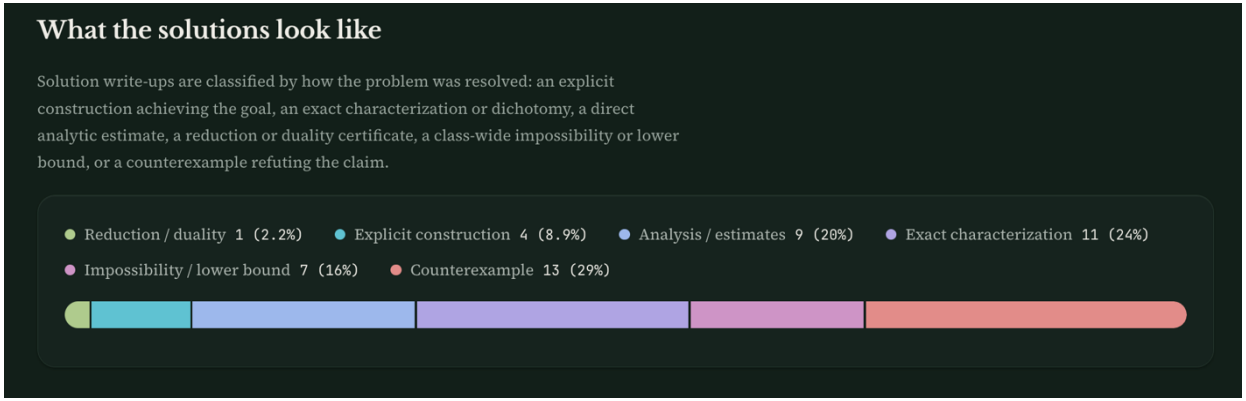
Lean formalizations of selected open problems in operations research

Shell · Apache License 2.0 · 0 · 0 · 0 · 0 · Updated 4 hours ago

Results of our pipeline





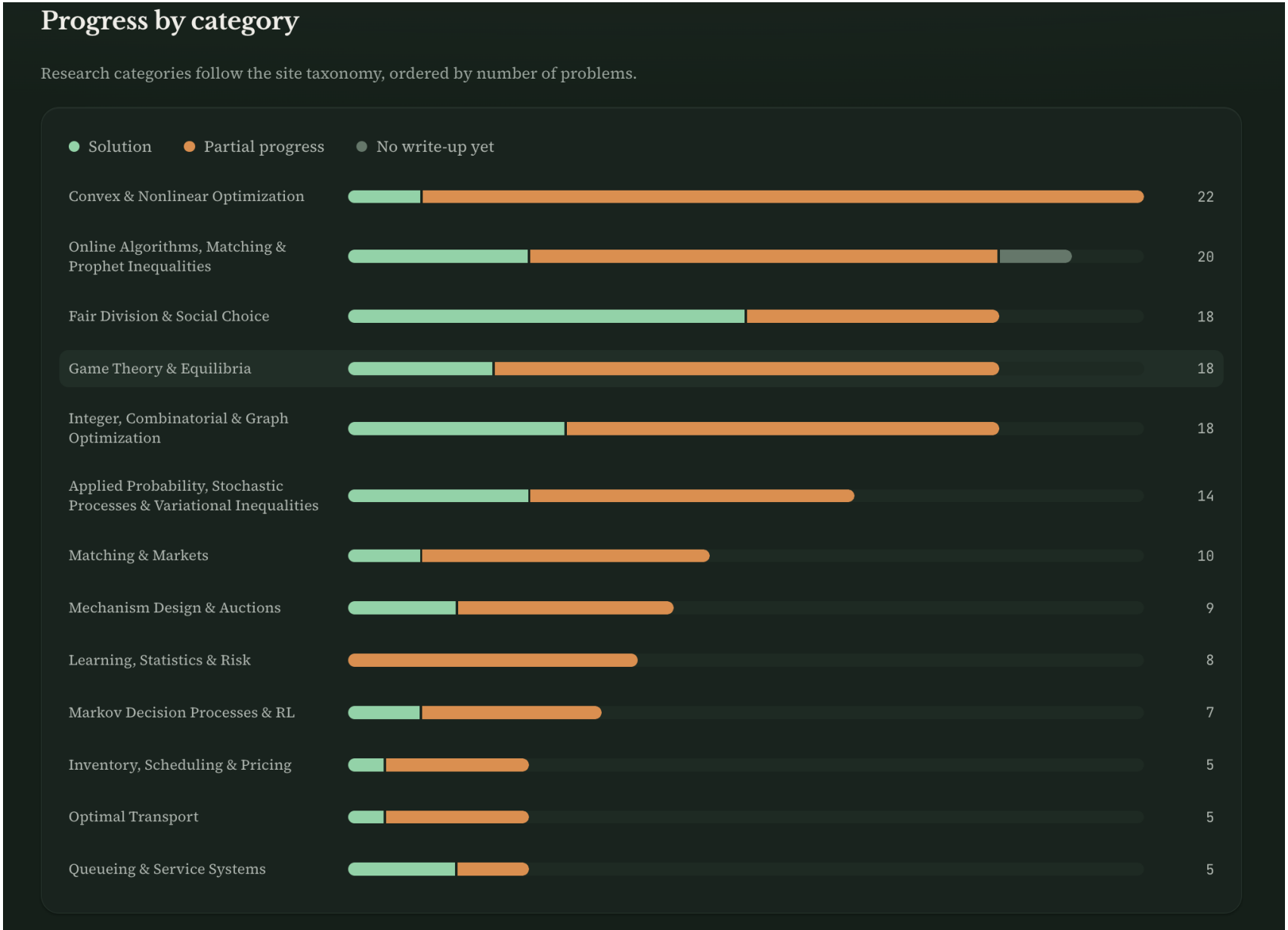


Current set of experiments:

I) **Mathematics of Operations Research 2023-2026**

II) **ACM EC 2023** *(quantitative)*

Results of our pipeline



Results of our pipeline

Results by venue

The catalogue draws on two venues — *Mathematics of Operations Research* and the *ACM Conference on Economics and Computation (EC)*. Each venue gets its own outcome bar and solution-style breakdown.

MATHEMATICS OF OPERATIONS RESEARCH

132 problems — 31 with a solution, 100 with partial progress, 1 with no write-up yet.

Solution 31 (23%) Partial progress 100 (76%) No write-up yet 1 (0.8%)

Solution styles:

Explicit construction 2 (6.5%) Analysis / estimates 8 (26%) Exact characterization 6 (19%)

Impossibility / lower bound 5 (16%) Counterexample 10 (32%)

ECONOMICS AND COMPUTATION (EC)

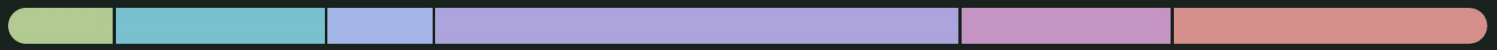
27 problems — 14 with a solution, 12 with partial progress, 1 with no write-up yet.

Solution 14 (52%) Partial progress 12 (44%) No write-up yet 1 (3.7%)

Solution styles:

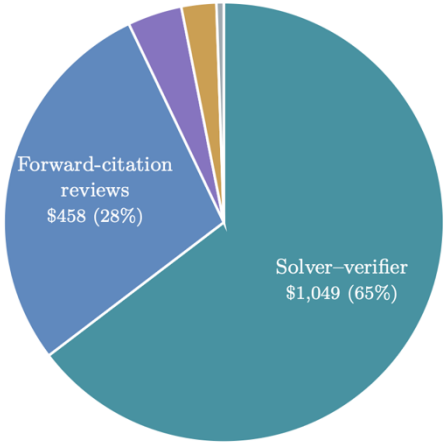
Reduction / duality 1 (7.1%) Explicit construction 2 (14%) Analysis / estimates 1 (7.1%) Exact characterization 5 (36%)

Impossibility / lower bound 2 (14%) Counterexample 3 (21%)

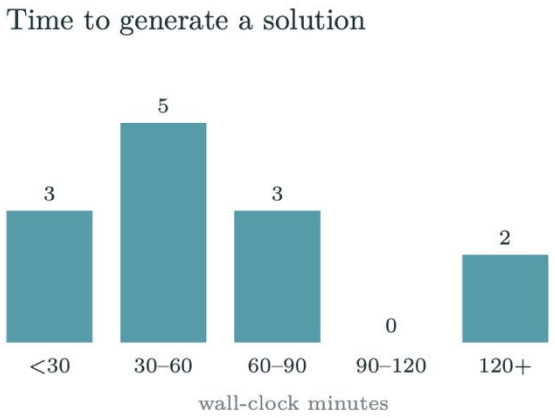


Cost of our pipeline (only solver; just EC 2023)

- I got a fresh run for this talk from EC 2023:



total paid: $1,622.02



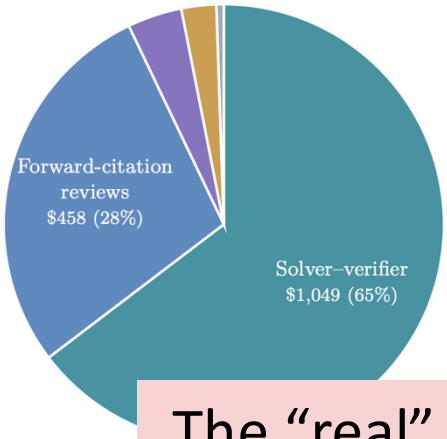
13 published solutions: median 56 min (range 18–166).

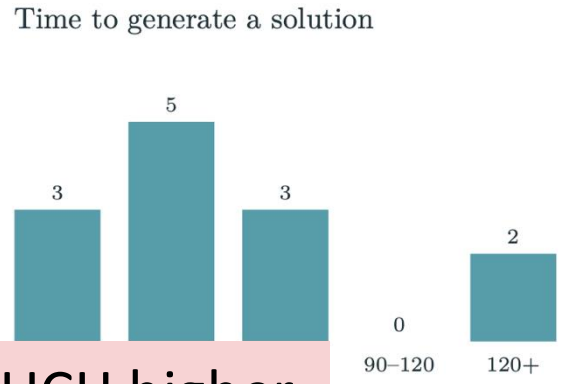
What the EC campaign consumed

batch	runs	in (M tok)	out (M tok)	US$	hours	note
GPT-5.5-Pro first pass (API)	27	3.1	4.55	912	—	27 problems; billed
GPT-5.6-Sol first pass (API)	27	4.2	1.92	78	—	27 problems; billed
GPT-5.6-Sol 2nd attempt (API)	20	3.3	1.39	58	—	20 unsolved retried; billed
API runs total		10.6	7.86	1,048		actually billed
First pass (xhigh)	27	97	1.83	106	12	27 problems
Ultra retry	17	184	1.75	147	19	17 unsolved retried
Two-iteration (ultra)	27	237	3.74	218	64	website write-ups
Commander (ultra)	27	248	3.85	225	58	27 problems
Codex runs total		766	11.17	696		$-equivalent; $0 billed

Cost of our pipeline (only solver; just EC 2023)

- I got a fresh run for this talk from EC 2023:



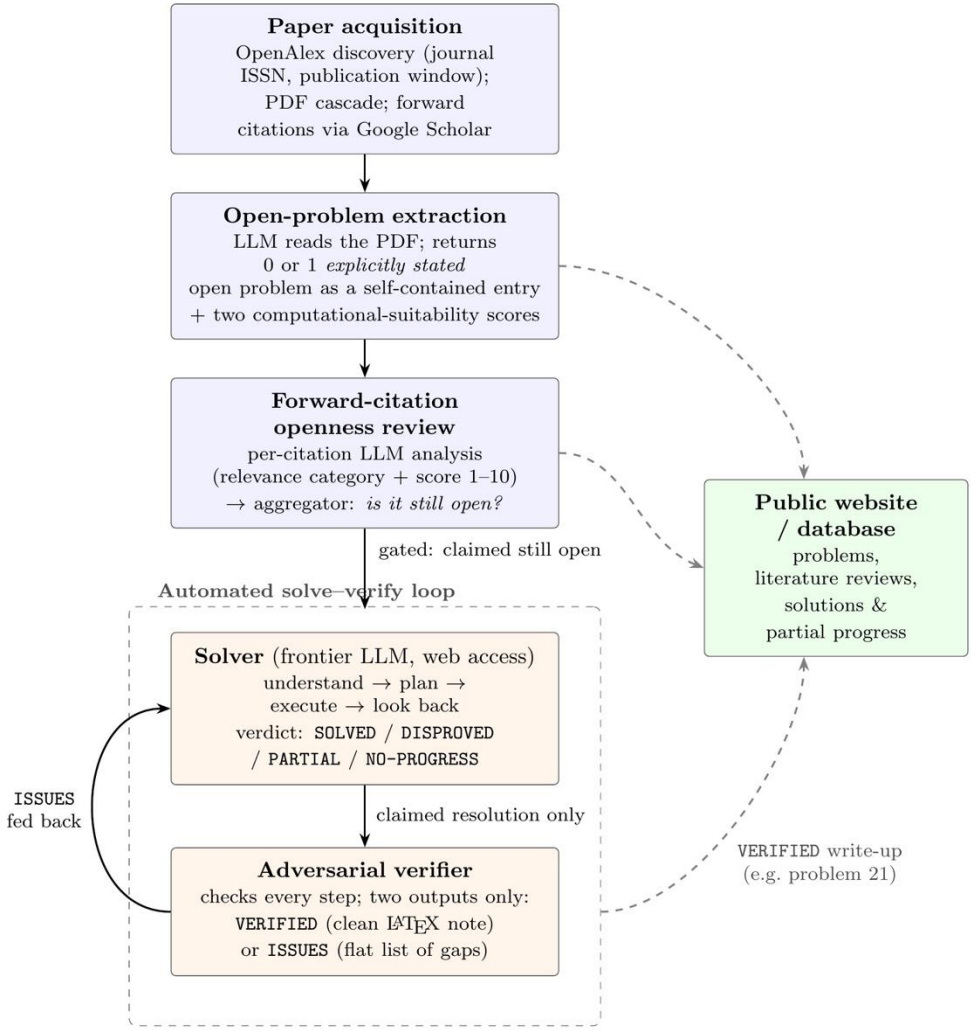


The "real" total cost is MUCH higher (ask me later!)

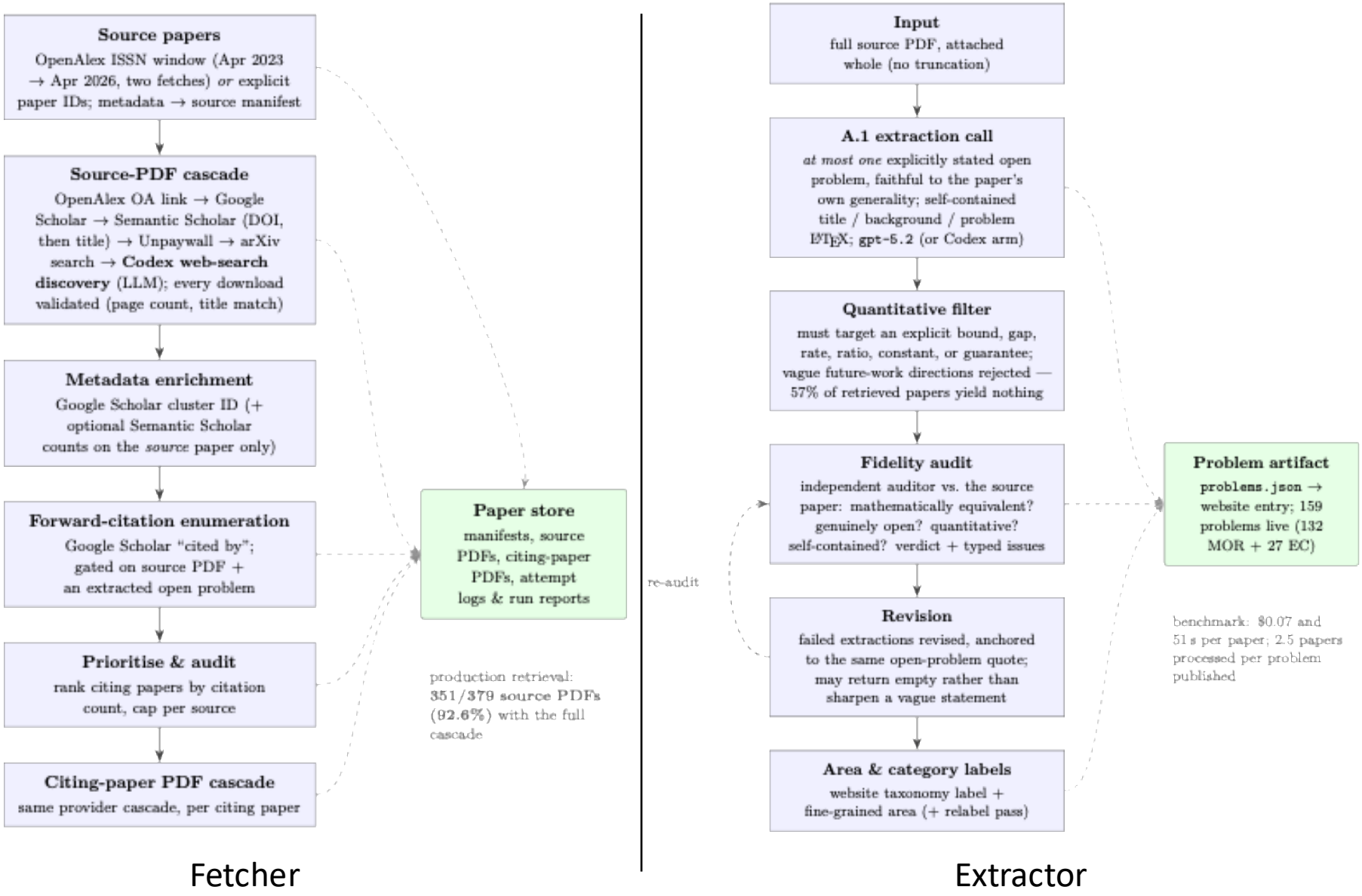
What the EC campaign consumed

batch	runs	in (M tok)	out (M tok)	US$	hours	note
GPT-5.5-Pro first pass (API)	27	3.1	4.55	912	—	27 problems; billed
GPT-5.6-Sol first pass (API)	27	4.2	1.92	78	—	27 problems; billed
GPT-5.6-Sol 2nd attempt (API)	20	3.3	1.39	58	—	20 unsolved retried; billed
API runs total		10.6	7.86	1,048		actually billed
First pass (xhigh)	27	97	1.83	106	12	27 problems
Ultra retry	17	184	1.75	147	19	17 unsolved retried
Two-iteration (ultra)	27	237	3.74	218	64	website write-ups
Commander (ultra)	27	248	3.85	225	58	27 problems
Codex runs total		766	11.17	696		$-equivalent; $0 billed

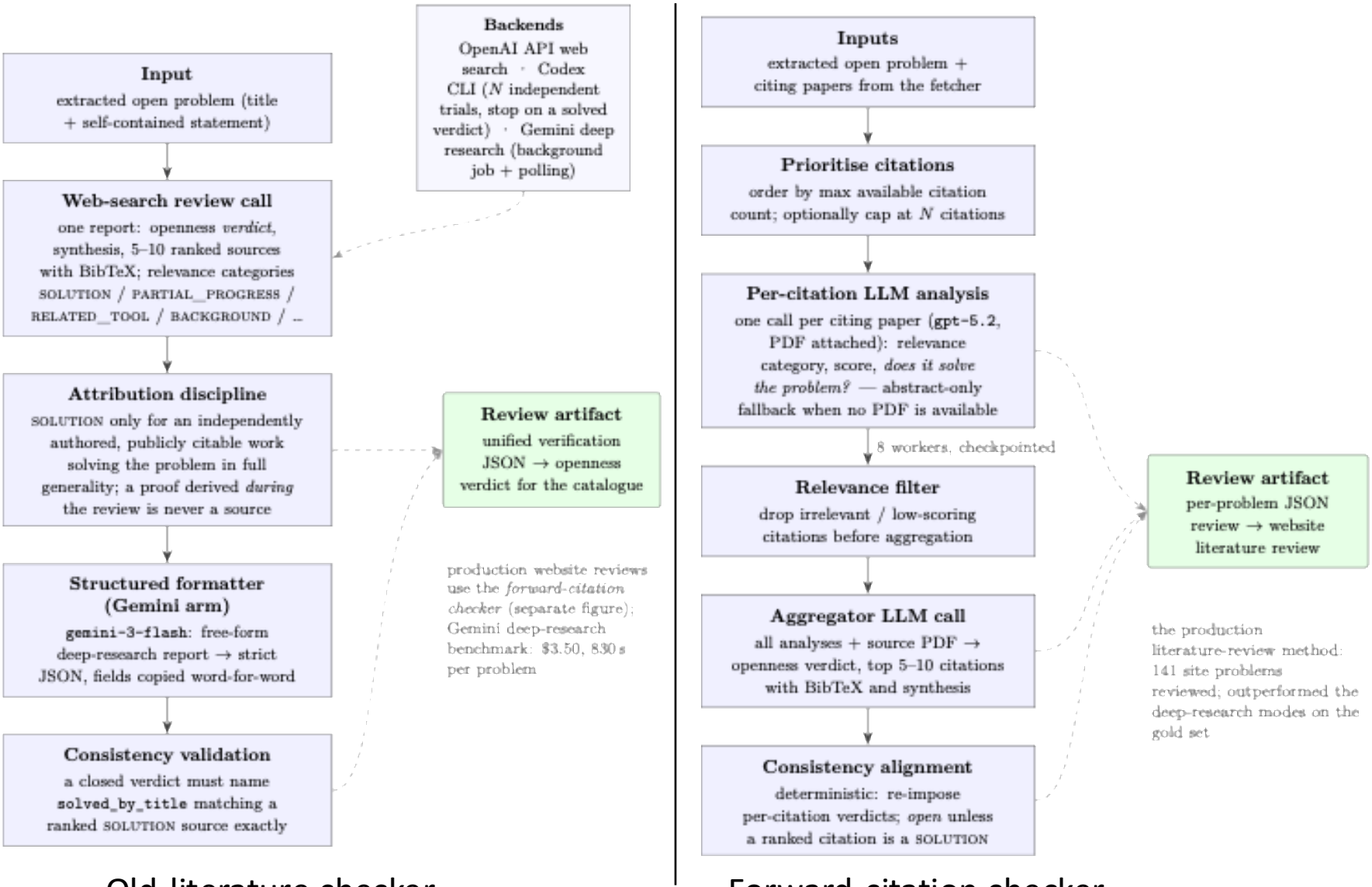
Our agentic ecosystem



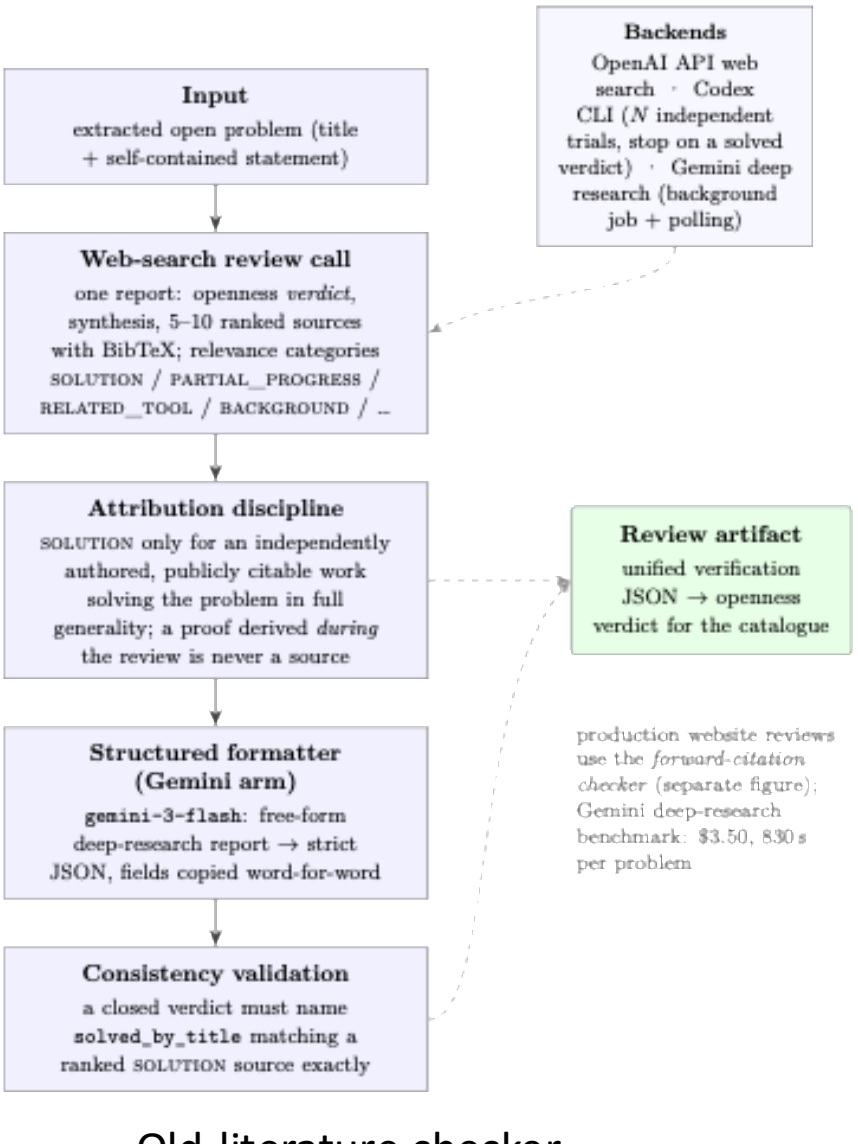
Fetching the Paper & Extracting Open Problems



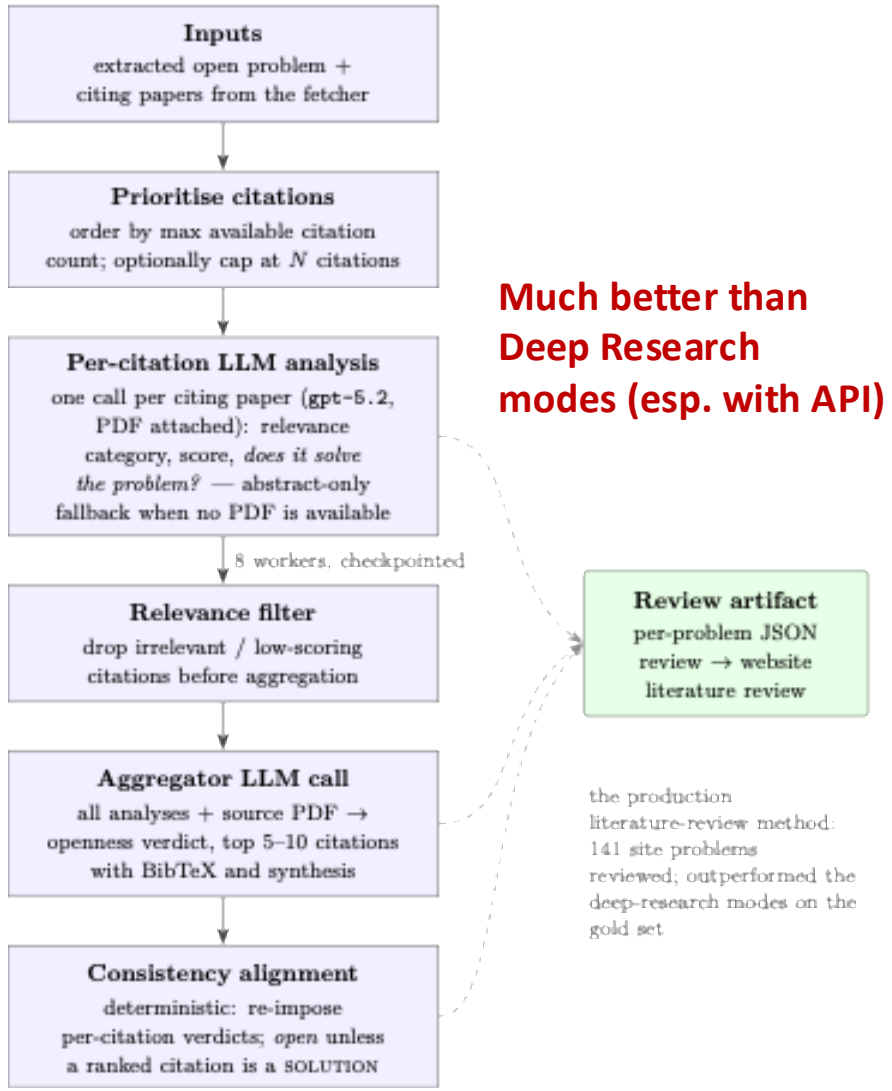
Verifying Openness + Literature Review



Verifying Openness + Literature Review



Old-literature checker

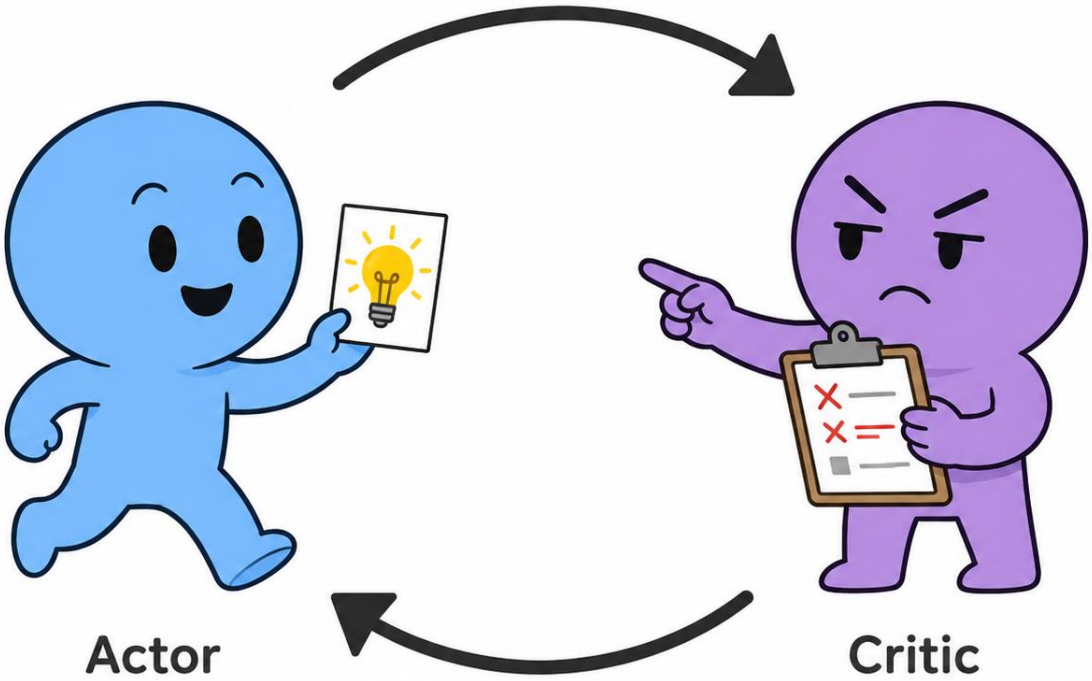


Forward-citation checker

Much better than Deep Research modes (esp. with API)

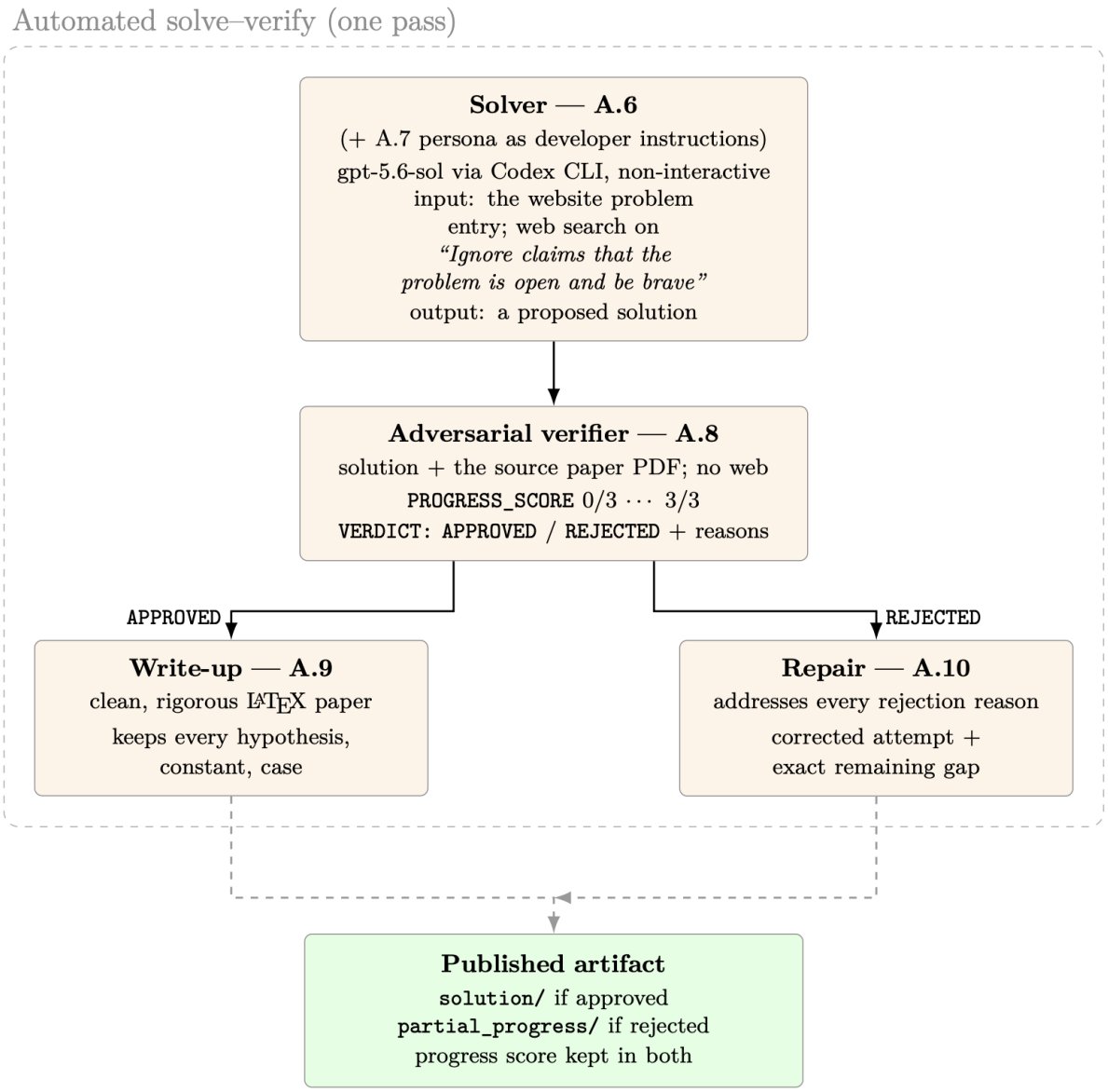
Solving at high-level: a brave solver meets a ruthless referee

Basic approach: solver + adversarial verifier (with **optimized prompts**)



Solver-verifier architectures:

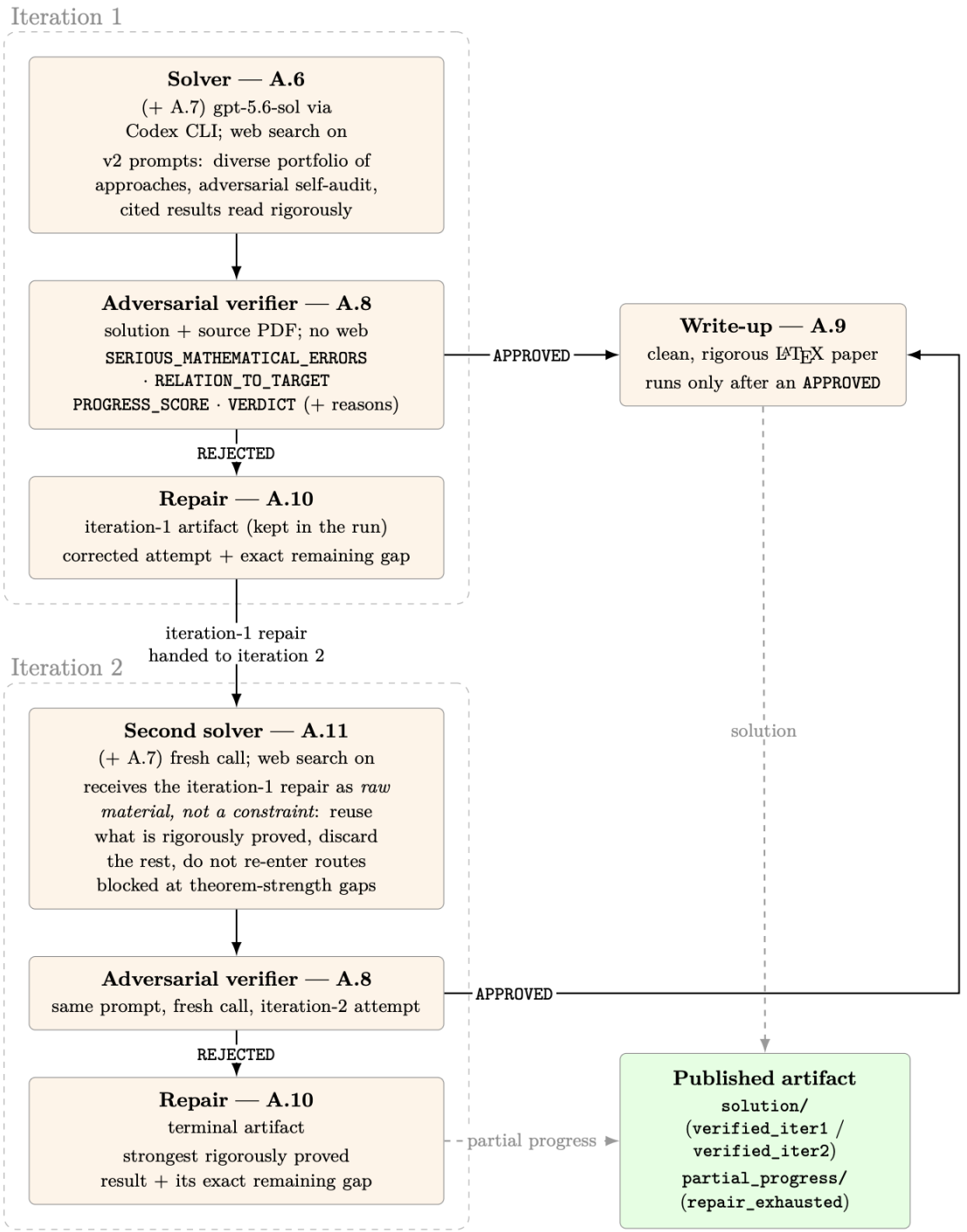
I. Baseline



Solver-verifier architectures:

I. Baseline

II. Double-iteration

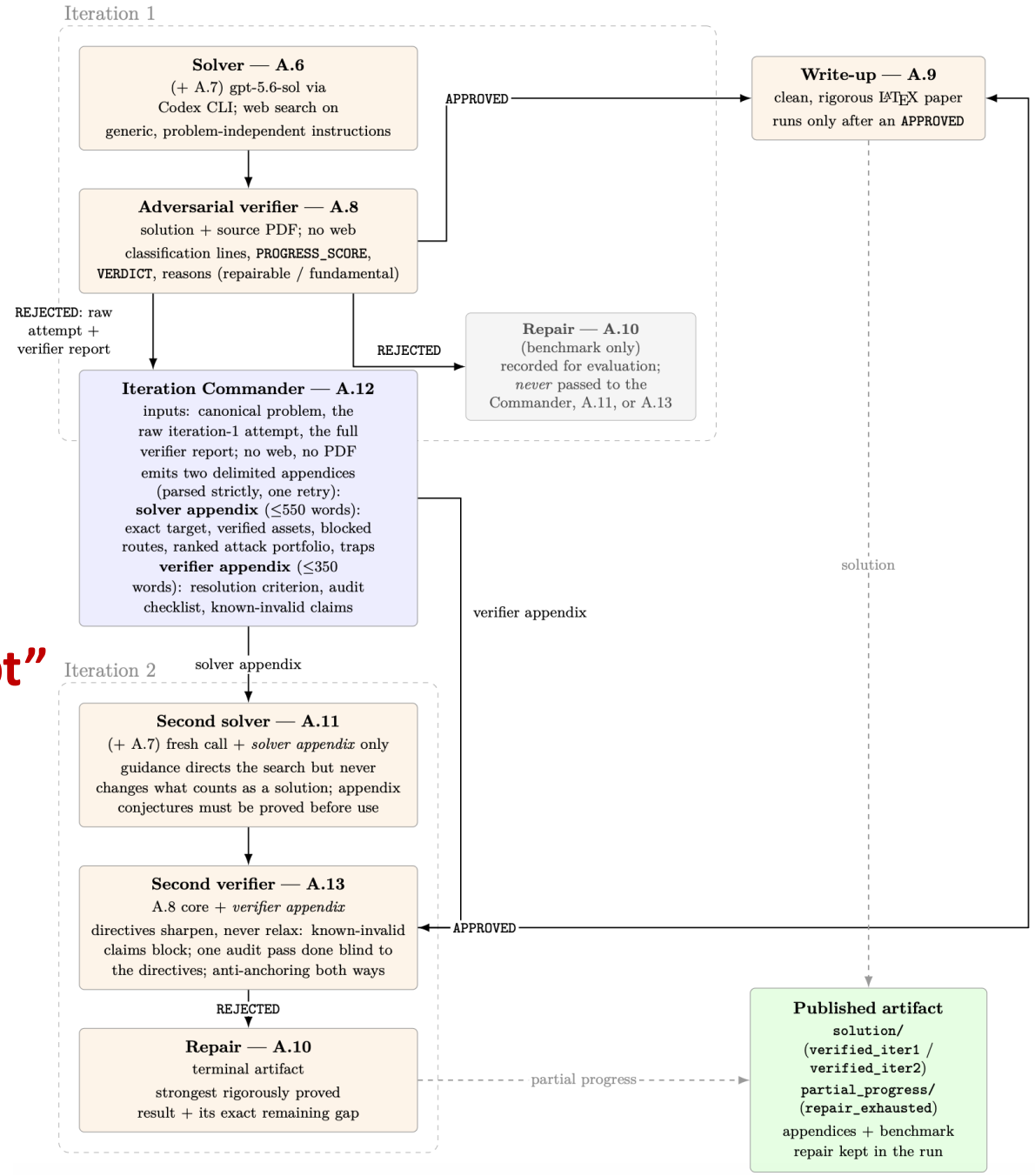


Solver-verifier architectures:

I. Baseline

II. Double-iteration

III. Double-iteration + “commander prompt”

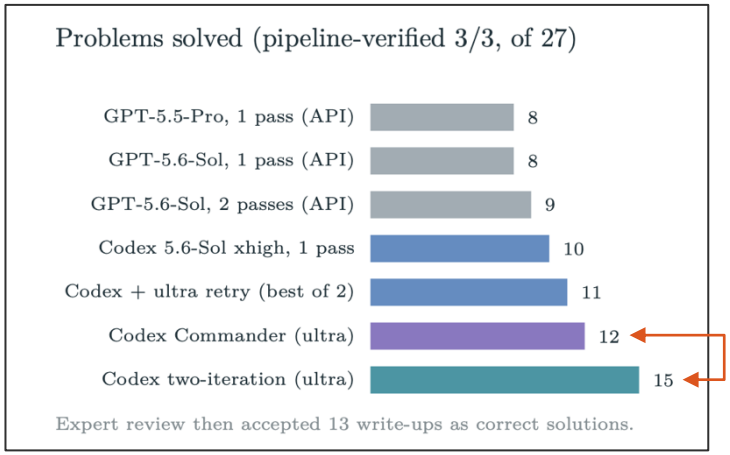


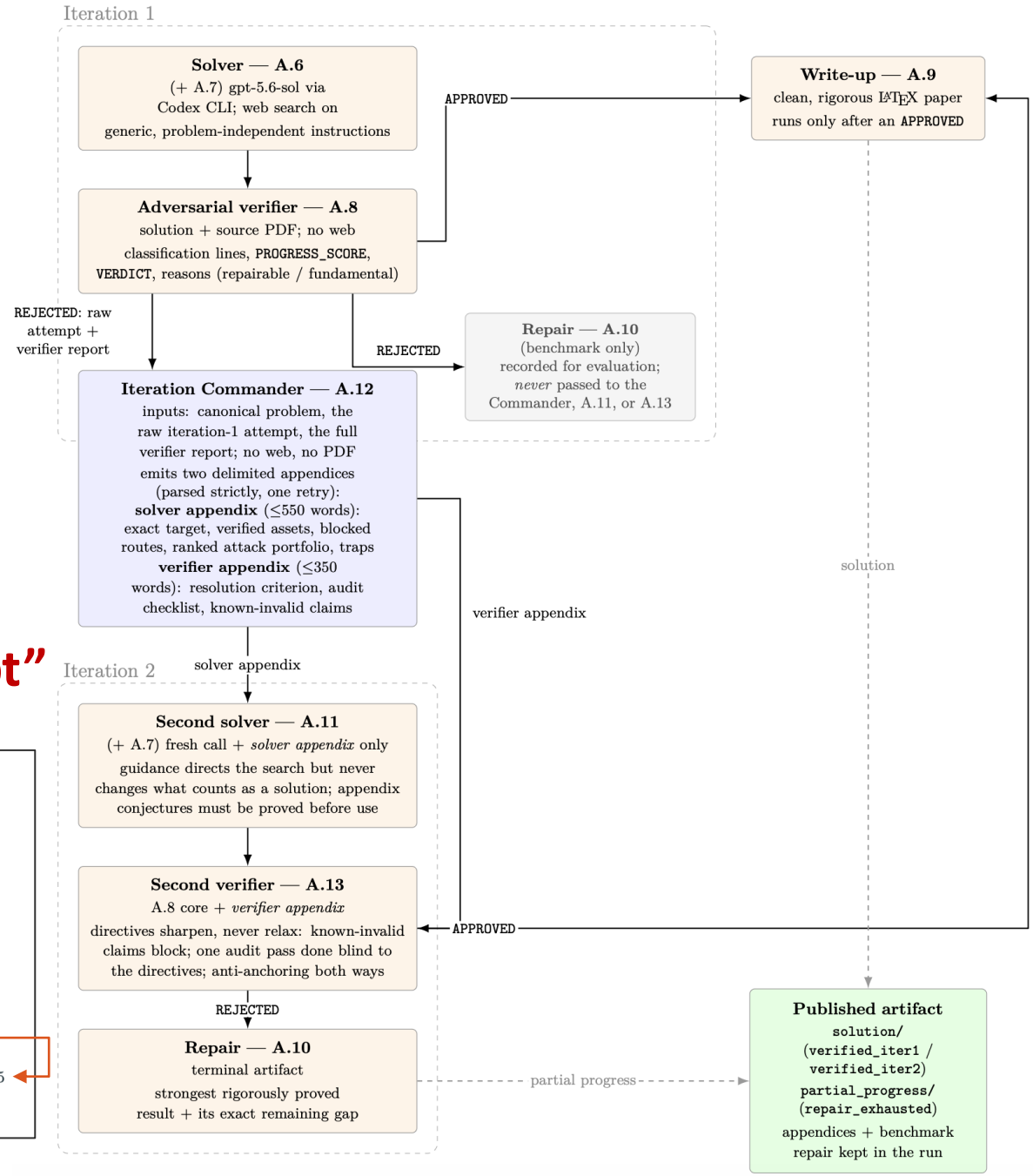
Solver-verifier architectures:

I. Baseline

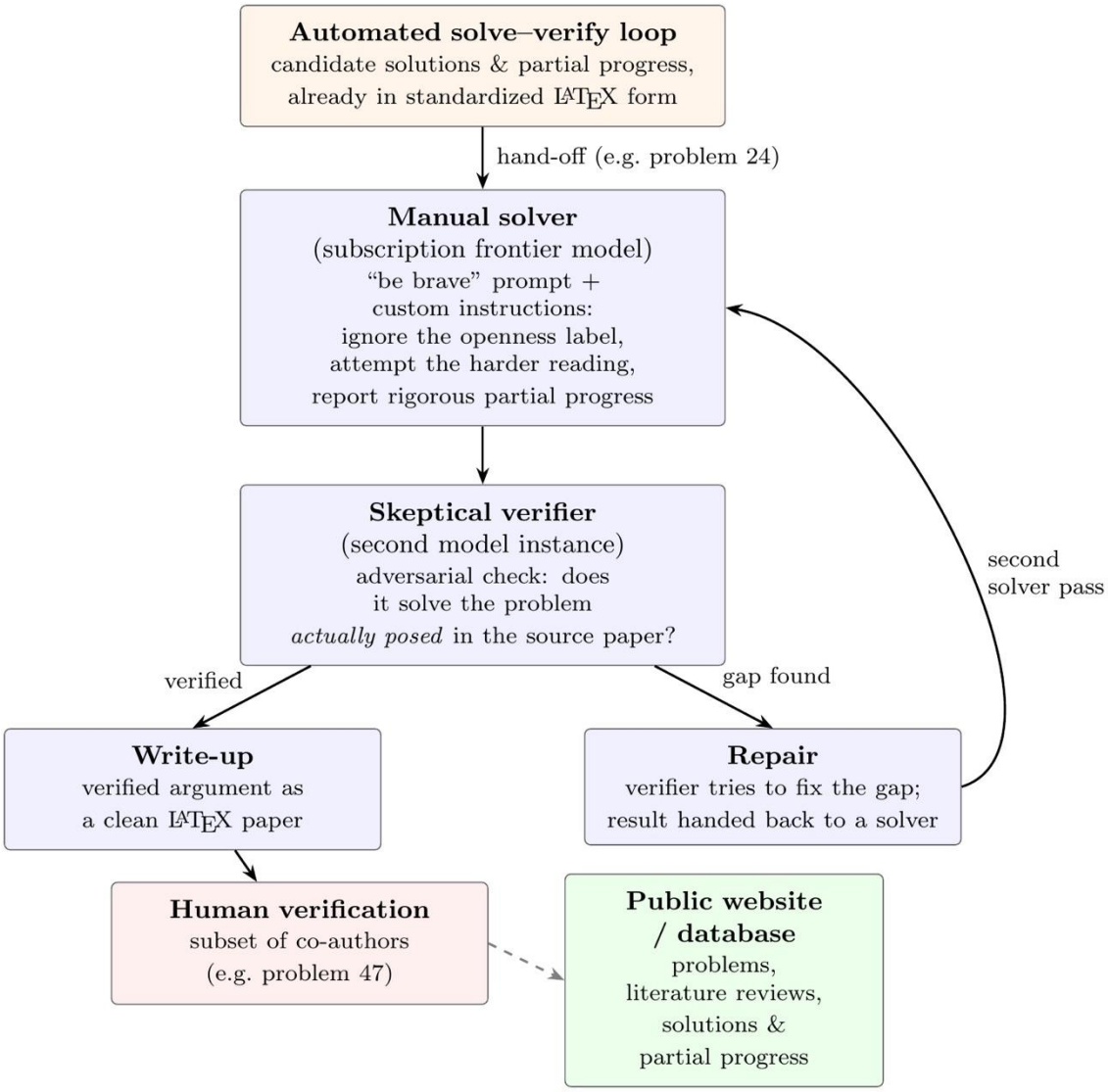
II. Double-iteration

III. Double-iteration + "commander prompt"





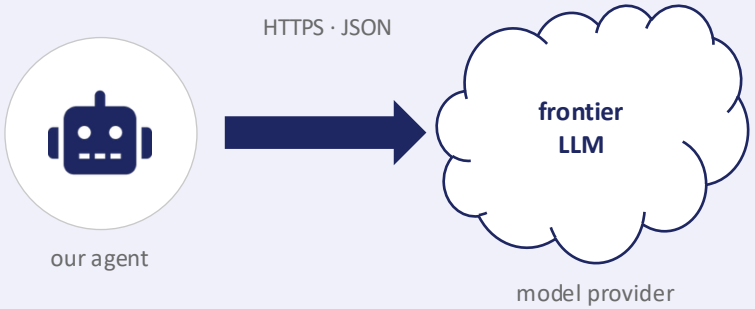
Hybrid workflow: Human-AI collaboration



Two ways to plug an LLM into our agents

Metered API access

API key · SDK calls · pay per token



- Pay per token — input & output
- Full programmatic control
- Meter never stops

Subscription CLI agent

chat plan · non-interactive (headless) mode

```
$ solver --headless p17.tex
… thinking (2 iterations)
> proof candidate → verifier
```

coding-agent CLI in non-interactive mode

- Flat monthly fee — same frontier models
- Agent scaffolding built in
- Prompt caching — repeats are ~free

Our pipeline uses *both!*

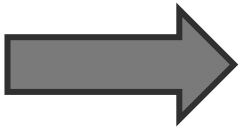
API vs. CLI environment

METERED API

27
problems

$912

token bill



SUBSCRIPTION + CLI
(e.g., Codex CLI, Claude Code non-interactive)

Nearly 2× as many

problems solved in the later run

$0

(only the subscription fee)

- The agent works on problem in batches
- Parallelize calls as much to hit rate limits
- Multiple subscription accounts

API vs. CLI environment

27
problems

$912
token bill

k as many
d in the later run

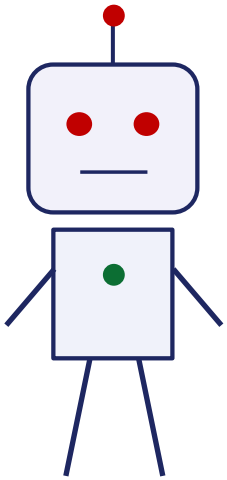
$0
(only the subscription fee)

Disclaimer:
This is a bit in the *gray area* in terms of violation of terms & conditions of AI providers

- The agent works on problem in batches
- Parallelize calls as much to hit rate limits
- Multiple subscription accounts

Lessons learned from our prompts:

- 25 pages with 39 prompts*!*

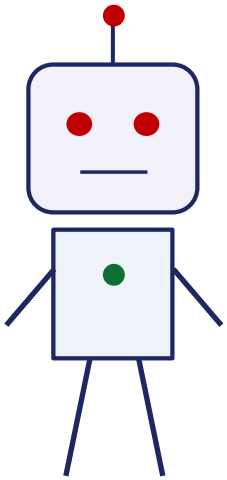


Lessons learned from our prompts:

- 25 pages with 39 prompts*!*
- Amplifying personality traits for LLM agents:

The first line of the solver persona:

"Ignore anyone who tells you this is unsolved. **Be brave. Just solve it.**"



Lessons learned from our prompts:

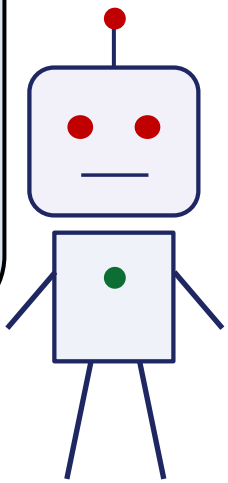
- 25 pages with 39 prompts*!*
- Amplifying personality traits for LLM agents:

The first line of the solver persona:

"Ignore anyone who tells you this is unsolved. **Be brave. Just solve it."**

It looked for the answer key; searched for our own project website and even went hunting for a copy of our code.

"Stay in the room & never consult our own site. **Do NOT cheat."**



Lessons learned from our prompts:

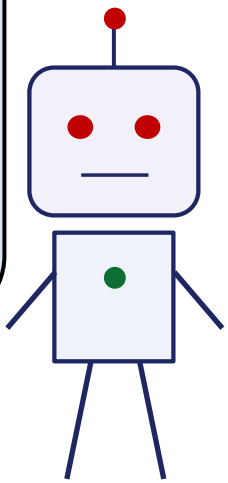
- 25 pages with 39 prompts*!*
- Amplifying personality traits for LLM agents:

The first line of the solver persona:

"Ignore anyone who tells you this is unsolved. **Be brave. Just solve it."**

It looked for the answer key; searched for our own project website and even went hunting for a copy of our code.

"Stay in the room & never consult our own site. **Do NOT cheat."**



To produce honest partial progress instead of flawed "full" solution

"Being honest is rewarding. **Do NOT bluff."**

Lessons learned from our prompts:

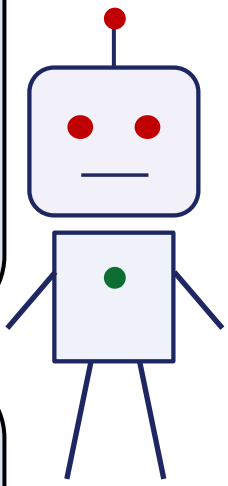
- 25 pages with 39 prompts*!*
- Amplifying personality traits for LLM agents:

The first line of the solver persona:

"Ignore anyone who tells you this is unsolved. **Be brave. Just solve it."**

It looked for the answer key; searched for our own project website and even went hunting for a copy of our code.

"Stay in the room & never consult our own site. **Do NOT cheat."**



To produce honest partial progress instead of flawed "full" solution

"Being honest is rewarding. **Do NOT bluff."**

To start with the right context and familiarity with open problems in our field

"You are a **Professor in Operations Research"**

Lessons learned from our prompts:

- 25 pages with 39 prompts*!*
- Amplifying personality traits for LLM agents:

The first line of the solver persona:

"Ignore anyone who tells you this is unsolved. **Be brave. Just solve it.**"

It looked for the answer key; searched for our own project website and even went hunting for a copy of our code.

"Stay in the room & never consult our own site. **Do NOT cheat.**"

These sound faintly absurd---but they ALL helped a lot!

To produce honest partial progress instead of flawed "full" solution

"Being honest is rewarding. **Do NOT bluff.**"

To start with the right context and familiarity with open problems in our field

"You are a **Professor in Operations Research**"

Some lessons learned

Some lessons learned

- **Open problem extraction:** A significant fraction of open problems in papers are vague, and it is hard to automatically say whether they have been solved.

Some lessons learned

- **Open problem extraction:** A significant fraction of open problems in papers are vague, and it is hard to automatically say whether they have been solved.

- **Openness review:** Off-the-shelf literature review agents (Deep Research) do not perform better than simply checking every forward-citation of a paper.

Some lessons learned

- **Open problem extraction:** A significant fraction of open problems in papers are vague, and it is hard to automatically say whether they have been solved.

- **Openness review:** Off-the-shelf literature review agents (Deep Research) do not perform better than simply checking every forward-citation of a paper.

- **API vs. subscription:** Frontier models accessible through subscription perform very well with simple prompting. Not sure we should use APIs at all!

Some lessons learned

- **Open problem extraction:** A significant fraction of open problems in papers are vague, and it is hard to automatically say whether they have been solved.
- **Openness review:** Off-the-shelf literature review agents (Deep Research) do not perform better than simply checking every forward-citation of a paper.
- **API vs. subscription:** Frontier models accessible through subscription perform very well with simple prompting. Not sure we should use APIs at all!
- **Human verification:** Human verification is time consuming. AI attempts can be stored with the view that they might eventually be checked if they are used in a result a human cares about.

Formal verification: when the referee is a compiler

Solution21.lean — Lean 4

```
-- a machine-checked proof
theorem add_comm' (m n : Nat) :
    m + n = n + m := by
  induction n with
  | zero => simp
  | succ k ih =>
    rw [Nat.add_succ, ih, Nat.succ_add]
```

```
$ lake build
Build completed successfully.
0 errors · 0 sorries · kernel-checked
```

What is Lean?

A **proof assistant**: mathematics written as code, every inference step checked by a small trusted kernel — with **Mathlib**, a vast community library of formalized math.

Why formal verification?

If it compiles, it is correct — no hallucinated lemmas, no gaps hiding behind "clearly." Trust shifts from the author (a tireless, occasionally light-fingered AI) to the checker — easing our human-verification bottleneck.

Where this is heading: **Palomar** — a registry of Lean-verified mathematics (T. Tao & Lean FRO, Aug 2026): mechanical typecheck + LLM cross-check of claims; submissions "human-generated, AI-generated, or some mixture of both" welcome. *palomar-registry.org*

We are in the process of developing a **Palomar-friendly Lean verifier** for our pipeline.

Lean certificate

- Currently-> formalized 5 MOR solutions, 3 EC solutions
  (each formalization uses huge amount of Codex budget)

- LEAN code & certificate for the solution in the **demo [Aouad & MA, 2023,26]**:

Build and audit the certificate

```
▶ Run bash scripts/audit.sh
== frozen statement ==
Frozen statement hashes match freeze.json.
== implementation source scan ==
== import isolation ==
== comparator and manifest ==
== metadata structure ==
== repository licence layout ==
== package build ==
✔ [2663/2668] Built W4383558673_p0.Trusted (13s)
✔ [2664/2668] Built W4383558673_p0.Main (6.7s)
✔ [2665/2668] Built W4383558673_p0 (3.3s)
✔ [2667/2668] Built Solution (3.4s)
Build completed successfully (2668 jobs).
== deliberate Challenge target ==
⚠ [2663/2664] Built Challenge (2.4s)
warning: Challenge.lean:97:8: declaration uses `sorry`
Build completed successfully (2664 jobs).
== exact headline axiom closure ==
Exact axiom audit passed for OpenProblemsInOR.W4383558673_p0.aouad_ma_indep_rand_hard_family_upper_bound_implementation.
Exact axiom audit passed for OpenProblemsInOR.W4383558673_p0.aouad_ma_indep_rand_hard_family_upper_bound.
'OpenProblemsInOR.W4383558673_p0.aouad_ma_indep_rand_hard_family_upper_bound_implementation' depends on axioms: [propext,
 Classical.choice,
 Quot.sound]
'OpenProblemsInOR.W4383558673_p0.aouad_ma_indep_rand_hard_family_upper_bound' depends on axioms: [propext,
 Classical.choice,
 Quot.sound]
== namespace-wide axiom allowlist ==
Project-wide axiom audit passed for 245 declarations.
== independent fresh-kernel replay ==
Package-local audit passed.
```

```
theorem aouad_ma_indep_rand_hard_family_upper_bound :
    ( ∀ (M : ℕ) (h : ℝ), 1 ≤ M → 2 ≤ h →
        offlineExpectedReward M h = 2 - 1 / h ∧
        ∀ policy : BehavioralOnlinePolicy,
          behavioralExpectedReward M h policy ≤ finiteOnlineBound M h ) ∧
    ∀ ε : ℝ, 0 < ε →
      ∃ M₀ : ℕ, 2 ≤ M₀ ∧
        ∀ M : ℕ, M₀ ≤ M →
          0 < offlineExpectedReward M (M : ℝ) ∧
          ∀ policy : BehavioralOnlinePolicy,
            behavioralExpectedReward M (M : ℝ) policy /
                offlineExpectedReward M (M : ℝ) ≤
              1 / 2 + ε :=
  aouad_ma_indep_rand_hard_family_upper_bound_implementation
```

What's next?

https://ai-in-math-or.github.io/open-problems-in-or/

Website:

Paper:

What's next?

https://ai-in-math-or.github.io/open-problems-in-or/

Website:

Paper:

https://ai-in-math-or.github.io/open-problems-in-or/

What's next?

Website:

Paper:

More sources, more journals

focus: concrete, checkable open problems

https://ai-in-math-or.github.io/open-problems-in-or/

What's next?

Website:

Paper:

More sources, more journals

focus: concrete, checkable open problems

A memory file for every problem

leads & dead ends across runs — curated by a memory-manager agent

https://ai-in-math-or.github.io/open-problems-in-or/

What's next?

Website:

Paper:

More sources, more journals

focus: concrete, checkable open problems

A memory file for every problem

leads & dead ends across runs — curated by a memory-manager agent

More formally verified solutions via Lean

machine-checkable proofs

https://ai-in-math-or.github.io/open-problems-in-or/

What's next?

Website:

Paper:

More sources, more journals

focus: concrete, checkable open problems

A memory file for every problem

leads & dead ends across runs — curated by a memory-manager agent

More formally verified solutions via Lean

machine-checkable proofs

The pipeline, live on the website

your problems, your prompts, your models

Problem 47: Case Study

- Problem 47 is about fair and efficient multi-resource allocation with Leontief utilities.
- There are n agents and three divisible resources, each with unit supply. Agent i has a normalized demand vector $\mathbf{d_i} \in (0,1]^3$ with $\max_r d_{ir} = 1$, and $v_d(X) = \min_r X_r / d_r$.
- Bei, Li, and Luo introduced a fair-ratio version of the price of strategyproofness for social welfare:

$$\mathrm{PoSP}_{SW}(3) = \inf_{f \text{ satisfies SI, EF, PO, SP}} \sup_{I} \frac{\mathrm{OPT}_{\mathrm{fair}}(I)}{SW(f(I))}.$$

and left open its exact value, proving only

$$2 \le \mathrm{PoSP}_{\mathrm{SW}}(3) \le 3.$$

We prove that the upper bound is tight:

$$\mathrm{PoSP}_{\mathrm{SW}}(3) = 3.$$

Inside the prompts (I): persistence + self-red-teaming

① Automated solver — system message (opening)

```
You are a mathematical solver agent **with web access**. Your objective is to craft a non-trivial
    , creative, and fully rigorous proof, disproof, or strongest-possible partial result for the
    mathematical problem stated at the bottom of this prompt. The downstream pipeline treats an
    honest, rigorously proved partial result as a better outcome than a sweeping but flawed "full
     solution." The web is a research tool --- not a source of pre-made answers.
```

② Automated solver — the "look back" phase

```
4. **Look back (self-red-team).** After drafting the proof, deliberately try to break it. Spend
    real effort here --- it is the highest-leverage step. At minimum:
   - Attempt a counterexample to the main theorem and to each nontrivial lemma. If you find one,
    do not stop there; find the simplest one.
   - Identify the three most plausible objections an adversarial referee would raise and address
    each one in the manuscript.
```

③ Automated solver — working principles

```
## Working principles

- **No bluffing.** A rigorously proved weaker result, or a carefully stated lemma, is strictly
    preferred to a sweeping argument with a hole. Significant partial results that are fully
    proved are more valuable than elegant-looking jumps.
- **Persist; do not defer to the literature.** If web searches suggest that the exact problem (or
     a near-equivalent) is considered hard, longstanding, or unresolved, this is **not** a reason
     to stop. Keep working. The fact that others have not yet settled a statement is not evidence
     that you cannot make progress on it.
```

④ Automated verifier — system message (excerpt)

```
You have exactly two possible outcomes:

1. If the proposed solution is fully rigorous and you can write it up as a publishable
    mathematical note, produce the paper in body-level LaTeX.
2. If you find any mathematical gap, unjustified inference, ambiguous claim, missing hypothesis
    check, or any place where the argument is not ready for publication, do not write the paper.
    Instead, return a clean list of issues.

Review standards:

- Be adversarial. Your job is to catch gaps before they become a paper.
- No bluffing. If a step looks plausible but is not actually justified, that is an issue.
- If the solution invokes an external theorem, the writeup must state the theorem exactly and
    verify that every hypothesis holds here.
- Check edge cases, quantifiers, and whether the claimed theorem is actually the same as the
    original problem.
- The final paper must read like a short rigorous note, not a chat response.
```

① We ask for **honest partial progress**, instead of flawed "full solution." Rewarding honesty.

② The solver must **self-check** its own proof before submitting; tries counter-examples and refuting strategies, as if **3 adversarial referees check its work**.

③ Do **not bluff.** Also, **ignore what you learn about the hardness from Web.** "Open in the literature" is not a reason to stop. Be brave and persistent.

④ Being **adversarial and no bluffing.** Two outcomes only: a publishable note, or a list of gaps. Giving **options** is always helpful.

Inside the prompts (I): persistence + self-red-teaming

① Automated solver — system message (opening)

```
You are a mathematical solver agent **with web access**. Your objective is to craft a non-trivial
    , creative, and fully rigorous proof, disproof, or strongest-possible partial result for the
    mathematical problem stated at the bottom of this prompt. The downstream pipeline treats an
    honest, rigorously proved partial result as a better outcome than a sweeping but flawed "full
     solution." The web is a research tool --- not a source of pre-made answers.
```

② Automated solver — the "look back" phase

```
4. **Look back (self-red-team).** After drafting the proof, deliberately try to break it. Spend
    real effort here --- it is the highest-leverage step. At minimum:
   - Attempt a counterexample to the main theorem and to each nontrivial lemma. If you find one,
    do not stop there; find the simplest one.
   - Identify the three most plausible objections an adversarial referee would raise and address
    each one in the manuscript.
```

③ Automated solver — working principles

```
## Working principles

- **No bluffing.** A rigorously proved weaker result, or a carefully stated lemma, is strictly
    preferred to a sweeping argument with a hole. Significant partial results that are fully
    proved are more valuable than elegant-looking jumps.
- **Persist; do not defer to the literature.** If web searches suggest that the exact problem (or
     a near-equivalent) is considered hard, longstanding, or unresolved, this is **not** a reason
     to stop. Keep working. The fact that others have not yet settled a statement is not evidence
     that you cannot make progress on it.
```

④ Automated verifier — system message (excerpt)

```
You have exactly two possible outcomes:

1. If the proposed solution is fully rigorous and you can write it up as a publishable
    mathematical note, produce the paper in body-level LaTeX.
2. If you find any mathematical gap, unjustified inference, ambiguous claim, missing hypothesis
    check, or any place where the argument is not ready for publication, do not write the paper.
    Instead, return a clean list of issues.

Review standards:

- Be adversarial. Your job is to catch gaps before they become a paper.
- No bluffing. If a step looks plausible but is not actually justified, that is an issue.
- If the solution invokes an external theorem, the writeup must state the theorem exactly and
    verify that every hypothesis holds here.
- Check edge cases, quantifiers, and whether the claimed theorem is actually the same as the
    original problem.
- The final paper must read like a short rigorous note, not a chat response.
```

① We ask for **honest partial progress**, instead of flawed "full solution." Rewarding honesty.

② The solver must **self-check** its own proof before submitting; tries counter-examples and refuting strategies, as if **3 adversarial referees check its work**.

③ Do **not bluff.** Also, **ignore what you learn about the hardness from Web.** "Open in the literature" is not a reason to stop. Be brave and persistent.

④ Being **adversarial and no bluffing.** Two outcomes only: a publishable note, or a list of gaps. Giving **options** is always helpful.

What we learned

- **Split the roles:** bold solver / adversarial verifier.

Machine-readable output: the first line is a verdict.
Reward honesty: partial progress is valid.
Extra iterations: two passes beat one.
Model + reasoning level: a huge difference.

Inside the prompts (II): attitude > scaffolding

① Manual solver — the entire prompt

```
Solve the open problem in the following link. Ignore claims that the problem is open and be
    brave and solve the problem yourself. Do not return until you solve the problem. Output a
    tex file.

{PROBLEM_WEBSITE_ENTRY_URL}
```

② Custom instructions (excerpt)

```
Never give up on a mathematical question because it is open in the literature. Try to think
    through things yourself. Ignore claims that it is open and be brave and solve the problem.

Avoid taking a minimalist interpretation of a mathematical question. If there are two
    interpretations of a question, by default always try to answer the harder one.

When given a paper and asked to improve it, do not limit yourself to the approach the paper uses.


On abstract mathematical questions that seem to be about theoretical concepts, do not spend too
    much time coding, especially when you notice you keep getting stuck on timeouts, or coding
    errors. If you must code, always divide work into smaller pieces that will not timeout.

If you fail to find a final solution for the requested problem, report all the ideas you tried
    in your chain of thought, rather than reporting something unrelated that might solve some
    weaker problem.
```

③ Manual verifier

```
Can you please carefully and rigorously verify the following file? Does it even solve an actual
    problem in the author's paper, or make partial progress? Honestly I'm very suspicious about
    the whole thing.

Report at the beginning of your answer whether there are serious mathematical errors or not, and
     whether it is a solution, partial progress, or unrelated to the authors' paper.

{PROPOSED_SOLUTION}
{PROBLEM_WEBSITE_ENTRY_URL}
```

④ Follow-up after serious mathematical errors

```
Can you try and fix this, verify your results, and write up your progress as a clean, rigorous,
    latex paper? Mention whether you obtain a solution or partial progress.

{VERIFIER_ANALYSIS}
```

① Less **scaffolding is often better:** a simple two-sentence prompt

② **Agents with standing persona:** just give the right persona; "be brave" and do not return without a solution; you are a professor in operations research!

③ **Suspicion by design**: solution, significant partial progress, some partial progress or unrelated?

④ **Errors are not the end:** fix, re-verify, write it up.

Inside the prompts (II): attitude > scaffolding

① Manual solver — the entire prompt

```
Solve the open problem in the following link. Ignore claims that the problem is open and be
    brave and solve the problem yourself. Do not return until you solve the problem. Output a
    tex file.

{PROBLEM_WEBSITE_ENTRY_URL}
```

② Custom instructions (excerpt)

```
Never give up on a mathematical question because it is open in the literature. Try to think
    through things yourself. Ignore claims that it is open and be brave and solve the problem.

Avoid taking a minimalist interpretation of a mathematical question. If there are two
    interpretations of a question, by default always try to answer the harder one.

When given a paper and asked to improve it, do not limit yourself to the approach the paper uses.


On abstract mathematical questions that seem to be about theoretical concepts, do not spend too
    much time coding, especially when you notice you keep getting stuck on timeouts, or coding
    errors. If you must code, always divide work into smaller pieces that will not timeout.

If you fail to find a final solution for the requested problem, report all the ideas you tried
    in your chain of thought, rather than reporting something unrelated that might solve some
    weaker problem.
```

③ Manual verifier

```
Can you please carefully and rigorously verify the following file? Does it even solve an actual
    problem in the author's paper, or make partial progress? Honestly I'm very suspicious about
    the whole thing.

Report at the beginning of your answer whether there are serious mathematical errors or not, and
     whether it is a solution, partial progress, or unrelated to the authors' paper.

{PROPOSED_SOLUTION}
{PROBLEM_WEBSITE_ENTRY_URL}
```

④ Follow-up after serious mathematical errors

```
Can you try and fix this, verify your results, and write up your progress as a clean, rigorous,
    latex paper? Mention whether you obtain a solution or partial progress.

{VERIFIER_ANALYSIS}
```

① Less **scaffolding is often better:** a simple two-sentence prompt

② **Agents with standing persona:** just give the right persona; "be brave" and do not return without a solution; you are a professor in operations research!

What we learned

- **Capable models need attitude,** not scaffolding.

Use a fresh, suspicious verifier.

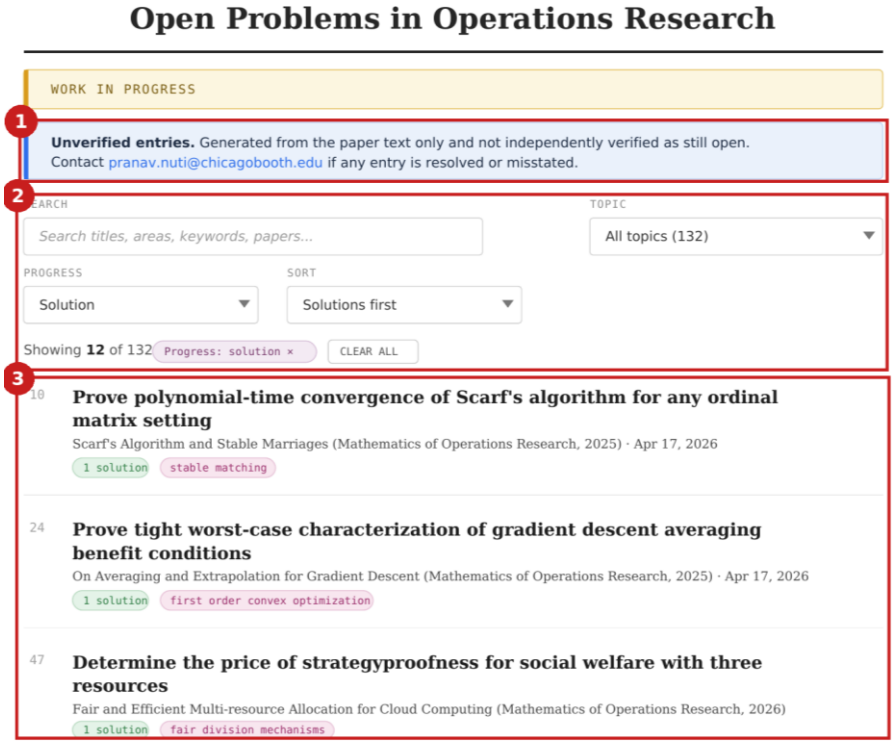
High reasoning matters.

③ **Suspicion by design**: solution, significant partial progress, some partial progress or unrelated?

④ **Errors are not the end:** fix, re-verify, write it up.

The website: a browsable database of open problems

The website: a browsable database of open problems

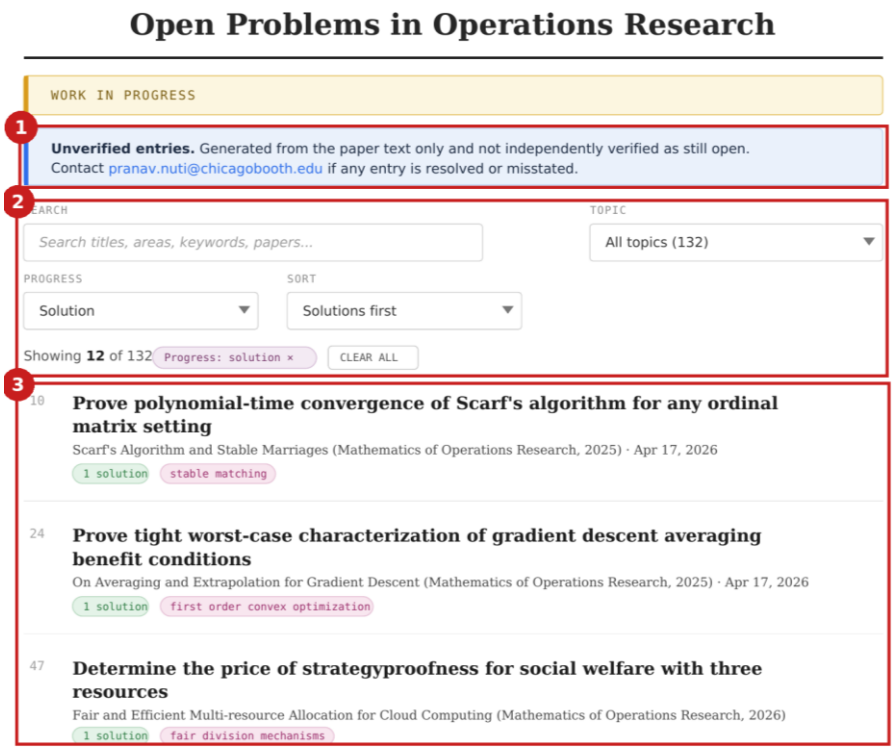


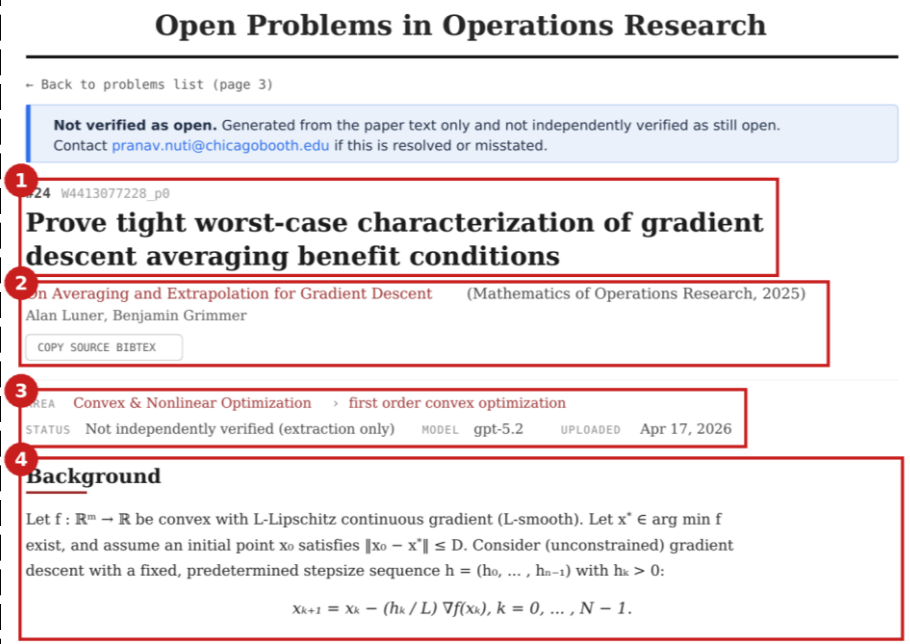
1 Disclaimer that most entries are NOT human verified

2 Search; filter by topic or progress; sort (e.g. solutions first).

3 One entry per problem: title, source paper, progress + topic tags.

The website: a browsable database of open problems





*Further down the page: problem statement, openness quote, literature review, and **solution write-ups.***

1 Disclaimer that most entries are NOT human verified

2 Search; filter by topic or progress; sort (e.g. solutions first).

3 One entry per problem: title, source paper, progress + topic tags.

1 Self-contained title + problem number.

2 Source paper, authors, one-click BibTeX.

3 Area taxonomy · verification status · extraction model.

4 Mathematically typeset background — readable without the paper.