Solving Open Problems in Operations Research Using AI*

Eric Fithian Rad Niazadeh Pranav Nuti

The University of Chicago Booth School of Business

Working Draft — June 3, 2026

Abstract

We describe a pipeline for AI-assisted proving in operations research. We use an LLM to create a large collection of open problems extracted from papers published over the last three years in the journal *Mathematics of Operations Research*. For each candidate open problem, we generate a literature review by checking each paper that cites the paper that introduced the problem, using it to judge whether the problem is in fact still open. We then use a frontier LLM to attempt to solve these problems, using a solve–verify loop, and we provide the resulting attempts, if any, to a second, 'manual' workflow. We organize all of these problems, literature reviews, and solution attempts (including partial progress) in a public database at https://pranav-nuti.github.io/open-problems-in-or/.

From the 132 open problems in the database, our workflows have currently produced candidate complete solutions for 12 problems and rigorous partial progress for a further 8. This includes problems solved entirely by our automated pipeline (problem 21 in our database), problems solved with a mix of the manual and automated pipelines (problem 24), and problems solved entirely by a manual pipeline (problem 47). We note that the manual pipeline is manual due to its dependence on using a subscription rather than API access, but is substantially mechanical in nature, and does not require real human expertise.

While we are still in the process of rigorously verifying all of these attempts, preliminary evidence suggests that frontier LLMs are especially promising as a tool for proposing examples and counterexamples to solve open problems in the operations research literature. We provide one such example that we have rigorously verified—to open problem 47 in our database, for fair multi-resource allocation with Leontief utilities.

For a fixed number m of divisible resources, let us denote by $\mathrm{PoSP}_{\mathrm{SW}}(m)$ the *price of strategyproofness* in this setting, i.e., the best possible fair-ratio for utilitarian social welfare among mechanisms satisfying the share incentive (SI), envy freeness (EF), Pareto optimality (PO), and strategyproofness (SP) properties, where the benchmark is the best feasible allocation satisfying the SI and EF properties. Bei, Li, and Luo introduced this benchmark and left open its exact value for $m = 3$, proving only $2 \le \mathrm{PoSP}_{\mathrm{SW}}(3) \le 3$. We prove:

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

Our lower bound is in fact stronger: every share-incentive and strategyproof mechanism has fair-ratio 3.

This document was generated in substantial part by an LLM and has been verified and edited by a subset of (human) authors.

*Prepared for the workshop AI-driven research in Econ-CS. Project website: https://pranav-nuti.github.io/open-problems-in-or/. We acknowledge funding from the Center for Applied Artificial Intelligence at the University of Chicago Booth School of Business.

Contents

1 Introduction and motivation

Large language models have recently displayed significantly improved abilities at solving tightly-scoped mathematical problems. Just in the last few months, an agent built on a frontier reasoning model autonomously solved a majority of the problems on the inaugural *FirstProof* challenge [1], and frontier models have been used to disprove the long-standing unit-distance conjecture [2].

Almost all of these successes have been in pure mathematics. Furthermore, many successes have come from individual researchers experimenting with AI tools on problems they are personally interested in. We are thus motivated to ask the question:

> *How powerful are off-the-shelf frontier models at solving open problems in the field of operations research, and can we build an agentic pipeline that solves such problems at scale?*

Some features of LLMs suggest that this question can be answered affirmatively. Their strengths are quite different in shape from a human expert's: the breadth of their knowledge, and their ability to tirelessly run computations and try variation after variation, cannot be matched by even the fastest human. Furthermore, one domain in which LLMs are *unusually* strong is counterexample discovery—which is precisely where human skill tends to be weakest. We become invested in problems and come to really want statements to be true. We feel that a proof would be beautiful, while a counterexample might be ugly. We find brute-force searches for counterexamples tiring and uninteresting. AI tools do not experience wishful thinking, wanting, beauty, tiredness, or boredom.

How can a research community best harness these abilities to improve its collective understanding of the literature? Attempts by individual researchers trying out AI tools have an ephemeral nature. If the attempt does not fully succeed, or is deemed to produce a not-interesting-enough result, it is usually lost. Another researcher then repeats the same interaction with the AI tool from scratch. This seems wasteful.

While humans also repeat each other's approaches, unaware of how they might succeed or fail, this repetition tends to build intuition and knowledge. Less knowledge is built through the repetitive use of an AI tool across a community. It is also difficult for humans to record all of their not-interesting-enough results, given the lack of incentive to spend the effort to do so, while it is easy to record AI attempts on problems. Furthermore, with the improved abilities of AI at search, there is less to lose with more systematic, extensive recording of results than there was before, when there was more of a risk of an interesting result being lost in a sea of uninteresting work.

These observations motivate our project, which is an experimental study of the effectiveness of frontier LLMs in operations research. We build a pipeline—a work in progress, but already running end to end—that extracts open problems from the operations research literature, decides whether each one is still open, attempts to solve it with frontier models, and records everything in a public database. The database is live at `https://pranav-nuti.github.io/open-problems-in-or/`, and this working draft reports our progress so far.

Our main finding is that these models, even with an extremely simple pipeline, are capable of making significant contributions. As we detail below, our project has produced complete solutions to a dozen previously open problems, which we are in the process of verifying.

Our current pipeline for building the database has three broad stages, which we describe in detail in Section 2 and summarize in Figure 1. First, we *extract* candidate open problems from recently published papers. We use the last three years of publications from the journal *Mathematics of Operations Research*, but plan to expand to other journals as well. We provide an LLM with a PDF of the paper from the arXiv, and ask it to extract exactly zero or one explicitly stated open problem

The Open Problems in OR database (snapshot, late May 2026)	Count
Open problems extracted from *Math. of OR* (last three years)	132
with an automated forward-citation literature review	105
Problems with at least one recorded solution	12
Problems with recorded partial progress	8
Problems with some recorded progress (solution or partial)	20

Table 1: A snapshot of the project's public database. All counts are work-in-progress and will change as the pipeline runs on more papers and as attempts are verified.

from the paper, choosing the most important and general problem if multiple are present. We then ask the LLM to create a website entry that can be understood by a mathematically educated reader without access to the paper. The LLM is instructed to include useful data, such as subject area, keywords, and potential for attack by counterexamples.

Second, for each candidate open problem, we generate a *literature review* by checking each paper that cites the paper that introduced the problem. For each citing paper, we ask an LLM to identify a key result, and its connection to the open problem. We also ask it to categorize its relevance (provides a solution, or is a related tool, for instance), and provide a relevance score. A different LLM with web-access is then asked to aggregate these citation analyses to provide an overall summary, judging whether the extracted problem is open, and selecting papers that are most relevant to display in the website entry.

Third, we use a frontier LLM to *solve* these problems. We start with an automated pipeline based on a refined version of a solver–verifier loop. Additionally, we sometimes take outputs produced by this pipeline, if any (including partial progress or solutions), and feed them into a 'manual' subscription-based pipeline with engineered prompts to generate solutions. We note that the manual pipeline is manual due to its dependence on using a subscription rather than API access, but is substantially mechanical in nature, and does not require real human expertise. A subscription provides access to better models at a cheaper price than through the API.

From the last three years of *Mathematics of Operations Research*, our pipeline extracted and published 132 open problems, attaching an automated 'forward-citation' based literature review to 105 of them. Across these problems, our combined automated and manual solving workflows have so far produced complete solution write-ups for 12 problems and rigorous partial progress for a further 8; Table 1 provides a summary. A subset of the (human) authors have personally verified the solution to problem 47, on fair multi-resource allocation with Leontief utilities, which we describe in detail in Section 4.

One of the complete solutions—to problem 21—was produced end-to-end by the automated solver–verifier loop, with no manual prompting (the unverified solution is included in Appendix B). In a second case, problem 24, the output of the automated loop was handed to the manual pipeline, which produced a solution (the unverified solution is included in Appendix C).

Human verification of solutions is expensive. In our experience, LLM-based (informal) verification is improving rapidly in quality. Still, it is necessarily imperfect. We have two responses to the issue.

We record yet-to-be-verified results as well as partial progress on our website because the central premise of the project is that AI-assisted attempts should not be ephemeral: recording a plausible, machine-checked attempt—together with the prompts, the literature review, and any partial progress—is potentially useful to the authors of the original paper and to later human or AI attempts.

In the long run, we envision a more automated verification process with auto-formalization of solutions and partial progress.

2 The pipeline

Our pipeline turns a journal and a publication window into a public database of open problems, literature reviews, and machine-generated solution attempts. It consists of the following parts: paper acquisition and open-problem extraction, a 'forward-citation' openness review, an automated solve–verify loop, and a manual workflow built on subscription frontier models, all feeding into a public website. Figure 1 shows how they connect. Every stage is implemented with off-the-shelf frontier LLMs and standard APIs.

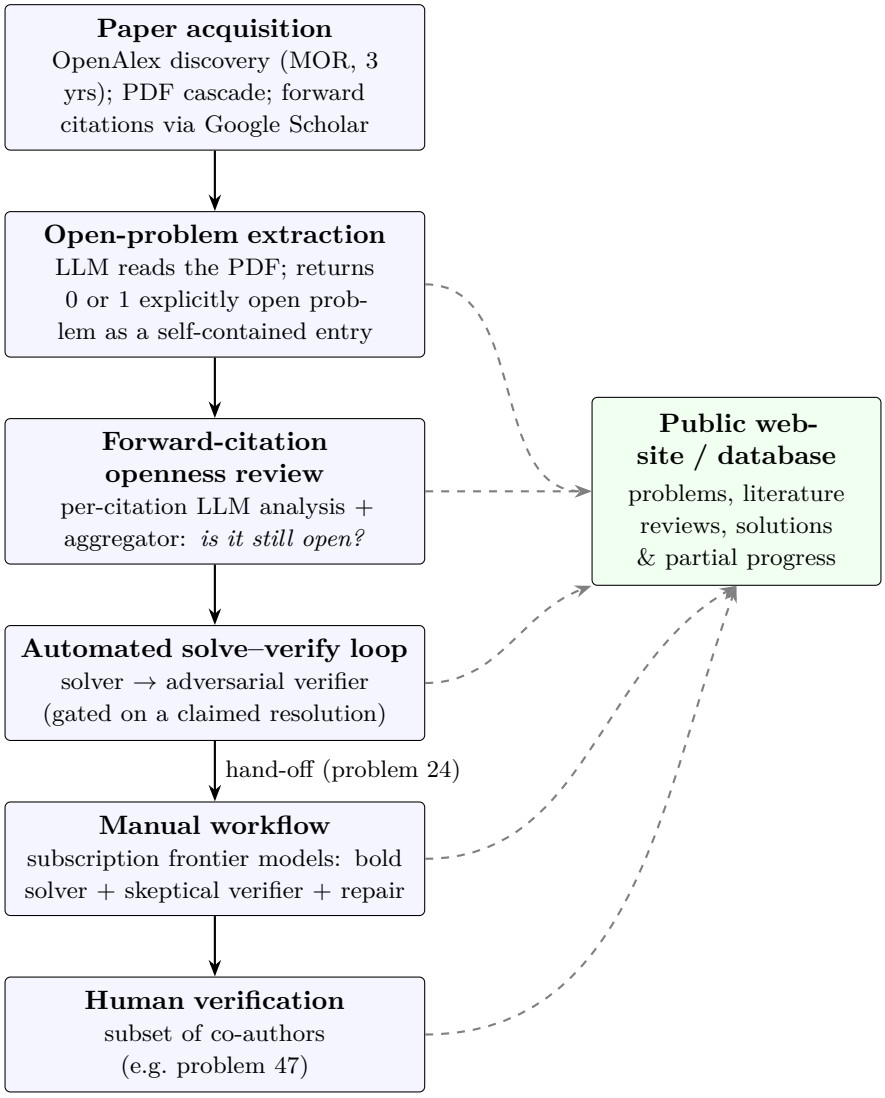

Figure 1: The pipeline.

2.1 Paper acquisition and open-problem extraction

The corpus of papers we consider is defined by a journal and a publication window. We discover these papers using OpenAlex keyed on the journal's ISSN. We then recover PDFs of papers using a cascade of sources, including Google Scholar search links, DOI-based lookups, open-access lookups, title-based lookup, and targeted arXiv search. For each paper, we record bibliographic metadata, DOI or source URL when available, and the candidate PDF used for extraction. In our current runs, the corpus is the last three years of *Mathematics of Operations Research*. A limitation of our method is that we cannot hope to find all the papers published in the journal this way, we sometimes access outdated versions of papers, and we occasionally miss papers whose titles have changed across various versions.

We then extract open questions from the candidate PDFs. The extraction prompt is available in Appendix A. The model is asked to return exactly zero or one explicitly stated unresolved mathematical problem, and instructed not to convert a vague future-work direction into a more precise conjecture. If no eligible problem is explicitly stated, the correct output is the empty list. If several eligible problems are present, the model is asked to choose the one with the highest field-level importance and generality, while staying faithful to the paper's wording.

In experiments, we find that it is necessary to repeatedly remind the model (in the prompt) to stick to explicitly stated open problems, and to not invent questions that are more concrete than what the paper contains. This sometimes leads to narrow technical issues arising from a particular analysis being labeled as open problems, which is arguably unfortunate, and a limitation of our current pipeline.

For each extracted problem, the model produces a self-contained entry. The fields include a title, background, a mathematical problem statement, the location of the open problem in the source paper, an exact quote from the paper witnessing openness, an area label, a top-level category, keywords, and two computational-suitability judgments. These suitability judgments ask whether automated counterexample search, or a code-based evolutionary search loop with a meaningful candidate representation and fitness signal, could plausibly make progress; they are what later let a user sort the database by how amenable a problem looks to machine attack. The entry is written so that a mathematically educated reader can understand it without the source paper, and the mathematical fields avoid any language asserting that the problem is open, keeping that evidence confined to the location and quote fields. Extraction in our current runs uses a frontier reasoning model (`gpt-5.2`).

The second of these two judgments deserves a word of explanation, because it points to a parallel goal of the project. AlphaEvolve is Google DeepMind's evolutionary coding agent, which searches a space of candidate programs against an automatic, computable fitness signal and has been used to discover new algorithms and mathematical constructions. Through a collaboration with Google DeepMind, our team is among its early testers. As we collect and verify open problems in operations research, a side goal is to identify the subset that are natural use-cases for a tool of this kind—problems that, at their core, amount to *searching for a constant* or an extremal object. A canonical example is improving a lower bound on the performance of an algorithm by searching over a suitably parameterized family of hard instances: the family is programmatically specifiable, and the realized performance ratio gives an honest, computable fitness signal. The extraction stage therefore scores every problem for both plain counterexample search and this evolutionary code-search route, and the website (Section 3) lets a visitor rank the whole database by either score.

2.2 Forward-citation openness review

For each extracted problem, our pipeline assembles the papers that cite the source paper—its 'forward citations', gathered from Google Scholar—and asks whether any of them has already resolved the question.

The per-citation analysis is run by an LLM that reads each citing paper (the PDF when available, otherwise the title and abstract, with long PDFs truncated to a fixed input budget). For each citing paper it returns a relevance category—one of SOLUTION, PARTIAL_PROGRESS, RELATED_TOOL, BACKGROUND, RESTATEMENT, or IRRELEVANT—together with a relevance score from 1 to 10, the citing paper's key result, its connection to the open problem, and a short summary. A separate aggregation step then selects the most relevant citing papers, rates the importance and difficulty of the problem, writes a short synthesis of the state of the art, and—most importantly—judges whether the problem still appears to be open. The forward-citation review uses the same frontier model (`gpt-5.2`); 105 of the 132 problems currently carry such a review. Missing reviews are due to the paper having been recently published, and therefore lacking citations.

In testing, our pipeline performs better than off-the-shelf literature review tools available through API access. However, it can miss relevant papers that do not cite the source. We plan to mitigate by adding a more general literature review step to our pipeline, but we believe that finding such relevant papers is fundamentally a hard search problem.

2.3 The automated solve–verify loop

Once a problem has been extracted and reviewed, it can be routed into the automated solving pipeline. The automated solver is a single frontier model call followed, when appropriate, by an adversarial verification call. The solver has web access, and the prompt treats the web as a research aid: it is asked to use search to orient itself in the literature, identify related techniques and named theorems, and cross-check citations.

The solver prompt is organized around a sequence of phases: understand, plan, execute, and look back. In the first phase the model restates the problem in its own notation, records the negation, and resolves possible alternative readings by attempting the strongest one. In the planning phase it proposes several possible attacks, chooses a primary and backup strategy, and, when the problem has natural parameters, starts with a small nontrivial case. In the execution phase it is asked to produce a theorem-and-lemma proof, avoiding unsupported appeals to "standard" arguments. In the final phase it red-teams its own proof by looking for counterexamples to the main theorem and to nontrivial lemmas, checking boundary cases, identifying likely referee objections, and auditing every cited theorem.

The model is instructed to prefer a rigorously proved partial result to a solution with gaps, to state all hypotheses explicitly, and encouraged to use non-elementary methods when appropriate. The prompt also supplies a problem-solving toolkit: the solver is encouraged to try solving modified versions of the problem, to look for analogies with related solved problems, to strip away unnecessary assumptions, and to consider multiple conceptual frames—geometric, combinatorial, probabilistic—before committing to a proof strategy.

The solver must return a single body-level LaTeX block with a verdict of the form `% Verdict: SOLVED`, `% Verdict: DISPROVED`, `% Verdict: PARTIAL`, or `% Verdict: NO-PROGRESS`. Then, required sections force the model to separate the problem restatement, alternative readings, related literature found, proof sketch, main result, proof, any gap to the full statement, self-red-team, invoked results, bibliography, and self-confidence. When the verdict is `PARTIAL` or `NO-PROGRESS`, the solver is required to report the substantive ideas it tried and the obstruction each one encountered.

When the solver claims a complete resolution, with verdict `SOLVED` or `DISPROVED`, the claimed solution is passed to a separate verifier. It receives the original problem statement and the proposed solution and has exactly two possible outputs. If the argument is ready for publication, it rewrites it as a short body-level LaTeX note with verdict `VERIFIED`, including title, abstract, problem statement, main result, proof, and remarks. If it finds any gap, missing hypothesis check, unjustified inference, ambiguous claim, edge-case failure, or mismatch between the theorem proved and the problem posed, it returns no paper and instead emits a flat list of blocking issues with verdict `ISSUES`. We may then provide the solver with this list of issues, and repeat this process, until the verifier returns a verdict of `VERIFIED`.

The automated loop solved problem 21 outright.

2.4 The manual workflow

The manual workflow is in fact entirely mechanical, but runs on subscription frontier models through their chat interfaces, with engineered prompts and custom instructions (all recorded in Appendix A). We find these models are more capable, require less detailed prompting, and are cheaper to use. The first prompt asks the model to solve the problem directly and to ignore the fact that it is labeled open—to "be brave". The accompanying custom instructions encouraged the model to attempt the harder interpretation when there were multiple readings, to avoid excessive computation for a theoretical problem, and to report rigorous partial progress if it could not solve the problem.

A second model instance is then used adversarially, asked to verify the argument carefully and, in particular, to check that it solves the problem actually posed in the source paper. A formatting-and-repair branch either writes up a verified argument as a clean LaTeX paper or, if the verifier finds a gap, tries to fix it and hands the result back to a solver for another pass.

The automated loop produces candidate solutions and partial progress that are already in a standardized form; the manual workflow can pick up any of these artifacts and carry them further. We note that we did not re-run the automated loop on problems the manual workflow had already solved.

3 The website

The public website at `https://pranav-nuti.github.io/open-problems-in-or/` aims to make AI-assisted work on open problems *cumulative* rather than ephemeral. Today, when a researcher asks a frontier model to attack a problem, the interaction—the literature the model considered, the constructions it tried, the partial progress, the dead ends—vanishes when the tab closes, and the next person starts over. Our website is an attempt to capture that work once and make it browsable, searchable, and citable by everyone. The website is live and currently hosts the 132 open problems extracted from *Mathematics of Operations Research*, each rendered as a self-contained, mathematically typeset entry.

What a visitor can do. The landing page (Figure 2) is a filterable, sortable index of every problem. A reader can full-text search across titles, areas, keywords, and source papers; filter by mathematical topic using a curated taxonomy; and filter by progress status—problems with a solution, with partial progress, with either, or with neither. The sort controls allow visitors to rank problems by the two computational-suitability scores assigned at extraction ("counterexample suitability" and code-search, i.e. "AlphaEvolve," suitability), effectively asking the database *which open problems look most amenable to machine attack right now.* Each row shows the problem

number, title, source paper with journal and year, and badges marking the area, and how many solutions or partial-progress write-ups are attached.

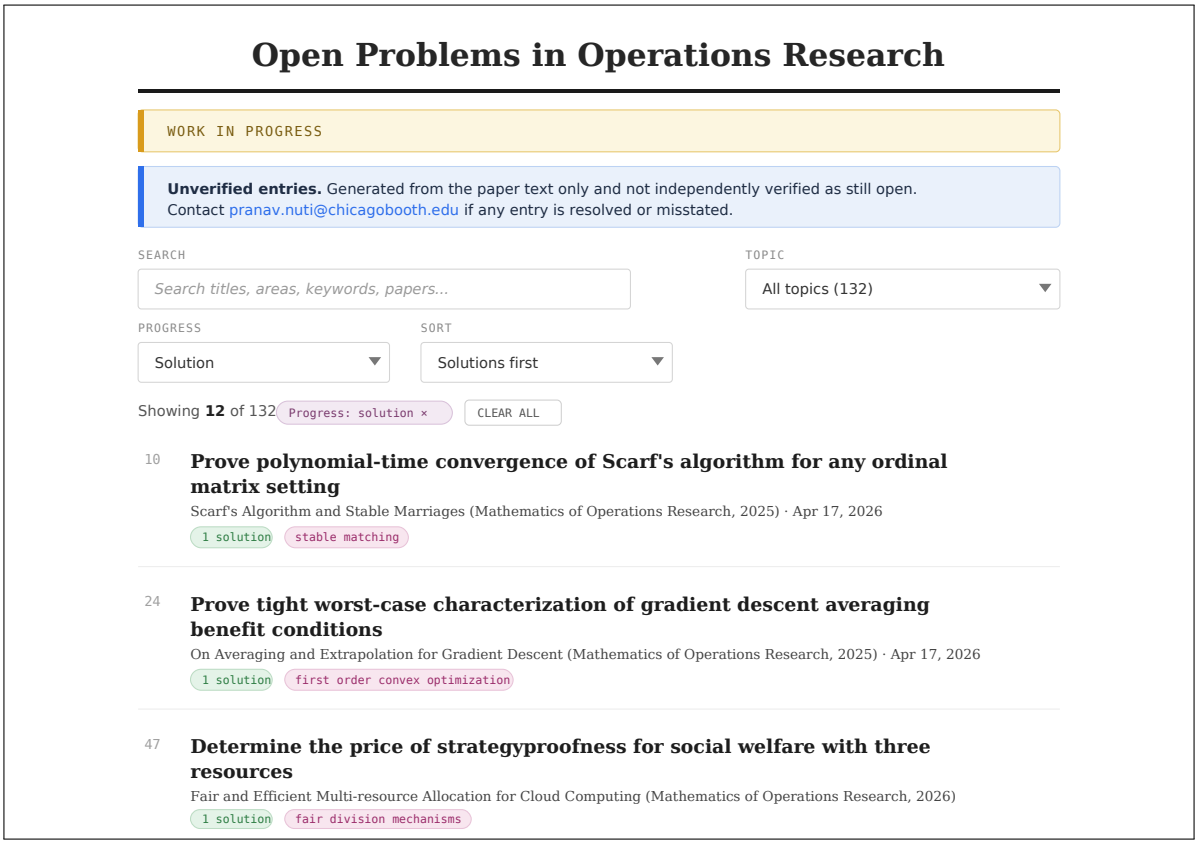

Figure 2: The listing view of https://pranav-nuti.github.io/open-problems-in-or/, filtered to problems with a recorded solution. Each entry shows its number, title, source paper, a progress badge, and an area tag. Problems 24 and 47, discussed in this paper, both appear.

Inside a problem entry. Clicking a problem opens its dedicated page (Figure 3). This is the self-contained entry the extraction and literature review stages produced: a background section that defines all notation, a precise problem statement, the exact quote from the source paper quote that witnesses openness, keywords, and the area and category labels. The metadata records the model that produced the entry, the upload date, and the verification status. A visitor can copy the background and problem statement as LaTeX, or view the raw source, so an entry can be dropped directly into one's own notes or into a prompt. Where a forward-citation review exists, it is shown as a collapsible literature review with the selected citing papers, their relevance, and a synthesis of the state of the art. When the pipeline has produced solution or partial-progress write-ups, they are attached to the entry as downloadable documents.

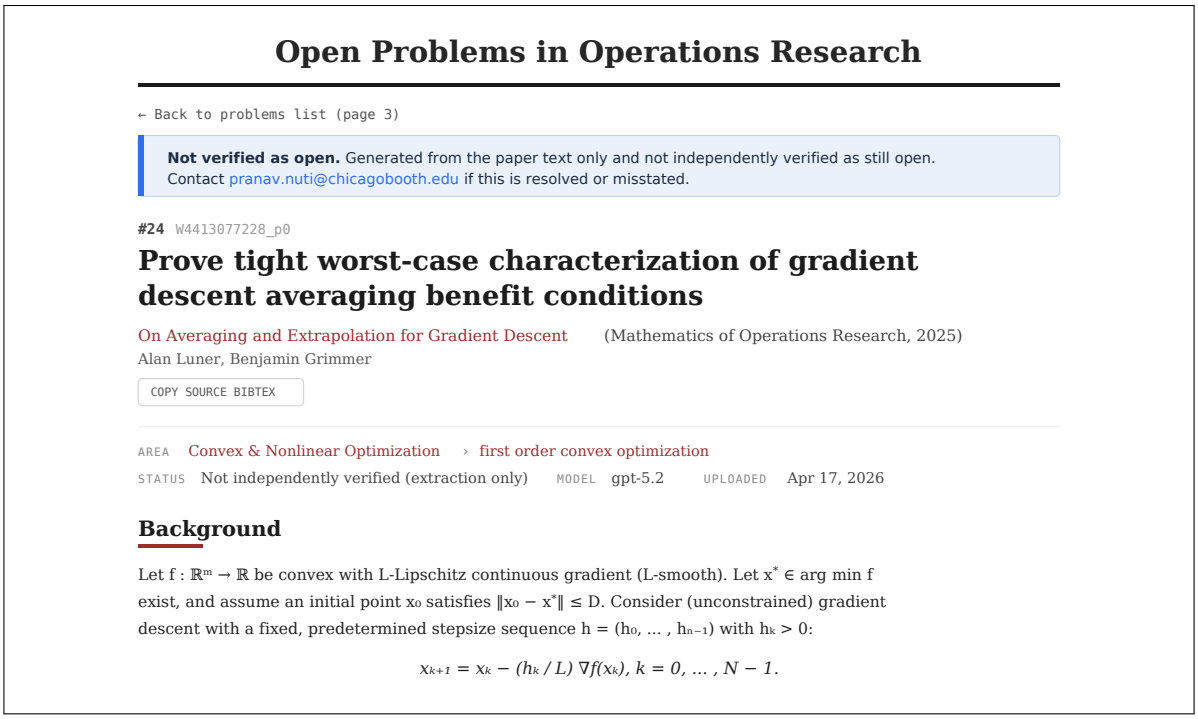

Figure 3: A problem entry on https://pranav-nuti.github.io/open-problems-in-or/ (problem 24). The page is self-contained: background, problem statement, source quote witnessing openness, keywords, metadata, a copy-to-LaTeX control, and—when available—a forward-citation literature review and downloadable solution or partial-progress write-ups.

Where it is going. The website is the front end for our larger program: expanding the corpus to other leading journals, attaching formally verified partial progress, and—most importantly—opening the solving pipeline itself to the community, so that researchers can point it at their own problems with their own prompts and models. Concretely, we plan to democratize the pipeline by hosting it on a server reachable directly from the website: a researcher will be able to enter an API key for their own preferred model, submit a paper of their choosing, and run the full extract–review–solve workflow—with their own prompts and model choices—without installing or maintaining anything, and have the resulting attempts recorded in the shared database. Section 6 reiterates this plan, and describes our broader roadmap.

4 Problem 47: verified case study

The proof below has been edited and verified by hand by a subset of the co-authors specifically for this paper. It is a slightly simplified version of the proof on the website, still generated by an LLM.

Problem 47 is our main case study. It comes from Bei, Li, and Luo's work on fair and efficient multi-resource allocation with Leontief utilities [3]. The source paper introduced a fair-ratio version of the price of strategyproofness for social welfare and left open the exact three-resource value, proving only

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

We prove that the upper bound is tight:

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

The lower bound is stronger than the benchmark definition requires: it applies to every deterministic mechanism satisfying only share incentive and strategyproofness.[1]

The public database entry is displayed as "Determine the price of strategyproofness for social welfare with three resources." The source-paper sentence extracted by the pipeline as evidence of openness was:

> For the case when $m = 3$, we show that the price of SP is still ∞ for utilization, while for SW we can only get a lower bound of 2. We leave the gap between 2 and 3 as an open question.

This entry was a natural target for AI-assisted work because the missing result was plausibly reducible to the discovery of a carefully designed hard instance.

The automated forward-citation review attached to this entry is empty since the paper had not yet accumulated citations in the citation source used by the pipeline. The source paper itself and the earlier dominant-resource-fairness literature remain relevant background [4, 5].

4.1 Model and result

We use the Leontief multi-resource allocation model. There are n agents and three divisible resources, each with unit supply. Agent i has a normalized demand vector $d_i = (d_{i1}, d_{i2}, d_{i3}) \in (0, 1]^3$ with $\max_r d_{ir} = 1$. Given a bundle $X \in \mathbb{R}^3_+$, the Leontief value of X for demand d is

$$v_d(X) = \min_{r \in \{1,2,3\}} \frac{X_r}{d_r}.$$

Thus, under allocation A, agent i receives utility $u_i(A_i) = v_{d_i}(A_i)$.

A feasible allocation satisfies $\sum_i A_{ir} \leq 1$ for every resource r. Share incentive (SI) means that every agent receives utility at least $1/n$. Envy freeness (EF) means that $v_{d_i}(A_i) \geq v_{d_i}(A_j)$ for all agents i, j. Pareto optimality (PO) means that no feasible allocation weakly improves all agents and strictly improves at least one. Strategyproofness (SP) means that, for every true demand d_i, replacing the report of agent i by another normalized demand vector cannot increase the utility evaluated with respect to d_i. All mechanisms in the theorem are deterministic.

The social welfare of an allocation is

$$\mathrm{SW}(A) = \sum_i u_i(A_i).$$

For an instance I, let

$$\mathrm{OPT}_{\mathrm{fair}}(I) = \max\{\mathrm{SW}(A) : A \text{ is feasible, SI and EF.}\}.$$

Note that we may also impose that the allocation satisfies PO, but this makes no difference, since the maximum in question is always attained by a PO allocation. For a mechanism f, define the fair-ratio:

$$\mathrm{FR}_{\mathrm{SW}}(f) = \sup_I \frac{\mathrm{OPT}_{\mathrm{fair}}(I)}{\mathrm{SW}(f(I))}.$$

Define the *price of strategyproofness*:

$$\mathrm{PoSP}_{\mathrm{SW}}(3) = \inf_{f \text{ satisfies SI, EF, PO, SP}} \mathrm{FR}_{\mathrm{SW}}(f).$$

[1] In fact, after the preparation of this draft, manually prompting an LLM in an ad-hoc way also provided an unverified result that applies to every deterministic mechanism satisfying *just* strategyproofness.

Theorem 4.1. *For deterministic mechanisms in the above model,*

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

More strongly, every deterministic mechanism satisfying SI and SP has $\mathrm{FR}_{\mathrm{SW}}(f) = 3$ *when there are three resources.*

Note that [3] already shows that for $m > 3$ resources, $\mathrm{PoSP}_{\mathrm{SW}}(m) = m$, while the exact value of $\mathrm{PoSP}_{\mathrm{SW}}(2)$ remains open.

4.2 Relation to previous lower-bound arguments

The proof belongs to a line of lower-bound constructions for strategyproof multi-resource allocation. Parkes, Procaccia, and Shah showed that, under the classical social-welfare approximation benchmark, the poor $1/m$ welfare behavior of dominant-resource-fairness (DRF) (a mechanism introduced by [4]) is unavoidable for mechanisms satisfying natural combinations of fairness and strategyproofness; their strategyproofness lower bound already uses the core idea of comparing a truthful profile with a profile in which one agent changes her reported demand [5]. Bei, Li, and Luo introduced the fair-ratio benchmark used here, proved matching bounds for several parameter regimes, and used related demand-scaling and hard-instance ideas in their price-of-strategyproofness analysis [3].

What appears to be new in the present proof is the harmonic-weighted multi-ladder selection step. The earlier constructions can force some strategically constrained agent to have low welfare, but the three-resource fair-ratio gap requires the selected agents' total welfare contribution to vanish after taking limits.

4.3 Proof of Theorem 4.1

4.3.1 The easy upper bound

Proposition 1. *For three resources, every SI mechanism has fair-ratio at most* 3*. Consequently, since dominant-resource-fairness mechanisms [4] provide examples satisfying SI, EF, PO, and SP, one has* $\mathrm{PoSP}_{\mathrm{SW}}(3) \le 3$*.*

Proof. Fix any feasible allocation A and, for every agent i, choose a dominant resource $r(i)$ with $d_{i,r(i)} = 1$. Then

$$u_i(A_i) = v_{d_i}(A_i) \le A_{i,r(i)}.$$

Summing by dominant resource,

$$\mathrm{SW}(A) \le \sum_{r=1}^{3} \sum_{i:r(i)=r} A_{ir} \le 3.$$

On the other hand, if f satisfies SI, then $\mathrm{SW}(f(I)) \ge n \cdot (1/n) = 1$ on every instance I. Hence $\mathrm{OPT}_{\mathrm{fair}}(I)/\mathrm{SW}(f(I)) \le 3$ for every I. □

It remains to prove that no SI and SP mechanism can do better.

4.3.2 A hard family of instances

Fix an integer $q \geq 3$. Later we take $q \to \infty$. Choose n sufficiently large as a function of q, say $n > 2q^4$. For $t = 1, \ldots, q$, set

$$\alpha_t = n^{t-q-2}, \qquad s_t = \frac{q-1}{tn}.$$

There are three groups of agents. The first group P contains $n - 2q$ agents, each with demand

$$a = \left(1, \frac{1}{n}, \frac{1}{n}\right).$$

The second group contains agents $B_1, \ldots, B_q$, with original demands

$$b_t = \left(\frac{1}{q}, 1, \alpha_t\right), \qquad t = 1, \ldots, q.$$

The third group contains agents $C_1, \ldots, C_q$, with original demands

$$c_t = \left(\frac{1}{q}, \alpha_t, 1\right), \qquad t = 1, \ldots, q.$$

We shall also use the lowered demands

$$\widehat{b}_t = (s_t, 1, \alpha_t), \qquad \widehat{c}_t = (s_t, \alpha_t, 1).$$

Fix an arbitrary deterministic mechanism f satisfying SI and SP. We shall construct a profile on which the fair benchmark has social welfare tending to 3, while f has social welfare tending to 1.

This profile will mostly consist of the agents in groups P, B and C, but with one agent in group B and one agents in group C replaced by agents with lowered demands.

With just the groups P, B, and C, no allocation can obtain a utility much better than 1 due to the demand for resource 1. However, if a B agent and a C agent are each replaced by agents with lowered demands for resource 1, these agents can be allocated more of the second or third resources, giving them utility almost 1, leading to a total utility of 3.

As we will show in Lemma 2, no mechanism has the capacity to deal with all such possible modifications of pairs of agents from groups B and C and simultaneously obtain a utility much greater than 1.

4.3.3 A weighted averaging lemma

Let

$$H_q = \sum_{t=1}^{q} \frac{1}{t}, \qquad S = \sum_{t=1}^{q} s_t = \frac{q-1}{n} H_q.$$

Define a probability distribution μ on $\{1, \ldots, q\}$ by

$$\mu_t = \frac{s_t}{S}.$$

For a fixed k, let $I^{\widehat{c_k}}$ be the profile in which only C_k's demand is changed from c_k to $\widehat{c}_k$; all B-agents have their original demands. Let $R^B_{j,k}$ be the amount of resource 1 allocated by f to agent B_j on profile $I^{\widehat{c_k}}$.

Similarly, for a fixed j, let $I^{\widehat{b_j}}$ be the profile in which only B_j's demand is changed from b_j to $\widehat{b}_j$; all C-agents have their original demands. Let $R^C_{j,k}$ be the amount of resource 1 allocated by f to agent C_k on profile $I^{\widehat{b_j}}$.

Lemma 1. *There exists a pair $(j,k) \in \{1,\dots,q\}^2$ such that*

$$\frac{R^B_{j,k}}{s_j} + \frac{R^C_{j,k}}{s_k} \le \frac{4q}{(q-1)H_q}.$$

Proof. Fix k. On profile $I^{\widehat{c}_k}$, SI gives each of the agents in group P utility at least $1/n$. Since their first demand coordinate is 1, these agents consume at least

$$\frac{n-2q}{n} = 1 - \frac{2q}{n}$$

units of resource 1. Therefore all agents outside P together consume at most $2q/n$ units of resource 1. In particular,

$$\sum_{j=1}^{q} \mu_j \frac{R^B_{j,k}}{s_j} = \frac{1}{S}\sum_{j=1}^{q} R^B_{j,k} \le \frac{2q}{nS} = \frac{2q}{(q-1)H_q}.$$

Averaging also over $k \sim \mu$, we obtain

$$\mathbb{E}_{j,k}\left[\frac{R^B_{j,k}}{s_j}\right] \le \frac{2q}{(q-1)H_q}.$$

The same argument, with the roles of B and C exchanged, gives

$$\mathbb{E}_{j,k}\left[\frac{R^C_{j,k}}{s_k}\right] \le \frac{2q}{(q-1)H_q}.$$

Therefore

$$\mathbb{E}_{j,k}\left[\frac{R^B_{j,k}}{s_j} + \frac{R^C_{j,k}}{s_k}\right] \le \frac{4q}{(q-1)H_q}.$$

Hence at least one pair (j,k) satisfies the desired inequality. □

Fix such a pair (j,k). Let $I^{j,k}$ denote the final profile obtained from the original profile by replacing b_j by $\widehat{b}_j$, and replacing c_k by $\widehat{c}_k$, leaving all other demands unchanged.

4.3.4 Strategyproofness makes the chosen agents low-utility

Lemma 2. *On the final profile $I^{j,k}$,*

$$SW(f(I^{j,k})) \le 1 + \frac{2q^2}{n} + \frac{4q}{(q-1)H_q}.$$

Proof. Let u_B be the utility that agent B_j receives on the final profile $I^{j,k}$, evaluated with respect to its final demand $\widehat{b}_j$. Compare $I^{j,k}$ with the profile $I^{\widehat{c}_k}$. On $I^{\widehat{c}_k}$, agent B_j still has original demand

$$b_j = \left(\frac{1}{q}, 1, \alpha_j\right).$$

By definition, f gives B_j exactly $R^B_{j,k}$ units of resource 1 on this profile. Since the first coordinate of b_j is $1/q$, its truthful utility on $I^{\widehat{c}_k}$ is at most

$$qR^B_{j,k}.$$

Now suppose that, on profile $I^{\widehat{c_k}}$, agent B_j misreports $\widehat{b}_j$. The reported profile is then exactly $I^{j,k}$. Since f is deterministic, B_j receives exactly the same bundle it receives on $I^{j,k}$.

That bundle has value u_B under

$$\widehat{b}_j = (s_j, 1, \alpha_j).$$

Therefore it contains at least $s_j u_B$ units of resource 1, at least u_B units of resource 2, and at least $\alpha_j u_B$ units of resource 3. Evaluated under the original demand

$$b_j = \left(\frac{1}{q}, 1, \alpha_j\right),$$

the same bundle therefore has value at least

$$\min\{qs_j u_B,\ u_B,\ u_B\} = qs_j u_B,$$

where we use $qs_j \le 1$, which holds for n large.

Strategyproofness on profile $I^{\widehat{c_k}}$ implies that this misreport cannot improve B_j's utility with respect to its true demand b_j. Hence

$$qs_j u_B \le qR^B_{j,k},$$

and therefore

$$u_B \le \frac{R^B_{j,k}}{s_j}.$$

The same argument, exchanging the roles of B and C, gives the following bound for the final utility u_C of C_k:

$$u_C \le \frac{R^C_{j,k}}{s_k}.$$

By the choice of (j, k),

$$u_B + u_C \le \frac{R^B_{j,k}}{s_j} + \frac{R^C_{j,k}}{s_k} \le \frac{4q}{(q-1)H_q}.$$

It remains to bound the welfare of all other agents. The agents in P have total utility at most 1, because each such agent has first demand coordinate 1, and the total supply of resource 1 is 1.

On the final profile $I^{j,k}$, SI again forces the $n - 2q$ agents in P to consume at least $(n - 2q)/n$ units of resource 1. Thus all agents outside P together receive at most $2q/n$ units of resource 1. Every unchanged non-P agent has first demand coordinate $1/q$, so the total utility of all unchanged non-P agents is at most

$$q \cdot \frac{2q}{n} = \frac{2q^2}{n}.$$

Adding the two changed agents gives

$$SW(f(I^{j,k})) \le 1 + \frac{2q^2}{n} + \frac{4q}{(q-1)H_q}.$$

□

4.3.5 A nearly welfare-three fair allocation for the same profile

We now show that the final profile $I^{j,k}$ admits an SI and EF allocation whose social welfare tends to 3. Let $\varepsilon = \frac{4q^2}{n}$, and define

$$x_B = (1-\varepsilon)s_j, \qquad x_C = (1-\varepsilon)s_k.$$

We define an allocation A^* as follows.

- Every agent in group P receives the bundle $(1/n)a$, and hence utility $1/n$.
- For $h < j$, agent B_h receives the bundle $(qx_B)b_h$, and hence utility qx_B.
- Agent B_j, whose final demand is $\widehat{b}_j$, receives the bundle $(1-\varepsilon)\widehat{b}_j$, and hence utility $1-\varepsilon$.
- For $h > j$, agent B_h receives the bundle $(1/n)b_h$, and hence utility $1/n$.
- For $\ell < k$, agent C_ℓ receives the bundle $(qx_C)c_\ell$, and hence utility qx_C.
- Agent C_k, whose final demand is $\widehat{c}_k$, receives the bundle $(1-\varepsilon)\widehat{c}_k$, and hence utility $1-\varepsilon$.
- For $\ell > k$, agent C_ℓ receives the bundle $(1/n)c_\ell$, and hence utility $1/n$.

The total utility of all the agents is at least the utility of B_j, C_k and the agents in group P. Therefore

$$\begin{aligned} SW(A^*) &\geq 2(1-\varepsilon) + \frac{n-2q}{n} \\ &\geq 3 - \frac{8q^2+2q}{n}. \end{aligned}$$

Lemma 3. *For each fixed q, if n is sufficiently large, then A^* is feasible.*

Proof. We check the three resources. Call an agent B_h is selected if $h \leq j$, and similarly for agents of type C. For resource 1, the agents in P use

$$\frac{n-2q}{n}$$

units. The selected B-agents use exactly jx_B units, and

$$jx_B \leq js_j = \frac{q-1}{n}.$$

The selected C-agents similarly use at most $(q-1)/n$. The unselected B-agents use at most $1/n$ units of resource 1, and the unselected C-agents use at most $1/n$. Hence total resource-1 usage is at most

$$\frac{n-2q}{n} + \frac{q-1}{n} + \frac{q-1}{n} + \frac{1}{n} + \frac{1}{n} = 1.$$

For resource 2, the selected B-agents use at most

$$(j-1)qx_B + (1-\varepsilon) = (1-\varepsilon)\big(1+q(j-1)s_j\big) \leq (1-\varepsilon)\big(1+\frac{q^2}{n}\big) \leq 1 - \frac{3q^2}{n}.$$

The unselected B-agents use at most q/n. The agents in P use at most $1/n$, because each receives resource 2 amount $1/n^2$.

It remains to bound the resource-2 usage of the C-agents. Since $\alpha_\ell \le \alpha_k \le n^{-2}$ for $\ell < k$, and since $qx_C k \le q^2/n$, the selected C-agents use at most

$$qx_C \sum_{\ell<k} \alpha_\ell + (1-\varepsilon)\alpha_k \le \left(1 + \frac{q^2}{n}\right) n^{-2}.$$

The unselected C-agents use at most qn^{-3}. Therefore total resource-2 usage is at most

$$1 - \frac{3q^2}{n} + \frac{q}{n} + \frac{1}{n} + \left(1 + \frac{q^2}{n}\right) n^{-2} + qn^{-3},$$

which is less than 1 for each fixed q and all sufficiently large n.

The resource-3 calculation is identical, with the roles of B and C exchanged. Thus A^* is feasible. □

Lemma 4. *For each fixed q, if n is sufficiently large, then A^* is SI and EF.*

Proof. Share incentive is immediate for every agent receiving a baseline bundle, since these agents receive utility $1/n$. The changed selected agents B_j and C_k receive utility $1-\varepsilon$, which is at least $1/n$ for n large.

For selected original B-agents, the utility is qx_B. Since $j \le q$,

$$qx_B = q(1-\varepsilon)\frac{q-1}{jn} \ge (1-\varepsilon)\frac{q-1}{n}.$$

For $q \ge 3$ and n large, this is at least $1/n$. The same argument applies to selected original C-agents. Hence A^* satisfies SI.

We now prove envy freeness. First observe that no agent with utility at least $1/n$ envies a baseline bundle of the form $(1/n)d'$, where d' is a normalized demand vector. Indeed, if d is the evaluating demand and r is a dominant resource for d, then $d_r = 1$ and $d'_r \le 1$, so

$$v_d((1/n)d') \le \frac{d'_r}{nd_r} \le \frac{1}{n}.$$

Thus it remains only to rule out envy toward selected B- and selected C-bundles.

Among the selected B-agents, all selected bundles contain exactly x_B units of resource 1. For each selected B-agent, the ratio of this common first-resource amount to her own first demand coordinate is exactly her own utility: it is qx_B for the original selected agents B_h, $h < j$, and it is $x_B/s_j = 1-\varepsilon$ for the changed agent B_j. Therefore no selected B-agent envies another selected B-bundle. The same argument applies among the selected C-agents.

Next consider cross-envy between selected B- and selected C-agents. Every selected C-bundle contains at most $\alpha_k \le n^{-2}$ units of resource 2, because $qx_C \le 1$ for n large. Every selected B-agent has second demand coordinate 1 and own utility at least $1/n$, so no selected B-agent envies a selected C-bundle. Symmetrically, no selected C-agent envies a selected B-bundle.

It remains to check agents whose own bundle is a baseline bundle. A group-P agent values any selected B-bundle at most through resource 3. Every selected B-bundle contains at most $\alpha_j \le n^{-2}$ units of resource 3, while a group-P agent has third demand coordinate $1/n$. Hence its value for any selected B-bundle is at most

$$n\alpha_j \le \frac{1}{n}.$$

The same argument, using resource 2, shows that group-P agents do not envy selected C-bundles.

Now take an unselected B_h, so $h > j$. Since $\alpha_h \geq n\alpha_j$, and since every selected B-bundle contains at most α_j units of resource 3, agent B_h's value for any selected B-bundle is at most

$$\frac{\alpha_j}{\alpha_h} \leq \frac{1}{n}.$$

Also, every selected C-bundle contains at most $\alpha_k \leq n^{-2} < 1/n$ units of resource 2, while B_h's second demand coordinate is 1. Hence B_h does not envy selected C-bundles either.

The argument for unselected C-agents is symmetric. Therefore A^* is envy-free. □

Combining the preceding two lemmas, the final profile $I^{j,k}$ admits a feasible SI and EF allocation with welfare at least

$$3 - \frac{8q^2 + 2q}{n}.$$

Thus

$$\mathrm{OPT}_{\mathrm{fair}}(I^{j,k}) \geq 3 - \frac{8q^2 + 2q}{n}.$$

Together with the mechanism-welfare bound, this gives

$$\mathrm{FR}^{\mathrm{SW}}(f) \geq \frac{3 - (8q^2 + 2q)/n}{1 + 2q^2/n + 4q/((q-1)H_q)}.$$

First let $n \to \infty$ for fixed q, and then let $q \to \infty$. Since $H_q \to \infty$, the right-hand side tends to 3. Therefore every deterministic SI and SP mechanism satisfies

$$\mathrm{FR}^{\mathrm{SW}}(f) \geq 3.$$

5 Conclusions

Our draft makes the case for a public database for AI-assisted work on open problems in operations research. Our pipeline turns open questions in a paper into website entries with attached literature review and stores both solution attempts and partial progress, and has the capacity to do this at scale. Storing AI generated work publicly reduces duplicated effort across the community.

Our main message about our pipeline is that we have strong preliminary evidence that it works, and with surprisingly modest machinery. Our pipeline, built entirely from off-the-shelf frontier models, a solve–verify loop, and standard scholarly APIs produces solution attempts for twelve open problems drawn from a leading operations research journal, with partial progress on eight more. While we have not verified all of these attempts, the solution we have in fact verified gives us belief in the viability of our approach.

Our experiments indicate that the place where AI models hold the clearest advantage over humans is in example and counterexample discovery. Problem 47, and other problems we have mentioned in this draft (problems 21 and 24) are good examples. We expect the database to be especially productive for counterexamples humans tend to miss. On the other hand, the pipeline is not *only* a counterexample machine. It also seems effective at positive results (for instance, see problems 52 and 73). We plan to talk about these results more once we have verified them.

6 Future work

This is a work in progress, and we plan to broaden our efforts in several ways.

First, we plan to use LLMs to extract open problems from many more leading journals, and organize them into our public database, making problems easy to find, cite, and share across the field. Our design is journal-agnostic, so expanding is just a matter of running the pipeline.

We also plan to strengthen our solving pipeline. The simple solve–verify loop used for the results in this paper is only the most basic member of a family. We have in fact already implemented a more sophisticated pipeline, and are currently running tests and tuning its components—how strategies are proposed and pruned, how solver and verifier interact, how compute is allocated across attempts.

Another architectural direction is managing the memory of various solvers better. After we attach a persistent *memory file* to every problem, recording the partial progress, promising leads, and dead ends accumulated across runs, we plan for a dedicated memory-management agent to curate this record and decide what to surface to each solver agent on a given run—which lemmas, failed strategies, or constructions are worth carrying forward—so that the relevant context is supplied without overwhelming the solver.

Formal verification is the natural endpoint of our adversarial-verification based approach. The difficulty, which we do not minimize, is that operations research papers usually require substantial formalization—the model, the fairness and incentive constraints, the asymptotic quantifiers, the background inequalities—before a proof can even begin.

Finally, the pipeline itself will be available on the website with an interface where researchers can try solving their own problems, as described in Section 3. This gives operations researchers an accessible, practical way to use AI-for-math tools.

We seek to build shared infrastructure and shared data—problems, literature reviews, counterexamples, and formally verified partial results—that reduces duplicated effort and can meaningfully speed up automated mathematical research in our field.

References

[1] Tony Feng, Junehyuk Jung, Sang-hyun Kim, Carlo Pagano, Sergei Gukov, et al. Aletheia tackles FirstProof autonomously. arXiv:2602.21201, 2026.

[2] Remarks on the disproof of the unit distance conjecture. arXiv:2605.20695, 2026.

[3] Xiaohui Bei, Zihao Li, and Junjie Luo. Fair and efficient multi-resource allocation for cloud computing: Beyond dominant resource fairness. *Mathematics of Operations Research*, 2026. DOI: 10.1287/moor.2024.0714.

[4] Ali Ghodsi, Matei Zaharia, Benjamin Hindman, Andy Konwinski, Scott Shenker, and Ion Stoica. Dominant resource fairness: Fair allocation of multiple resource types. In *Proceedings of the 8th USENIX Symposium on Networked Systems Design and Implementation*, 2011.

[5] David C. Parkes, Ariel D. Procaccia, and Nisarg Shah. Beyond dominant resource fairness: Extensions, limitations, and indivisibilities. *ACM Transactions on Economics and Computation*, 3(1), Article 3, 2015. DOI: 10.1145/2739040.

[6] Open Problems in OR. Determine the price of strategyproofness for social welfare with three resources. Problem entry `W7128591806_p0`. https://pranav-nuti.github.io/open-problems-in-or/problem.html?id=W7128591806_p0.

A Prompts

This appendix records the natural-language prompts used by the extraction, citation-review, automated solver–verifier, and manual solving workflows. Placeholders in braces denote fields filled by the surrounding pipeline. The token `__CATEGORY_LIST__` in the extraction prompt is replaced at runtime by the current website taxonomy.

A.1 Open-problem extraction

System message.

```
You are a professor in operations research doing theoretical research. Your task is to identify
    the single most important explicitly stated mathematical open problem that is not solved in
    this paper. The extracted output will be published on a public open-problems website and read
     in isolation. Write for a mathematically educated graduate-level reader who may not know
    this specific subfield and may not have access to the source paper. Every field must be self-
    contained and understandable without paper-internal references such as 'Assumption 1', '
    Theorem 5', 'Equation (3)', or 'see Section 4'.

Rules:

- Only consider mathematical problems that are EXPLICITLY stated as open or unsolved in the paper.
     Treat explicitly stated conjectures, explicitly stated unsolved problems, and explicitly
    stated open questions as eligible. Make sure the paper itself does not solve the stated open
    problem. If the paper resolves a stated open problem, do not extract it. Do not treat generic
     future-work directions as open problems. Do not infer or speculate about unstated problems.

- If the paper contains no explicitly stated open problems, return an empty list. Return an empty
     list also when the paper itself resolves the stated open problem. Return an empty list ONLY
    when there is no remaining explicitly stated unsolved problem in the paper. Return exactly
    zero or one problem.

- If multiple open problems are explicitly stated, select the one with the highest field-level
    importance and generality, while remaining explicitly stated in the paper. Prioritize
    problems framed as central, fundamental, broad, or likely to influence multiple follow-on
    works; and de-prioritize narrow technical issues that arise from the authors' analysis. Among
     explicit candidates, prefer less vague statements when possible. Use prominence cues when
    available: emphasis in title/abstract/introduction/conclusion, or dedicated 'open problems'
    discussion.

- Faithfulness rule: do not make the extracted problem more concrete than the paper's own
    statement. Do not invent missing objects, assumptions, objectives, parameter ranges, or
    constraints. It is acceptable for the extracted problem to remain somewhat vague if the paper
     is vague; but never add concrete details not present in the paper.

- In title, background_latex, and problem_definition_latex, describe only the mathematical
    content of the problem. Do NOT mention that it is open/unsolved, and do NOT include phrases
    like 'open problem', 'it remains open', or 'future work'. Keep openness evidence only in
    location_in_paper and open_problem_quote.

For the selected problem:

1. Give a concise, descriptive title that is self-contained and meaningful without reading the
    source paper.
   - Title must be plain text only (no LaTeX, no math delimiters, no TeX commands).
   - Title length must be 8-18 words.
   - Title must follow this structure: task + object + setting.
```

```
   - Avoid paper-internal references such as 'Conjecture 3', 'Theorem 5', 'Assumption 1', or
     section/equation numbers.
   - Avoid vague placeholders such as 'Algorithm Class 1' or generic phrasing like 'An open
     question in optimization'.
   - Example good title: 'Tight approximation ratio for anonymous-reserve auctions with regular
     values'.
   - Example bad title: 'Open problem from Theorem 5 under Assumption 1'.
2. Write two separate LaTeX fields:
   - background_latex: setup, definitions, assumptions, and notation only.
   - background_latex may be long; prioritize completeness and clarity over brevity.
   - Assume the reader knows standard math basics (e.g., random variables, expectation, linear
     algebra) but not domain-specific concepts from this paper's area. Explicitly define all
     domain-specific terms before use (for example in prophet inequalities: policy, stopping rule,
      and benchmark such as prophet value/competitive ratio). Define all notation and operators
     used later, including norms, objective functions, feasibility constraints, and any
     assumptions (e.g., independence, continuity, bounded support, moment assumptions).
   - problem_definition_latex: the exact mathematical problem to solve.
   - Keep problem_definition_latex tightly anchored to the quoted open statement; paraphrase for
     clarity but do not add new mathematical commitments.
   - Every symbol that appears in problem_definition_latex must be defined in background_latex
     first. Do not introduce new notation in problem_definition_latex; introduce it in
     background_latex first.
   - Never rely on undefined references such as 'as in Section 3' or 'using notation above'.
   - Both fields must be non-empty.
3. State where this problem is explicitly stated in the original paper (page number and section
     number).
4. Provide the exact 1-3 sentences from the paper that mentions this problem as open.
   - Do not wrap the quote in outer quotation marks.
   - If the quoted sentence contains math notation, normalize math fragments into valid LaTeX
     with inline delimiters \(...\) while preserving the sentence meaning.
   - If the quoted text contains obvious PDF ligature artifacts (e.g., missing letters where "fi
     ", "fl", "ff", "ffi", "ffl" ligatures were not decoded -- appearing as missing characters,
     stray backslashes, or garbled bytes like \xed), silently correct them to standard ASCII text
     (e.g., "de\ntion" ? "definition", "pro\le" ? "profile").

Formatting rules for website compatibility:

- Use plain prose plus math only. For inline math use \(...\). For display math use \[...\].
- Lists are allowed when they improve clarity. For website compatibility, use numbered plain-text
     items (for example: '1) ...', '2) ...'); avoid hyphen-led bullet style.
- You may use simple math/text commands such as \frac, \sum, \prod, \log, \cdot, \text, \mathrm.
- Do NOT use environment wrappers such as \begin{problem}, \begin{itemize}, \begin{enumerate}, \
    begin{theorem}, \begin{align}, or any custom environment.
- Do NOT use document-level commands (no \documentclass, \begin{document}, etc.) and do NOT
    define custom macros (no \newcommand, \def).
Produce LaTeX that can be rendered directly on a web page without preprocessing.
- Do not emit any LaTeX environment delimiters; the output must not contain \begin{...} or \end
    {...} anywhere.

5. Provide one area label for the problem:
   - area: a short label (typically 1-4 words, lowercase, no punctuation) naming the problem area.

   - Priority: first check whether the problem clearly fits one of the labels below. If yes, use
     that string exactly (verbatim). If none of them are a clear fit, generate a custom label. Do
     NOT force a fit to the preferred list -- the list is not exhaustive, and custom labels are
     expected whenever the match is not clear-cut.
   - Labels (use only when it is a high confidence match):
     - online matching
     - prophet inequalities
```

```
      - multi-armed bandits
   - Custom Label Guidelines:
      - Length: 1-4 words, typically 2-3.
      - Case: all lowercase, no punctuation, no hyphens unless standard in the field (e.g., "
    multi-armed").
      - Specificity: name the mathematical problem area, not the technique, application domain,
    or result. For example, prefer "secretary problem" over "optimal stopping algorithm" or "
    hiring decisions"; prefer "stochastic knapsack" over "dynamic programming" or "resource
    allocation".
      - Descriptiveness: the label should be recognizable to an OR theorist as a named subfield
    or problem class (e.g., "online bin packing", "assortment optimization", "contextual bandits
    ", "network revenue management", "pandora's box").
   - Every problem must receive a meaningful area label; do not use placeholders such as "none"
    or "other".
   - category: in addition to the fine-grained area, assign exactly one top-level category from
    this fixed list (copy the chosen string verbatim, including capitalization and ampersands).
    This drives the website's topic browser, so choose the single best fit:
__CATEGORY_LIST__

6. Provide 3-8 keywords capturing the problem's core topics.
7. Assess computational discovery suitability in two separate fields:
   - counterexample_suitability: choose exactly one of high, medium, low.
   - counterexample_suitability_rationale: 1-3 sentences focused on whether a large automated
    instance search is feasible.
   - alphaevolve_suitability: choose exactly one of high, medium, low.
   - alphaevolve_suitability_rationale: 1-3 sentences focused on whether an evolutionary coding-
    agent loop can systematically search a space of candidate constructions/counterexamples/
    algorithms against a computable fitness signal.
   - Be reasonably conservative but do not penalize a problem merely because the final resolution
     requires a proof: AlphaEvolve is a code-search loop, not a theorem prover, and "high" can
    apply when it could plausibly produce explicit constructions, counterexamples, or improved
    numerical bounds that meaningfully advance the problem.
   - For counterexample_suitability, high requires a concrete finite search space or efficiently
    testable constraints/objective across many generated instances.
   - For alphaevolve_suitability, high requires BOTH (a) a programmatically specifiable candidate
     representation (construction, counterexample, policy, algorithm, parameters) and (b) an
    automatic, reasonably efficient fitness signal that is a faithful proxy for the mathematical
    quantity the problem asks about (e.g., computing the realized competitive ratio /
    approximation gap / attained value on a representative family of instances). Iterative
    compute-time feedback is assumed available and does not need to be separately justified.
   - Use medium when a fitness signal exists but is expensive, noisy, or only a partial proxy; or
     when the candidate space is only loosely specifiable.
   - Use low when there is no plausible computable fitness signal, or the problem is purely about
     proving a universal statement with no search-able space of candidate objects whose scores
    would advance it.
```

A.2 Forward-citation analysis

System message.

```
You are an expert mathematics researcher. You will be given:
1. An open problem statement from a research paper
2. A forward-citing paper (a paper that cites the original)

Your task is to analyze how the citing paper relates to the open problem.

Determine:
- Does this paper SOLVE the open problem in its full stated generality? (not just a special case)
```

```
- What is the relevance category?
  SOLUTION: Completely solves the problem as stated
  PARTIAL_PROGRESS: Makes partial progress (solves special cases, proves weaker results,
    establishes bounds)
  RELATED_TOOL: Introduces a tool or technique relevant to the problem
  BACKGROUND: Provides context about why the problem matters
  RESTATEMENT: Restates or reformulates the problem
  IRRELEVANT: Not meaningfully related to the problem
- Relevance score from 1-10 (10 = most relevant)
- The key result from the paper (state the main theorem/result precisely)
- How the paper connects to the open problem
- A brief summary of the paper's contribution

Be precise and honest. If the paper is not relevant, say so. If you cannot determine the paper's
    content from the provided material, say so and rate relevance as 1.

Consistency: if relevance_category is SOLUTION then you MUST set solves_problem to true (SOLUTION
     means the cited paper fully resolves the open problem as stated). If the paper only
    partially resolves it, use PARTIAL_PROGRESS and solves_problem false.

The response tool schema enforces maximum lengths on every text field. Do not attempt full proofs
     or exhaustive exposition: one crisp theorem statement or result in key_result, short
    connection, short summary.

Formatting rules for key_result and connection_to_problem fields:
- Use plain prose plus math only.
- For inline math use \(...\). For display math use \[...\].
- For lists, use plain text bullets beginning with '- '.
- Do NOT use LaTeX environment wrappers.
```

A.3 Forward-citation aggregation

System message.

```
You are an expert mathematics researcher compiling a literature review.

You will be given:
1. An open problem statement
2. Summaries of forward-citing papers and their relevance to the problem
3. The full text of the original source paper (attached as a PDF file). Use it to ground the
    synthesis in the context, notation, and motivation of the original problem statement.

Your task is to:
1. Select the 5-10 most relevant papers (prioritize SOLUTION and PARTIAL_PROGRESS over BACKGROUND
    )
2. For each selected paper, provide a complete BibTeX entry and source URL
3. Determine whether the problem is still open or has been solved
   - If solved, you MUST identify the solving paper via solved_by_title (the
     paper's title as it appears in the citation analysis). The title must
     exactly match the title of one of the papers you put in top_citations.
     Do not use any internal paper id or DOI slug here, just the human title.
4. Rate the importance (1-10) and difficulty (1-10) of the problem
5. Write a synthesis paragraph summarizing the state of the art

For each paper you select for top_citations, reuse the exact relevance_category and
    relevance_score from that paper's forward-citation analysis block; do not change them.
```

```
For BibTeX entries: construct them from the paper metadata provided. Include author, title, year,
     and doi when available. For arXiv papers, include the eprint field.

For the synthesis: write 2-4 paragraphs covering the main advances, remaining gaps, and promising
     directions. If is_open is false, explicitly name the solving paper by its human title (
    exactly as given in the citation analysis) in the synthesis. Never cite the paper by an
    internal paper_id or DOI slug in the synthesis.

Formatting rules for key_result, connection_to_problem, and synthesis:
- Use plain prose plus math only.
- For inline math use \(...\). For display math use \[...\].
- For lists, use plain text bullets beginning with '- '.
- Do NOT use LaTeX environment wrappers.
```

A.4 Automated solver

System message.

```
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.

## Phase-based workflow

Structure your reasoning internally along these four phases, modeled on Polya's *Understand /
    Plan / Execute / Look back*.

1. **Understand.** Quote the problem in your own notation. List the objects, parameters, and
    quantifiers; spell out what is assumed and what is concluded; write down the negation so you
    know exactly what a disproof would entail. If the statement admits more than one reading,
    enumerate the readings and commit to the hardest one; retreat to a weaker reading only if the
     hardest reading is genuinely out of reach, and when you do, say so plainly.

2. **Plan.** Before computing, generate a short portfolio of candidate strategies. You are
    encouraged to draw on the problem-solving toolkit below. Pick one primary strategy and one
    backup. If the problem has natural parameters, plan to solve a small, nontrivial warm-up case
     first (e.g., small $n$, $d=1$ or $d=2$, deterministic before stochastic, unconstrained
    before constrained). A clean proof of a representative special case often makes the general
    case legible and is itself a publishable partial result. **Use the web here** to orient
    yourself: locate relevant techniques, surveys, and named theorems in the area; identify
    closest-variant problems that are already solved; and take note of methods that have
    succeeded on near neighbors.

3. **Execute.** Carry out the primary strategy as a rigorous proof. Use theorems and lemmas as
    the main structural unit. Define notation before using it. Avoid the phrases *clearly*, *
    obviously*, *easy to see*, *standard argument shows*, and *by symmetry* as substitutes for a
    derivation --- if an inference is genuinely routine, spend one or two extra lines actually
    making it. If the primary strategy fails, fall back to the backup and keep a clean record of
    what obstruction the primary strategy hit.

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.
```

```
   - Re-check that your proof does not secretly prove a statement that is known to fail (check
     degenerate / boundary / limiting cases).
   - Re-read every step and ask: is this a definition, a calculation, a citation, or a nontrivial
     inference? Nontrivial inferences must be fully justified in line.
   - **Cross-check each named citation on the web** --- see "Citation hygiene" below.

## 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.
- **Non-elementary methods are welcome.** There is no requirement that the proof be short or
    elementary.
- **Be explicit about assumptions.** State every hypothesis you use. Flag any step that assumes
    more than the problem provides.
- **Counterexample discipline.** Never stop at the first counterexample --- minimize its size and
     the complexity of its parameters, and report the search space you explored.
- **Do not over-code abstract questions.** On questions about conceptual mathematical structures,
     favor pen-and-paper reasoning over numerical experimentation. If you are stuck in repeated
    computational errors, step back and reason symbolically.
- **Report the idea trail.** If the verdict is not `SOLVED` / `DISPROVED`, your output must
    include every substantive approach you tried. Do not replace failed attempts with an
    unrelated result from the literature that happens to be provable.
- **Quote the problem.** The restatement you produce must be logically equivalent to the input
    statement. If you change it, justify the change explicitly.

## Web-use protocol

Treat the web as a tool for navigation, verification, and method discovery --- not as a source of
     ready-made proofs.

1. **Separate what you prove from what you cite.** Every line of the final proof must either (a)
    be derived by you in the manuscript, or (b) carry an explicit citation whose statement and
    hypotheses you have audited. Maintain a strict visual separation between these.

2. **Do not paste completed proofs.** If a reference appears to solve the exact problem, record
    that fact, record the citation, and then still write your own, independent derivation. If you
     cannot reproduce the argument independently, clearly label the solution as `RELIES-ON-
    EXTERNAL-PROOF` and describe exactly what you imported and what hypotheses you verified.

3. **Source classification.** For every source you use or find potentially relevant, classify it
    as one of:
   - `SOLVES` --- claims to resolve the exact statement (or a logical equivalent).
   - `PARTIAL` --- resolves a strictly weaker or special case.
   - `TOOL` --- supplies a method or lemma you invoke, but does not address the problem directly.
   - `BACKGROUND` --- provides context, definitions, or notation.

   Only `TOOL` and `BACKGROUND` sources are used as steps inside your proof. `SOLVES` and `
    PARTIAL` sources are reported in a dedicated section even when you also derive the result
    yourself.

4. **Citation hypothesis-match audit (non-optional).** For every cited theorem used as a step in
    the proof, write out its full statement and verify, hypothesis by hypothesis, that each
    assumption of the cited theorem is met in the current setting. A citation without such an
    audit is not allowed. Beware of lookalikes whose hypotheses differ by a seemingly-minor
```

```
    condition (compactness, finiteness, convexity, boundedness, integrability, independence,
    regularity).

5. **Source quality.** Prefer peer-reviewed journals, authoritative surveys, and reputable
    monographs. Preprints (arXiv, HAL, working papers) are acceptable but must be flagged as such
    . Avoid Wikipedia and blog posts as primary citations --- use them only as pointers toward
    primary sources.

6. **Bibliography.** Provide a BibTeX entry for every source used or prominently discussed. Use
    author, title, venue, year, DOI or arXiv id when available.

## Problem-solving toolkit (pick one or two that seem most relevant)

1. **Modify the problem.** Simplify (small cases, reduce to another problem), generalize (add a
    parameter, strengthen the inductive hypothesis), work with a similar object (continuous/
    discrete analogue), take the contrapositive, make conjectures.
2. **Use analogy.** Think about what you know about similar problems; try imitating arguments
    that worked in related settings. Web search is especially useful here --- look for solved
    near-neighbors.
3. **Strip what you don't need.** Ask what could possibly work. Do not impose constraints that
    aren't there. Use reductions; consider what weaker statement would already suffice. Make sure
     you are using every hypothesis; and check whether any hypothesis is in fact unnecessary.
4. **Give names and notation.** Good notation is half the proof. Introduce definitions for
    recurring objects. Consider multiple conceptual frames (geometric, algebraic, probabilistic,
    combinatorial, optimization-dual, information-theoretic) and adopt the one that makes the
    structure cleanest.

Polya's *How to Solve It* is a useful reference in spirit.

## Output format -- a single LaTeX code block

Emit exactly one LaTeX code block (fenced with triple backticks and the language tag `latex`) and
     nothing else before or after it. The first non-empty line inside the block must be one of `%
     Verdict: SOLVED`, `% Verdict: DISPROVED`, `% Verdict: PARTIAL`, or `% Verdict: NO-PROGRESS`.
     The block is consumed by a downstream formatter and verifier; do not include `\documentclass
    `, `\begin{document}`, `\end{document}`, or package imports.

1. A one-line LaTeX comment at the very top giving your verdict:
   `% Verdict: SOLVED` / `% Verdict: DISPROVED` / `% Verdict: PARTIAL` / `% Verdict: NO-PROGRESS`.

2. `\section*{Problem}` --- concise restatement in your own notation, with all hypotheses and
    quantifiers.
3. `\section*{Alternative readings considered}` --- short; empty if the statement is unambiguous.
4. `\section*{Related literature found}` --- brief summary of the web search you performed,
    listing each source you consulted with its classification (`SOLVES` / `PARTIAL` / `TOOL` / `
    BACKGROUND`) and a one-sentence note on relevance. Include sources that looked promising but
    turned out not to apply, and say why they did not apply.
5. `\section*{Proof sketch}` --- a conceptual outline (a few paragraphs) of the argument you will
     then execute, so that a reader can follow the logical flow before the formal proof.
6. `\section*{Main result}` --- a `\begin{theorem} ... \end{theorem}` block stating exactly what
    you prove (the original statement, its negation, or a precise weaker variant).
7. `\section*{Proof}` --- the rigorous proof, preceded by lemma blocks as needed (`\begin{lemma}
    ... \end{lemma}`, `\begin{proof} ... \end{proof}`). At every citation, insert a short "
    Hypothesis audit" remark verifying that each hypothesis of the cited theorem holds in the
    present setting.
8. `\section*{Gap to the full statement}` --- present whenever the main result is weaker than the
     full problem; precisely state what remains unproved.
9. `\section*{Self red-team}` --- the counterexample attempts you ran, the three most plausible
    adversarial objections, the boundary / degenerate cases you checked, and what survived.
```

```
10. `\section*{Ideas tried}` --- mandatory when the verdict is `PARTIAL` or `NO-PROGRESS`. Aim
    for at least three distinct, substantively different approaches. For each: the idea, the
    specific obstruction, and the most natural next step.
11. `\section*{Invoked results}` --- every cited theorem used as a step in the proof. For each:
    the full statement; classification (`TOOL` or --- if essential --- `SOLVES`/`PARTIAL`); the
    citation (author, year, venue, DOI/arXiv id); and the hypothesis-match audit.
12. `\section*{Bibliography}` --- BibTeX entries for every source you used or prominently
    discussed. Use standard BibTeX syntax inside `\begin{filecontents*}{refs.bib} ... \end{
    filecontents*}` or as plain `@article{...}` entries.
13. `\section*{Self-confidence}` --- a single integer from 0 to 4 and a one-sentence
    justification, using this scale:
    - 0 = I am not confident any step is right.
    - 1 = The high-level plan seems plausible but I could not verify critical steps.
    - 2 = Most steps are checked but at least one nontrivial step is a leap I could not justify.
    - 3 = I checked every step and could not break the proof, but I may have missed something.
    - 4 = I am as confident as I can be without external verification that the proof is complete
    and correct.

Keep the LaTeX valid and self-contained at the body level; do not introduce custom packages or
    unusual macros.

## Problem statement

{PROBLEM_LATEX}
```

A.5 Automated verifier

System message.

```
You are a mathematical reviewer and writer agent. You do not have web access.

Your task is to review the mathematical problem and proposed solution below.

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.

Output format:

Return exactly one fenced code block and nothing else before or after it.

If the solution is publication-ready, return exactly one fenced `latex` code block whose first
    non-empty line is:
```

```
‘% Verdict: VERIFIED‘

Then write a body-level LaTeX paper with the following sections in order:

1. ‘\section*{Title}‘
2. ‘\section*{Abstract}‘
3. ‘\section*{Problem statement}‘
4. ‘\section*{Main result}‘
5. ‘\section*{Proof}‘
6. ‘\section*{Remarks}‘

Use theorem / lemma / proof environments when useful. Do not include ‘\documentclass‘, package
    imports, ‘\begin{document}‘, or ‘\end{document}‘.

If you find issues, return exactly one fenced ‘text‘ code block whose first non-empty line is:

‘% Verdict: ISSUES‘

Then provide a flat bullet list of the concrete issues that block publication. Each bullet should
     identify a specific flaw or missing justification.

Problem statement:

{PROBLEM_LATEX}

Proposed solution:

{PROBLEM_SOLUTION}
```

A.6 Manual solver

User message.

```
Solve the open problem in the following link. Ignore claims that the problem is open and be brave
     and solve the problem yourself.

{PROBLEM_WEBSITE_ENTRY_URL}
```

A.7 Manual custom instructions

Custom instructions.

```
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.
```

```
Unless explicitly asked, avoid summarizing previous messages.

Be open to using non-elementary methods to solve questions. There is never a requirement that
    your solution has to be simple.

When searching for a mathematical counterexample, don't stop after finding one. Always aim to
    find the simplest possible one with the smallest size and the simplest numbers.

Never feel rushed to answer a mathematical question. Think calmly and systematically.
```

A.8 Manual verifier

User message.

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

{PROPOSED_SOLUTION}
```

A.9 Manual follow-up after successful verification

User message.

```
Can you write up the proof as a clean, latex paper?

{VERIFIER_ANALYSIS_AND_VERIFIED_SOLUTION}
```

A.10 Manual follow-up after failed verification

User message.

```
Can you try and fix this, and write up your progress as a clean, latex paper?

{VERIFIER_ISSUES_AND_PROPOSED_SOLUTION}
```

A.11 Manual second solver pass after attempted repair

User message.

```
The following is an attempt to solve the problem in this link:

{PROBLEM_WEBSITE_ENTRY_URL}

Can you solve the problem? Ignore claims that the problem is open and be brave and solve the
    problem yourself.

{ATTEMPTED_REPAIR_OR_LATEX_WRITEUP}
```

B Problem 21: a fully automated case study

The solution below is generated from our website. In contrast to Problem 47, the following solution has NOT been edited or verified by a human, and we add it here for the sake of completeness. We plan to verify this solution in future iterations of our paper.

Problem 21 comes from Diao, Dai, and Zhang's work on stability for Nash equilibrium problems.[2] Studying when the KKT solution mapping of a parameterized Nash equilibrium problem is isolatedly calm, the paper introduces a second-order "I-property" and proves that it is *sufficient* for isolated calmness, leaving its necessity open. The database entry is displayed as "Characterize necessity of the I-property for isolated calmness of the KKT mapping," and the sentence extracted as evidence of openness was:

> Nevertheless, there are many other unresolved stability issues in NEPs. For instance, is the I-property also necessary for the isolated calmness of S_{KKT}?

This problem was solved end-to-end by the automated solver–verifier loop, with no manual prompting: the solver returned the verdict `DISPROVED`, and the adversarial verifier returned `VERIFIED` together with the write-up reproduced below.

B.1 Model and result

Consider a Nash equilibrium problem (NEP) with N players. Player k chooses a strategy $x_k \in \mathbb{R}^{n_k}$ and, given the strategies x_{-k} of the other players and a perturbation parameter $p_k = (u^k, v_k, w_k)$, solves

$$\min_{x_k \in \mathbb{R}^{n_k}} \; f_k(x_k, x_{-k}; w_k) - \langle v_k, x_k \rangle \quad \text{s.t.} \quad g_i^k(x_k; w_k) = u_i^k \;\; (i \le s_k), \quad g_j^k(x_k; w_k) \le u_j^k \;\; (j > s_k).$$

Write $x = (x_1, \dots, x_N) \in \mathbb{R}^n$ with $n = \sum_k n_k$, and $p = (p_1, \dots, p_N)$. With Lagrange multipliers $\lambda_k \in \mathbb{R}^{m_k}$ (nonnegative for the inequality constraints), the ordinary Lagrangian of player k is

$$L_k(x_k, x_{-k}, \lambda_k; w_k) = f_k(x_k, x_{-k}; w_k) + \sum_{i=1}^{m_k} \lambda_i^k \, g_i^k(x_k; w_k).$$

The (set-valued) KKT solution mapping S_{KKT} assigns to p the set of KKT pairs (x, λ) satisfying the players' stationarity, feasibility, and complementarity conditions. Fix a reference parameter $\bar{p}$ and a reference KKT point $(\bar{x}, \bar{\lambda}) \in S_{\mathrm{KKT}}(\bar{p})$.

The mapping S_{KKT} is *isolated calm* at $\bar{p}$ for $(\bar{x}, \bar{\lambda})$ if there exist neighborhoods U of $\bar{p}$, V of $(\bar{x}, \bar{\lambda})$, and a constant $\kappa \ge 0$ such that

$$S_{\mathrm{KKT}}(p) \cap V \subset \{(\bar{x}, \bar{\lambda})\} + \kappa \|p - \bar{p}\| B \qquad \forall p \in U,$$

where B is the unit ball. For each player k, let I_1^k collect the equality constraints together with the active inequality constraints having a strictly positive multiplier, and let I_2^k collect the active inequality constraints with zero multiplier; set $I_1 = \bigcup_k I_1^k$, $I_2 = \bigcup_k I_2^k$, and define the critical cone

$$K(I_1, I_2) := \Big\{ y = (y_1, \dots, y_N) \in \mathbb{R}^n : \; \nabla_{x_k} g_i^k(\bar{x}_k; \bar{w}_k)^\top y_k = 0 \; \forall i \in I_1^k, \;\; \nabla_{x_k} g_i^k(\bar{x}_k; \bar{w}_k)^\top y_k \le 0 \; \forall i \in I_2^k \Big\}.$$

The paper's *I-property* at $(\bar{p}, \bar{x}, \bar{\lambda})$ on $K(I_1, I_2)$ states that, for any $y \in K(I_1, I_2)$, if for every $k = 1, \dots, N$

$$\sum_{i=1}^{N} y_k^\top \, \nabla^2_{x_k x_i} L_k(\bar{x}_k, \bar{x}_{-k}, \bar{\lambda}_k; \bar{w}_k) \, y_i = 0,$$

then $y = 0$. The paper proves the I-property is sufficient for isolated calmness of S_{KKT}; the open question is whether it is also necessary.

[2] Ruoyu Diao, Yu-Hong Dai, and Liwei Zhang, *Stability for Nash Equilibrium Problems*, *Mathematics of Operations Research*, 2025; DOI: 10.1287/moor.2024.0609. Database entry: https://pranav-nuti.github.io/open-problems-in-or/problem.html?id=W4411959281_p0.

Theorem B.1. *The implication "S_{KKT} is isolated calm at $\bar{p}$ for $(\bar{x}, \bar{\lambda}) \implies$ the I-property holds at $(\bar{p}, \bar{x}, \bar{\lambda})$ on $K(I_1, I_2)$" is false in general. More precisely, there exists a smooth two-player Nash equilibrium problem with canonical perturbations for which the KKT solution mapping S_{KKT} is isolated calm at a reference parameter for a reference KKT point, yet the I-property fails at that point.*

B.2 Proof of Theorem B.1

Proof. Consider $N = 2$, $n_1 = n_2 = 2$, and no constraints, so $m_1 = m_2 = 0$; the multiplier component is vacuous. Introduce dummy parameters $w_1, w_2 \in \mathbb{R}$ on which the objectives do not depend. For $x_1, x_2 \in \mathbb{R}^2$, define

$$f_1(x_1, x_2; w_1) = \tfrac{1}{2}\|x_1\|^2 + x_1^\top A x_2, \qquad f_2(x_2, x_1; w_2) = \tfrac{1}{2}\|x_2\|^2 + x_2^\top A^\top x_1, \qquad A = \begin{pmatrix} -1 & 0 \\ 1 & 0 \end{pmatrix}.$$

The perturbed player problems are therefore

$$\min_{x_1 \in \mathbb{R}^2} \ \tfrac{1}{2}\|x_1\|^2 + x_1^\top A x_2 - \langle v_1, x_1 \rangle, \qquad \min_{x_2 \in \mathbb{R}^2} \ \tfrac{1}{2}\|x_2\|^2 + x_2^\top A^\top x_1 - \langle v_2, x_2 \rangle.$$

Set $\bar{p} = (\bar{v}_1, \bar{v}_2, \bar{w}_1, \bar{w}_2) = (0, 0, 0, 0)$ and $\bar{x} = (0, 0)$. Since there are no constraints, the Lagrangians are the objectives, $L_1 = f_1$ and $L_2 = f_2$.

Isolated calmness. For this unconstrained game the KKT system is the stationarity system $\nabla_{x_1} f_1(x_1, x_2) - v_1 = 0$ and $\nabla_{x_2} f_2(x_2, x_1) - v_2 = 0$. Since $\nabla_{x_1} f_1(x_1, x_2) = x_1 + A x_2$ and $\nabla_{x_2} f_2(x_2, x_1) = x_2 + A^\top x_1$, the system becomes

$$x_1 + A x_2 = v_1, \qquad A^\top x_1 + x_2 = v_2.$$

Writing $x = (x_1, x_2) \in \mathbb{R}^4$ and $v = (v_1, v_2) \in \mathbb{R}^4$, this is $Mx = v$ with $M = \begin{pmatrix} I_2 & A \\ A^\top & I_2 \end{pmatrix}$. Since $A^\top A = \begin{pmatrix} 2 & 0 \\ 0 & 0 \end{pmatrix}$, the Schur complement of the upper-left block I_2 is $I_2 - A^\top A = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$, with determinant $-1 \neq 0$. Hence $\det M = \det(I_2) \det(I_2 - A^\top A) \neq 0$, so M is invertible. Therefore, for every parameter p,

$$S_{\mathrm{KKT}}(p) = \{(M^{-1} v,\, 0_{\mathbb{R}^0})\},$$

and in particular $S_{\mathrm{KKT}}(\bar{p}) = \{(\bar{x}, 0_{\mathbb{R}^0})\}$. Moreover

$$\|(M^{-1} v, 0_{\mathbb{R}^0}) - (\bar{x}, 0_{\mathbb{R}^0})\| = \|M^{-1} v\| \leq \|M^{-1}\| \, \|v\| \leq \|M^{-1}\| \, \|p - \bar{p}\|,$$

so S_{KKT} is isolated calm at $\bar{p}$ for $(\bar{x}, 0_{\mathbb{R}^0})$ (indeed it is single-valued and globally Lipschitz).

Failure of the I-property. Since $m_1 = m_2 = 0$, all index sets I_1^k and I_2^k are empty, so $K(I_1, I_2) = \mathbb{R}^4$. The relevant Hessians are

$$\nabla^2_{x_1 x_1} L_1 = I_2, \quad \nabla^2_{x_1 x_2} L_1 = A, \quad \nabla^2_{x_2 x_1} L_2 = A^\top, \quad \nabla^2_{x_2 x_2} L_2 = I_2.$$

Let $e = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and take $y_1 = e$, $y_2 = e$, so $y = (y_1, y_2) \neq 0$. Then

$$I_2 e + A e = \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \begin{pmatrix} -1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \qquad \text{so} \qquad y_1^\top (\nabla^2_{x_1 x_1} L_1 \, y_1 + \nabla^2_{x_1 x_2} L_1 \, y_2) = e^\top \begin{pmatrix} 0 \\ 1 \end{pmatrix} = 0.$$

Likewise

$$A^\top e + I_2 e = \begin{pmatrix} -1 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \qquad \text{so} \qquad y_2^\top (\nabla^2_{x_2 x_1} L_2\, y_1 + \nabla^2_{x_2 x_2} L_2\, y_2) = e^\top \begin{pmatrix} 0 \\ 0 \end{pmatrix} = 0.$$

Thus $y \in K(I_1, I_2)$, $y \neq 0$, and for each player $k = 1, 2$ the quadratic form $\sum_{i=1}^2 y_k^\top \nabla^2_{x_k x_i} L_k\, y_i$ vanishes. The premise of the I-property therefore holds for a nonzero y, which means the I-property fails. Hence isolated calmness of S_{KKT} does not imply the I-property in general. □

Remark 1. *The counterexample is fully smooth and unconstrained, so the failure is not caused by pathological constraints. Each player's objective is strongly convex in its own decision variable (the Hessian with respect to x_k is I_2), so the stationarity condition for each player is sufficient for optimality, and $\bar{x} = (0, 0)$ is in fact the unique Nash equilibrium at $\bar{p}$. Consequently, any valid converse implication from isolated calmness to the I-property must require additional hypotheses beyond isolated calmness alone.*

C Problem 24: a mixed automated and manual case study

The solution below is generated from our website. In contrast to Problem 47, the following solution has NOT been edited or verified by a human, and we add it here for the sake of completeness. We plan to verify this solution in future iterations of our paper.

Problem 24 comes from Luner and Grimmer's work on averaging and extrapolation for gradient descent.[3] Their paper studies whether averaging the iterates of gradient descent can improve worst-case guarantees over simply returning the last iterate, and proves that, whenever the last-iterate worst case matches an explicit Huber-type benchmark, the last iterate is in fact optimal among all averages. Conjecture 3.3 of their paper asserts the converse. Our pipeline extracted that conjecture as the open problem; the database entry is displayed as "Prove tight worst-case characterization of gradient descent averaging benefit conditions," and the sentence extracted as evidence of openness was:

> Numerical results suggest that the converses of Theorem 3.1 and Theorem 3.2 are true, i.e., (C1)/(C2) hold if and only if averaging is not beneficial. We state this formally in the following conjecture.

This problem was solved by handing the output of the automated solver–verifier loop to the manual workflow, which produced the solution below: an explicit one-step counterexample disproving the conjecture.

C.1 Model and result

Fix $L > 0$, $D > 0$, and a dimension $m \geq 1$. Let $\mathcal{F}_L(\mathbb{R}^m)$ denote the class of convex differentiable functions $f : \mathbb{R}^m \to \mathbb{R}$ whose gradient is L-Lipschitz. An admissible instance is a triple

$$(f, x_0, x^\star) \quad \text{with } f \in \mathcal{F}_L(\mathbb{R}^m), \quad x^\star \in \arg\min f, \quad \|x_0 - x^\star\| \leq D.$$

[3] Alan Luner and Benjamin Grimmer, *On Averaging and Extrapolation for Gradient Descent*, *Mathematics of Operations Research*, 2025; arXiv:2402.12493. Database entry: https://pranav-nuti.github.io/open-problems-in-or/problem.html?id=W4413077228_p0.

Denote the set of such instances by $\mathcal{I}_{L,D,m}$. In the one-step case with dimensionless stepsize $h > 0$, gradient descent produces

$$x_1 = x_0 - \frac{h}{L}\nabla f(x_0).$$

For $\sigma = (\sigma_0, \sigma_1)$ in the simplex $\Delta_2 = \{(\sigma_0, \sigma_1) \in \mathbb{R}^2_+ : \sigma_0 + \sigma_1 = 1\}$, the averaged output is $x_\sigma = \sigma_0 x_0 + \sigma_1 x_1$. The last-iterate scheme is $\sigma^{\text{last}} = (0, 1)$; we call σ nondegenerate when $\sigma \neq \sigma^{\text{last}}$.

For a fixed one-step length h, define the objective-gap and gradient-norm worst cases of the reported point x_σ by

$$W_{\text{obj}}(\sigma; h) = \sup_{(f,x_0,x^\star)\in\mathcal{I}_{L,D,m}} (f(x_\sigma) - f(x^\star)), \qquad W_{\text{grad}}(\sigma; h) = \sup_{(f,x_0,x^\star)\in\mathcal{I}_{L,D,m}} \|\nabla f(x_\sigma)\|.$$

The one-step benchmark quantities are

$$B_{\text{obj}}(h) = \frac{LD^2}{4h+2}, \qquad B_{\text{grad}}(h) = \frac{LD}{h+1}.$$

The benchmark converses under consideration are the implications

$$W_{\text{obj}}(\sigma^{\text{last}}; h) \neq B_{\text{obj}}(h) \implies \exists\, \sigma \in \Delta_2 \setminus \{\sigma^{\text{last}}\} \text{ with } W_{\text{obj}}(\sigma; h) < B_{\text{obj}}(h),$$

$$W_{\text{grad}}(\sigma^{\text{last}}; h) \neq B_{\text{grad}}(h) \implies \exists\, \sigma \in \Delta_2 \setminus \{\sigma^{\text{last}}\} \text{ with } W_{\text{grad}}(\sigma; h) < B_{\text{grad}}(h).$$

These are precisely the one-step versions of the displayed benchmark converses in which $\sum_i h_i = h$.

We prove that both converses fail already for a single gradient step.

Theorem C.1 (Failure of the benchmark converses)**.** *The benchmark objective-gap converse and benchmark gradient-norm converse are false already for $N = 1$. Fix a one-step length $h > 0$.*

1. *If $h > 3/2$, then $W_{\text{obj}}(\sigma^{\text{last}}; h) > B_{\text{obj}}(h)$, yet every nondegenerate $\sigma \in \Delta_2 \setminus \{\sigma^{\text{last}}\}$ satisfies $W_{\text{obj}}(\sigma; h) > B_{\text{obj}}(h)$. Thus the antecedent of the objective-gap converse holds while its conclusion fails.*
2. *If $h > \sqrt{2}$, then $W_{\text{grad}}(\sigma^{\text{last}}; h) > B_{\text{grad}}(h)$, yet every nondegenerate σ satisfies $W_{\text{grad}}(\sigma; h) > B_{\text{grad}}(h)$. Thus the antecedent of the gradient-norm converse holds while its conclusion fails.*

C.2 Relation to previous literature

The benchmark converse comes from Luner and Grimmer's study of when averaging the iterates of fixed-step gradient descent can help. They prove the forward direction—whenever the last-iterate worst case equals the Huber-type benchmark (their conditions (C1)/(C2)), the last iterate is optimal among all averages—and conjecture the converse that we refute here. The automated forward-citation review attached to this database entry surfaces a line of closely related work, all of which reinforces the empirical alignment that motivated the conjecture without resolving it.

Several papers enlarge the catalogue of stepsize schedules for which the last-iterate worst case is known to take exactly the (C1)/(C2) form. Wang, Ma, Yang, and Zhou establish tight last-iterate bounds for gradient descent under the silver stepsize schedule, matching the benchmark expressions.[4] Grimmer, Shu, and Wang prove tight last-iterate guarantees for certain long-step schedules and, importantly for the present question, exhibit explicit extremal quadratic and Huber instances and observe that for those schedules averaging is strictly worse than returning the last

[4] Bofan Wang, Shiqian Ma, Junfeng Yang, and Danqing Zhou, *Relaxed Proximal Point Algorithm: Tight Complexity Bounds and Acceleration without Momentum*, *INFORMS Journal on Optimization*, 2025; DOI: 10.1287/ijoo.2025.0075.

iterate[5]—consistent with the "no averaging benefit" side of the conjectured equivalence. The same authors' composition theory builds schedules with certifiable tight last-iterate certificates, giving a systematic way to generate further instances where (C1)/(C2) hold.[6]

What none of this work addresses is the converse implication itself: whether failure of the last-iterate benchmark identity forces the existence of a strictly better nondegenerate average. Our solution settles this negatively. The construction is in the same spirit as the extremal Huber and quadratic instances above, but it is used in the opposite direction: rather than certifying tightness for a cleverly chosen schedule, a single Huber/quadratic pair already at $N = 1$ shows that the antecedent of the converse can hold while its conclusion fails. The mechanism is the elementary observation that, after one step, averaging *is* a shorter gradient step, so no nondegenerate average can beat the one-step benchmark.

C.3 Proof of Theorem C.1

We first record two Huber lower bounds and an overshoot computation, and then combine them.

C.3.1 Huber lower bounds for one effective step

We first record two elementary lower bounds. They use only explicit Huber functions and do not rely on any external performance-estimation theorem.

Lemma 1 (A smooth Huber family)**.** *For $L, \eta > 0$, define $\Phi_{L,\eta} : \mathbb{R} \to \mathbb{R}$ by*

$$\Phi_{L,\eta}(u) = \begin{cases} \frac{L}{2}u^2, & |u| \le \eta, \\ L\eta|u| - \frac{L}{2}\eta^2, & |u| > \eta. \end{cases}$$

Then $\Phi_{L,\eta}$ is convex and L-smooth, meaning that its derivative is globally L-Lipschitz. Consequently, for any $m \ge 1$, the function $f(x) = \Phi_{L,\eta}(x_1)$, $x = (x_1, \dots, x_m) \in \mathbb{R}^m$, is convex and L-smooth on $\mathbb{R}^m$, and 0 is a minimizer of f.

Proof. The derivative of $\Phi_{L,\eta}$ is

$$\Phi'_{L,\eta}(u) = \begin{cases} -L\eta, & u \le -\eta, \\ Lu, & |u| \le \eta, \\ L\eta, & u \ge \eta, \end{cases} \qquad \text{equivalently} \qquad \Phi'_{L,\eta}(u) = \operatorname{proj}_{[-L\eta,\, L\eta]}(Lu).$$

Since projection onto an interval is nondecreasing and 1-Lipschitz, $\Phi'_{L,\eta}$ is nondecreasing and L-Lipschitz. Thus $\Phi_{L,\eta}$ is convex and L-smooth. For $f(x) = \Phi_{L,\eta}(x_1)$ we have $\nabla f(x) = \Phi'_{L,\eta}(x_1)e_1$, where $e_1 = (1, 0, \dots, 0)$. Hence, for all $x, y \in \mathbb{R}^m$,

$$\|\nabla f(x) - \nabla f(y)\| = |\Phi'_{L,\eta}(x_1) - \Phi'_{L,\eta}(y_1)| \le L|x_1 - y_1| \le L\|x - y\|.$$

Convexity of f follows from convexity of $\Phi_{L,\eta}$ and linearity of $x \mapsto x_1$. Finally $\Phi_{L,\eta} \ge 0$ and $\Phi_{L,\eta}(0) = 0$, so 0 is a minimizer of f. □

Lemma 2 (Huber lower bounds for one effective step)**.** *Fix $t \ge 0$ and let $e_1 = (1, 0, \dots, 0) \in \mathbb{R}^m$.*

[5] Benjamin Grimmer, Kevin Shu, and Alex L. Wang, *Accelerated Objective Gap and Gradient Norm Convergence for Gradient Descent via Long Steps*, *INFORMS Journal on Optimization*, 2025; DOI: 10.1287/ijoo.2024.0057.

[6] Benjamin Grimmer, Kevin Shu, and Alex L. Wang, *Composing Optimized Stepsize Schedules for Gradient Descent*, arXiv:2410.16249, 2024.

1. *There exists an instance* $(f, x_0, x^\star) \in \mathcal{I}_{L,D,m}$ *such that, for* $x^+ = x_0 - \frac{t}{L}\nabla f(x_0)$*, one has* $f(x^+) - f(x^\star) = \dfrac{LD^2}{4t+2}$.
2. *There exists an instance* $(f, x_0, x^\star) \in \mathcal{I}_{L,D,m}$ *such that, for the same update rule,* $\|\nabla f(x^+)\| = \dfrac{LD}{t+1}$.

Proof. Set $x^\star = 0$ and $x_0 = De_1$, so $\|x_0 - x^\star\| = D$.

For the objective-gap bound, choose $\eta = \dfrac{D}{2t+1}$ and $f(x) = \Phi_{L,\eta}(x_1)$. By Lemma 1 this is admissible. Since $D \geq \eta$, we have $\nabla f(x_0) = L\eta e_1$, so

$$x^+ = De_1 - t\eta e_1 = \frac{(t+1)D}{2t+1}e_1,$$

and $\frac{(t+1)D}{2t+1} \geq \frac{D}{2t+1} = \eta$, so x^+ lies on the affine branch of the Huber function. Hence

$$f(x^+) - f(x^\star) = L\eta\,\frac{(t+1)D}{2t+1} - \frac{L}{2}\eta^2 = \frac{LD^2}{(2t+1)^2}\Big(t+1-\tfrac{1}{2}\Big) = \frac{LD^2}{2(2t+1)} = \frac{LD^2}{4t+2}.$$

For the gradient-norm bound, choose instead $\eta = \dfrac{D}{t+1}$ and $f(x) = \Phi_{L,\eta}(x_1)$. Again $D \geq \eta$ and $\nabla f(x_0) = L\eta e_1$, so

$$x^+ = De_1 - t\eta e_1 = \Big(D - \tfrac{tD}{t+1}\Big)e_1 = \frac{D}{t+1}e_1 = \eta e_1.$$

At $u = \eta$ the derivative of $\Phi_{L,\eta}$ is $L\eta$, so $\|\nabla f(x^+)\| = L\eta = \dfrac{LD}{t+1}$. □

C.3.2 Quadratic overshoot of the last iterate

Lemma 3 (Quadratic overshoot)**.** *Let* $q(x) = \frac{L}{2}x_1^2$*,* $x_0 = De_1$*,* $x^\star = 0$*. For the one-step update* $x_1 = x_0 - \frac{h}{L}\nabla q(x_0)$ *one has* $x_1 = (1-h)De_1$*, and consequently*

$$q(x_1) - q(x^\star) = \frac{LD^2}{2}(1-h)^2, \qquad \|\nabla q(x_1)\| = LD\,|1-h|.$$

In particular,

$$h > \tfrac{3}{2} \implies \frac{LD^2}{2}(1-h)^2 > \frac{LD^2}{4h+2}, \qquad h > \sqrt{2} \implies LD\,|1-h| > \frac{LD}{h+1}.$$

Proof. Since $\nabla q(x) = Lx_1e_1$, we have $\nabla q(x_0) = LDe_1$, hence $x_1 = De_1 - \frac{h}{L}LDe_1 = (1-h)De_1$; the displayed formulas follow immediately. If $h > 3/2$, then $h > 1$ and

$$\frac{(h-1)^2}{2} - \frac{1}{4h+2} = \frac{(h-1)^2(2h+1)-1}{2(2h+1)} = \frac{2h^3-3h^2}{2(2h+1)} = \frac{h^2(2h-3)}{2(2h+1)} > 0;$$

multiplying by LD^2 gives the objective-gap inequality. If $h > \sqrt{2}$, then $h > 1$ and

$$(h-1) - \frac{1}{h+1} = \frac{(h-1)(h+1)-1}{h+1} = \frac{h^2-2}{h+1} > 0;$$

multiplying by LD gives the gradient-norm inequality. □

C.3.3 Putting the pieces together

Proof of Theorem C.1. Let $\sigma = (\sigma_0, \sigma_1) \in \Delta_2$. For every admissible instance,

$$x_\sigma = \sigma_0 x_0 + \sigma_1 x_1 = \sigma_0 x_0 + \sigma_1\Big(x_0 - \tfrac{h}{L}\nabla f(x_0)\Big) = x_0 - \frac{h\sigma_1}{L}\nabla f(x_0).$$

Thus, after one gradient step, averaging is exactly one gradient step with effective dimensionless stepsize $t = h\sigma_1$. If σ is nondegenerate then $\sigma_1 < 1$, so $0 \le t < h$.

We prove the objective claim; assume $h > 3/2$. Applying Lemma 3 to the quadratic instance gives

$$W_{\mathrm{obj}}(\sigma^{\mathrm{last}}; h) \ge \frac{LD^2}{2}(h-1)^2 > \frac{LD^2}{4h+2} = B_{\mathrm{obj}}(h),$$

so the last-iterate worst case is not the benchmark. Now take any nondegenerate σ and set $t = h\sigma_1 < h$. Since x_σ is one effective step of length t, Lemma 2 gives $W_{\mathrm{obj}}(\sigma; h) \ge \frac{LD^2}{4t+2}$. As $u \mapsto 1/(4u+2)$ is strictly decreasing and $t < h$,

$$\frac{LD^2}{4t+2} > \frac{LD^2}{4h+2} = B_{\mathrm{obj}}(h).$$

Hence $W_{\mathrm{obj}}(\sigma; h) > B_{\mathrm{obj}}(h)$ for every nondegenerate σ. The gradient-norm argument is identical, using $h > \sqrt{2}$, Lemma 3, Lemma 2, and the strict monotonicity of $u \mapsto 1/(u+1)$. □

Corollary 1 (A simultaneous counterexample with $h < 2$)**.** *Take $N = 1$ and $h = 7/4$. Then both benchmark converses fail simultaneously. Explicitly,*

$$B_{\mathrm{obj}}(7/4) = \frac{LD^2}{9}, \qquad B_{\mathrm{grad}}(7/4) = \frac{4LD}{11},$$

while the quadratic instance has last-iterate values

$$q(x_1) - q(x^\star) = \frac{9LD^2}{32} > \frac{LD^2}{9}, \qquad \|\nabla q(x_1)\| = \frac{3LD}{4} > \frac{4LD}{11},$$

so both last-iterate benchmark identities fail; nevertheless every nondegenerate average $\sigma \ne (0,1)$ satisfies $W_{\mathrm{obj}}(\sigma; 7/4) > \frac{LD^2}{9}$ and $W_{\mathrm{grad}}(\sigma; 7/4) > \frac{4LD}{11}$.

Proof. The value $h = 7/4$ satisfies $h > 3/2$ and $h > \sqrt{2}$, so the result follows from Theorem C.1; the displayed numbers are obtained by direct substitution into Lemma 3 and the definitions of B_{obj} and B_{grad}. □

Remark 1 (Scope)**.** *The theorem refutes the benchmark converse: failure of the last-iterate benchmark identity need not imply the existence of a nondegenerate average whose worst-case guarantee is strictly below the same Huber benchmark. It does not address the logically different statement that, whenever the benchmark identity fails, some average improves on the* actual *worst-case performance of the last iterate. In the counterexample above the actual last-iterate worst case is larger than the Huber benchmark, and these two comparison targets should not be conflated.*

Remark 2 (Dimension)**.** *All instances used in the proof are one-dimensional, or depend only on the first coordinate. Hence the counterexample works in dimension $m = 1$ and embeds verbatim in every dimension $m \ge 1$.*