SGU-Editorial: A Small Dataset of Competitive Programming Problems with LLM-Enhanced Editorials

Radoslav Dimitrov
contact@radoslav11.com

Abstract

State-of-the-art large language models (LLMs) perform poorly on problems from the Saratov State University (SGU) Online Judge, likely because SGU problems feature a diverse range of algorithmic ideas underrepresented in existing training datasets. To address this, we present SGU-Editorial, a dataset covering 452 of the 453 problems in the SGU archive. Each problem includes the original statement, an accepted solution in C++ or Python, and a detailed LLM-generated editorial with algorithm explanations, implementation guidance, and reference implementations.

Keywords: competitive programming, dataset, LLM editorials, code reasoning

1 Introduction

The intersection of large language models (LLMs) and competitive programming has emerged as an important area of research. Recent advances have shown that LLMs can solve programming problems at varying difficulty levels, from basic coding exercises to International Olympiad in Informatics (IOI) problems [1, 6]. For example, OlympicCoder models have demonstrated strong performance on IOI problems, even outperforming some closed-source models [10]. However, state-of-the-art models still struggle with certain problem sources. In particular, we observe that LLMs perform poorly on problems from the Saratov State University (SGU) Online Judge.

The SGU (Saratov State University) Online Judge, known as acm.sgu.ru, was created approximately 20 years ago by competitive programming enthusiasts from Saratov University, including Mike Mirzayanov (who later founded Codeforces) and Andrew Lazarev. The platform hosted ACM ICPC-style problems and was widely used for training. Although the original infrastructure became outdated, the problems were preserved and migrated to Codeforces as a dedicated "acmsguru" section [4], reflecting continued interest in this historical problemset.

We hypothesize that LLMs struggle with SGU problems because they feature a diverse range of algorithmic ideas and problem-solving techniques that are underrepresented in existing competitive programming datasets. The SGU archive contains problems with unique formulations and solution approaches that may not appear frequently in more commonly scraped sources like modern Codeforces contests or LeetCode.

Existing competitive programming datasets such as POJ-104 [8], COFO [2], and CodeContests [6] focus primarily on collecting source code submissions for tasks like program classification, clone detection, and code generation. While valuable, these datasets draw from a limited set of sources and typically lack detailed explanations of *why* particular algorithms work, *how* to arrive at the solution, and *what* insights are needed to solve the problem.

In competitive programming, such explanations are called *editorials*: documents that provide problem analysis, algorithm descriptions, implementation guidance, and complexity analysis. Editorials are essential learning resources but are time-consuming to write and often unavailable or incomplete for many problems.

In this work, we present SGU-Editorial, a dataset that addresses this gap. It contains:

1. A collection of 452 of the 453 competitive programming problems from the SGU Online Judge with original problem statements and *commented* accepted solutions. The only excluded problem, number 145, is currently broken on the judge: there are no passing solutions, as the checker rejects any path other than the single one it expects despite the statement permitting any valid path.

2. LLM-generated editorials for each problem, created using reasoning models (OpenAI's o1, o3, and the GPT-5 series), containing structured explanations with algorithm analysis, implementation guides, and reference solutions in both C++ and Python. For the later problems, parts of the reference implementations were produced with the assistance of coding agents (Claude Code and Codex) working from a solution idea and outline provided by the author.

2 Related Work

Several datasets have been proposed for competitive programming and code understanding tasks.

POJ-104 [8] consists of 104 programming problems from the Peking University Online Judge with approximately 500 C/C++ solutions per problem, totaling around 52K programs. It is primarily used for program classification tasks.

COFO [2] is a larger dataset scraped from Codeforces containing 809 problems with 369K source codes in C, C++, Java, and Python. It includes metadata such as problem tags, specifications, and test cases, targeting classification and tagging tasks.

CodeContests [6] is a dataset of competitive programming problems used to train AlphaCode, containing problems from Codeforces, Description2Code, and CodeNet with input-output examples and solutions.

APPS [3] contains 10,000 coding problems of varying difficulty with test cases and solutions, designed for evaluating code generation capabilities.

These datasets focus on the *code* aspect of competitive programming. In contrast, SGU-Editorial focuses on the *reasoning* aspect by providing detailed editorials that explain the problem-solving process.

3 Methodology

3.1 Source Problems

The SGU (Saratov State University) Online Judge was one of the pioneering competitive programming platforms, hosting problems from ACM ICPC-style competitions. The problems span various algorithmic topics including graph theory, dynamic programming, number theory, computational geometry, and combinatorics.

We collected 452 of the 453 problems in the SGU archive. For each problem, we obtained:

- The original problem statement with input/output specifications.

- Sample input and output test cases.
- An accepted solution written by the first author in C++ or Python.

3.2 Editorial Generation

For each problem, we generated editorials using a combination of reasoning-capable language models, including OpenAI's o1, o3, GPT-5, GPT-5.1, and GPT-5.2. The input to each model consisted of the problem statement, sample I/O, and an accepted solution. The generated editorials were then manually reviewed and edited to correct any inaccuracies or improve clarity. For the later problems in the archive, parts of the accepted solutions themselves were implemented with the help of coding agents (Claude Code and Codex), guided by a solution idea and outline supplied by the author; the high-level algorithmic approach in every case was determined by the author rather than the model.

3.3 Quality Assurance

All solutions in the dataset are *accepted* solutions that have passed the online judge's test suite. Furthermore, a large chunk of the solutions were made to already contain a brief description of the approach as comments. This makes LLM-generated editorials on average being in a good shape, but we also did do some basic spot-checking for correctness.

4 Data Description

The dataset is organized in two main directories: `dataset/` and `problems/`. The `dataset/` directory contains three files for each problem: `pXXX.txt`, the enhanced editorial; `pXXX_raw.txt`, which contains the original problem statement, solution, and sample I/O; and `pXXX_finetune.txt`, which is a version of the data formatted for model training. The `problems/` directory contains the same information as the `_raw.txt` files, but organized as individual files for each problem (e.g., `statement.txt`, `pXXX.cpp`).

Each editorial follows a consistent structure, including a concise problem statement, a detailed editorial with algorithm explanations, commented reference implementations in C++ and Python, and a brief summary for experienced programmers. Figure 1 shows short excerpts of the three file types for a single problem.

4.1 Dataset Statistics

Table 1 summarizes the dataset, and Figure 2 shows the distributions of editorial and solution sizes. We report token counts using OpenAI's `tiktoken` library with the `cl100k_base` encoding, the byte-pair [12] tokenizer used by the GPT-3.5 and GPT-4 model families. Each file is tokenized as raw UTF-8 text, so the counts include natural-language prose, code, markup, and whitespace exactly as stored; they are intended as a consistent relative measure rather than an exact match to any specific model's billing.

Each of the 452 covered problems has an editorial; 425 ship a C++ reference solution and 27 a Python one. The reference solutions average roughly 179 lines (median 140), reflecting the implementation-heavy nature of many SGU problems, and about 0.63M tokens in total. Editorials are substantially longer than the code they explain, averaging about 3,500 tokens (median 3,100) and ranging from a few hundred tokens for the simplest problems to over 13,000

p100_raw.txt (statement + accepted solution)

```
p100.cpp
======================
#include <bits/stdc++.h>
...
void solve() { cout << a + b << endl; }
...
=================
statement.txt
100. A+B ... Read integers A and B ...
```

p100.txt (editorial)

```
1. Abridged Problem Statement
Given two positive integers A and B, compute A + B.

2. Detailed Editorial
... A, B <= 10000 => sum fits in a 32-bit int ...

3. Provided C++ Solution with Detailed Comments
...
```

p100_finetune.txt (instruction/response)

```
<|instruction|>
Solve the below problem. The solution should
start with an abridged problem statement ...
100. A+B ... Read integers A and B ...
<|response|>
1. Abridged problem statement
Given two positive integers A and B ...
```

Figure 1: Excerpts of the three dataset/ files for problem 100. The _raw file bundles the statement and accepted solution; pXXX.txt is the structured editorial; _finetune reframes the same content as an instruction/response pair for training.

for the most involved ones, totaling roughly 1.6M tokens. Combined, the dataset comprises about 2.2M tokens of editorials and reference solutions.

Figure 2(d) plots editorial length against solution length for each problem. The two are positively correlated, as expected, but the spread is wide: some short solutions rely on a single non-obvious idea that takes many tokens to explain, while some long, mechanical solutions need comparatively little exposition. This is consistent with our motivation that the *reasoning* content of SGU problems is not captured by code alone.
These statistics are computed over the complete set of 452 editorials. Future versions may regenerate the editorials with stronger models, which could shift the per-editorial figures.

4.2 Thematic Clustering

To characterize what the archive actually covers, we embed each editorial with OpenAI's text-embedding-3-large model [9] and group the resulting vectors with k-means [7] ($k = 8$). A chat model then assigns each cluster a short tag and a one-line theme from a sample of its members. Figure 3 shows a t-SNE [13] projection of the editorial embeddings colored by cluster, and Table 2 lists the resulting groups. Appendix A gives the full procedure, and Appendix D examines per-problem accepted-solution counts across the archive.

The clusters recover recognizable competitive-programming categories: elementary number

Table 1: Summary statistics for SGU-Editorial. Token counts use the `cl100k_base` tokenizer.

Quantity	Value
Problems covered (of 453)	452
with a C++ reference solution	425
with a Python reference solution	27
Editorials	452
Mean editorial length (tokens)	3,543
Median editorial length (tokens)	3,119
Max editorial length (tokens)	13,608
Mean solution length (lines)	179
Median solution length (lines)	140
Mean solution length (tokens)	1,395
Editorial tokens (total)	≈1.6M
Solution tokens (total)	≈0.63M
Combined tokens	≈2.2M

Table 2: Thematic clusters of the editorials ($k = 8$), sorted by size, with the chat-model tag and theme.

Tag	#	Theme
Basic Math	79	Elementary number theory / arithmetic
Mixed Topics	67	Diverse algorithmic techniques
Geometry & DP	67	Geometry with dynamic programming
Diophantine & DP	56	DP and number-theoretic constructions
Graph Theory	50	Graph and tree algorithms
Constructive/DP	45	Constructive algorithms and DP
Simulation / Algebra	45	Stateful simulation, algebraic transforms
Comp. Geometry	43	Planar and spatial geometry

theory and arithmetic, computational geometry, graph and tree algorithms, dynamic programming, and constructive or simulation-style problems. The boundaries are soft, since many problems combine ideas (for example, geometry solved with dynamic programming, or number theory solved by construction), which is itself consistent with the breadth that makes the archive challenging. The largest group, basic number theory and arithmetic, reflects the many introductory problems in the early part of the archive, while geometry and graph problems form distinct, well-separated regions. We treat this taxonomy as an exploratory map rather than a strict labeling; the per-problem assignments are released alongside the dataset.

5 Future Work

SGU-Editorial is an ongoing effort. We plan to extend this work in several directions:

Beyond SGU: With the SGU archive now essentially complete (452 of 453 problems), a natural next step is to apply the same editorial-generation methodology to other historical problem sources that are similarly underrepresented in existing training data.

Finetuning Analysis: We intend to conduct experiments finetuning LLMs on the SGU-Editorial dataset to measure whether exposure to these problems and editorials improves model

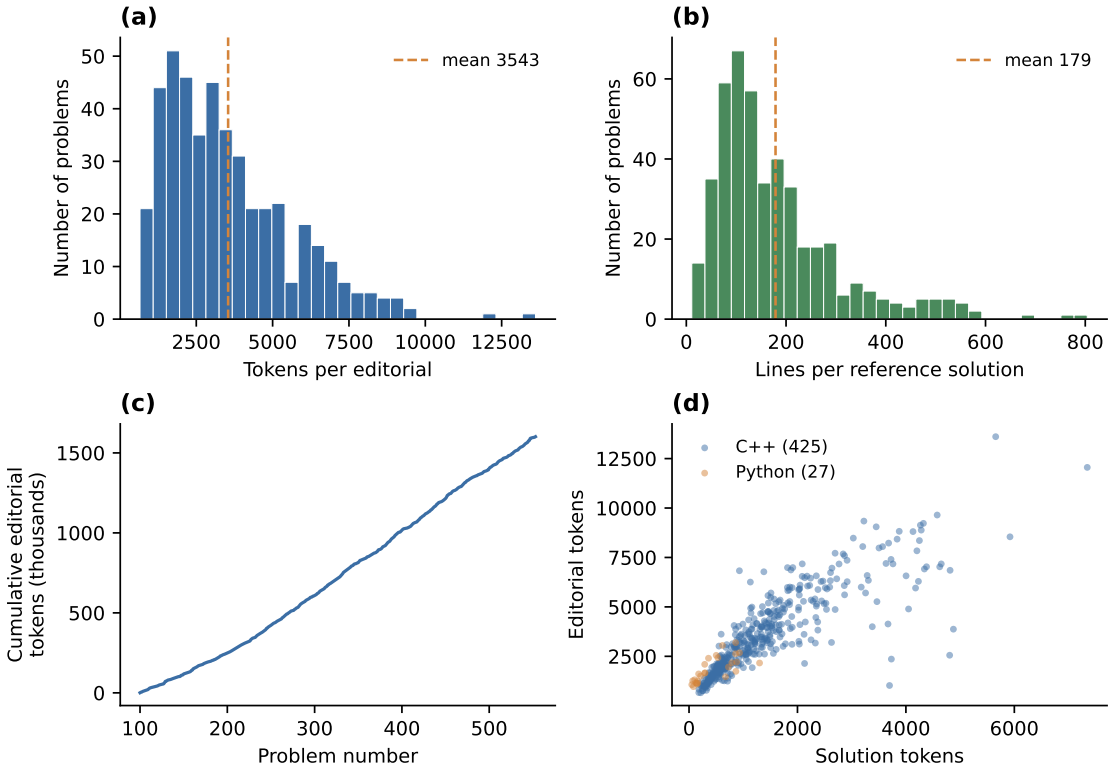

Figure 2: Dataset statistics: (a) editorial length (tokens), (b) solution length (lines), (c) cumulative editorial tokens by problem number, and (d) editorial vs. solution length, colored by language.

performance on SGU and similar challenging problem sets. Some preliminary experiments with small open source models suggest that this might be the case.

Evaluation Benchmark: We plan to develop SGU-Editorial into a proper evaluation benchmark by generating comprehensive test data for each problem. This would enable standardized evaluation of code generation models on problems that current state-of-the-art systems find difficult.

Refining the Taxonomy: The thematic clustering of Section 4 is a first pass. Finer-grained or hierarchical groupings, and a per-problem difficulty estimate (e.g. from Codeforces ratings), would make the archive easier to navigate by topic and difficulty and support curriculum-style study and targeted finetuning.

6 Conclusion

We presented SGU-Editorial, a dataset containing 452 of the 453 competitive programming problems from the SGU Online Judge, enhanced with detailed LLM-generated editorials. The only omission, problem 145, is currently broken on the judge and has no passing solutions. SGU problems present a challenge for current state-of-the-art LLMs, likely due to the diverse algorithmic ideas underrepresented in existing training data. Each problem in our dataset includes a structured editorial with algorithm explanations, implementation guides, and reference solutions in multiple languages.

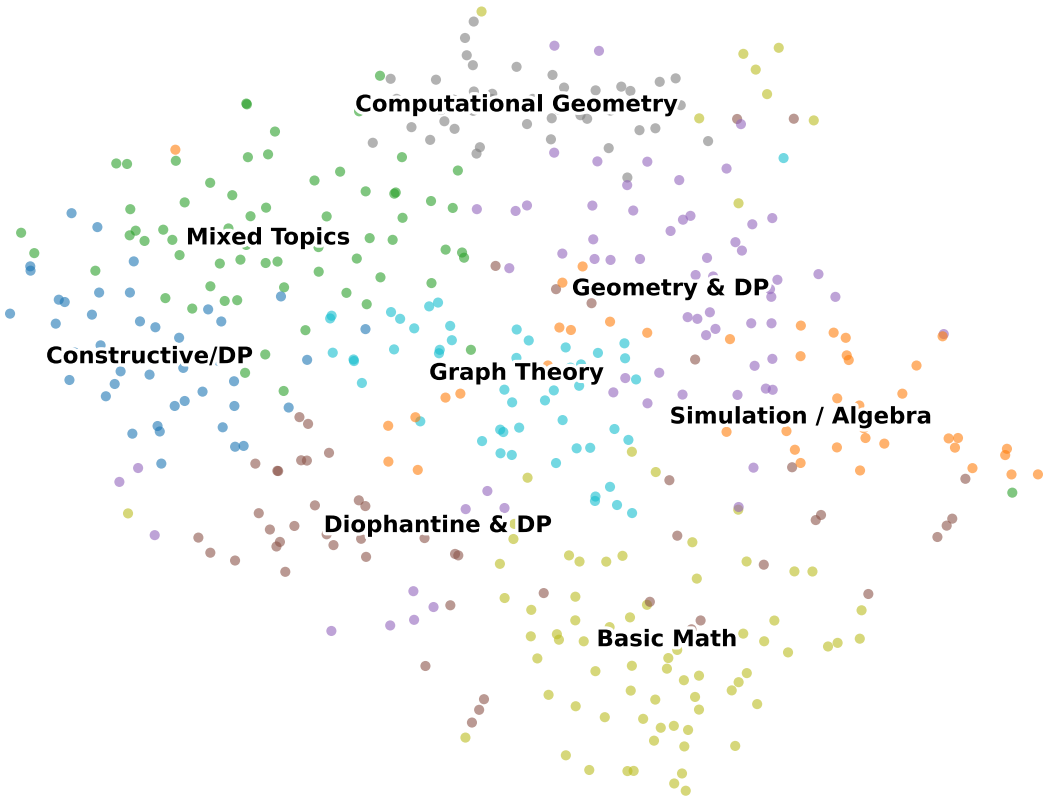

Figure 3: t-SNE projection of the 452 editorial embeddings, colored by k-means cluster and annotated with each cluster's tag (Table 2).

SGU-Editorial has certain limitations, including the provision of a single reference solution per problem and the potential for subtle inaccuracies in LLM-generated editorials.

Data Availability

The SGU-Editorial project page, with documentation and downloads, is at `https://radoslav11.com/sgu-dataset/`. The dataset is continuously updated in the repository at `https://github.com/radoslav11/acm-sgu`. We also provide versioned zip snapshots:

- **v1** (initial release, 250 problems): `https://radoslav11.com/sgu-dataset/sgu-editorial-dataset-v1.zip`
- **v2** (near-complete, 452 of 453 problems): `https://radoslav11.com/sgu-dataset/sgu-editorial-dataset-v2.zip`

Each snapshot bundles the `problems/` and `dataset/` directories at that version. The original problems are hosted in the "acmsguru" section of Codeforces [4].

Acknowledgments

We thank Saratov State University for the original problem archive, OpenAI for access to the reasoning models and the Codex coding agent used in editorial and solution generation, and Anthropic for Claude Code.

References

[1] Google DeepMind AlphaCode Team. Alphacode 2 technical report. 2023.

[2] Kuldeep Gautam, S VenkataKeerthy, and Ramakrishna Upadrasta. Cofo: Codeforces dataset for program classification, recognition and tagging. *arXiv preprint arXiv:2503.18251*, 2025.

[3] Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, et al. Measuring coding challenge competence with apps. *arXiv preprint arXiv:2105.09938*, 2021.

[4] Vitaliy Kudasov. Codeforces: acm.sgu.ru comes back. `https://codeforces.com/blog/entry/59070`, 2018. Codeforces Blog.

[5] Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. Deduplicating training data makes language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 8424–8445, 2022.

[6] Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022.

[7] Stuart Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[8] Lili Mou, Ge Li, Lu Zhang, Tao Wang, and Zhi Jin. Convolutional neural networks over tree structures for programming language processing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016.

[9] Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, et al. Text and code embeddings by contrastive pre-training. *arXiv preprint arXiv:2201.10005*, 2022.

[10] Guilherme Penedo, Anton Lozhkov, Hynek Kydlíček, Loubna Ben Allal, Edward Beeching, Agustín Piqueres Lajarín, Quentin Gallouédec, Nathan Habib, Lewis Tunstall, and Leandro von Werra. Codeforces cots. `https://huggingface.co/datasets/open-r1/codeforces-cots`, 2025.

[11] Peter J Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.

[12] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1715–1725, 2016.

[13] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

A Analysis Methods

This appendix details the auxiliary analyses of Section 4. The corresponding scripts and their outputs are released with the dataset under `src/` and `analysis/`.

Tokenization. All token counts in this paper are computed with OpenAI's `tiktoken` library using the `cl100k_base` encoding, a byte-pair encoding (BPE) [12] with a vocabulary of about 100k subwords that underlies the GPT-3.5 and GPT-4 model families. BPE starts from bytes and greedily merges the most frequent adjacent pairs, so the same tokenizer handles prose, source code, and Unicode uniformly. We tokenize each file as raw UTF-8 text without any preprocessing, so the reported counts include code, markup, and whitespace exactly as stored. Because different model families use different vocabularies, these counts are intended as a consistent relative measure across the dataset rather than an exact match to any single model's billing.

Embedding and clustering. For the thematic analysis we embed the first 8,000 characters of each editorial (which contain the abridged statement and the core of the discussion) with OpenAI's `text-embedding-3-large` model [9], producing a 3072-dimensional vector per problem. We L_2-normalize the vectors so that Euclidean operations correspond to cosine similarity, then cluster with k-means [7]. We select $k = 8$; over $k \in \{8, \ldots, 27\}$ the silhouette coefficient [11] is uniformly low (≈ 0.03) and nearly flat, which is expected for dense text embeddings and indicates soft rather than sharply separated clusters. A chat model assigns each cluster a short tag and a one-line theme from a sample of eight member editorials. For visualization only, we project the embeddings to two dimensions with t-SNE [13] (perplexity 30, PCA initialization). We treat the resulting taxonomy as an exploratory map; the per-problem assignments are released with the dataset and listed in full in Appendix C.

B Semantic Redundancy and Diversity

Near-duplicate content is a known problem for LLM corpora, where it wastes capacity and contaminates evaluation [5]; conversely, a useful dataset should span many distinct ideas. We check both directions using the editorial embeddings of Appendix A (cosine similarity throughout); Figure 4 summarizes the results.

Redundancy. Pairwise similarities are concentrated around a mean of 0.48, and only 0.007% of the roughly 102k problem pairs exceed 0.8. The most similar pairs (Table 3) are not duplicates but deliberate variants or problems that share a technique: the closest pair, *Little Bishops* and *Big Bishops* (0.92), belongs to a family of chess-piece placement problems, and the two max-flow construction problems *Flow construction* and *Reactor Cooling* sit at 0.85. Notably, the eight pairs of problems that share a title (e.g. two problems each named *Traffic Lights*, *Coins*, or *Elevator*) do *not* appear among the closest pairs, confirming that they are genuinely different problems rather than re-uploads.

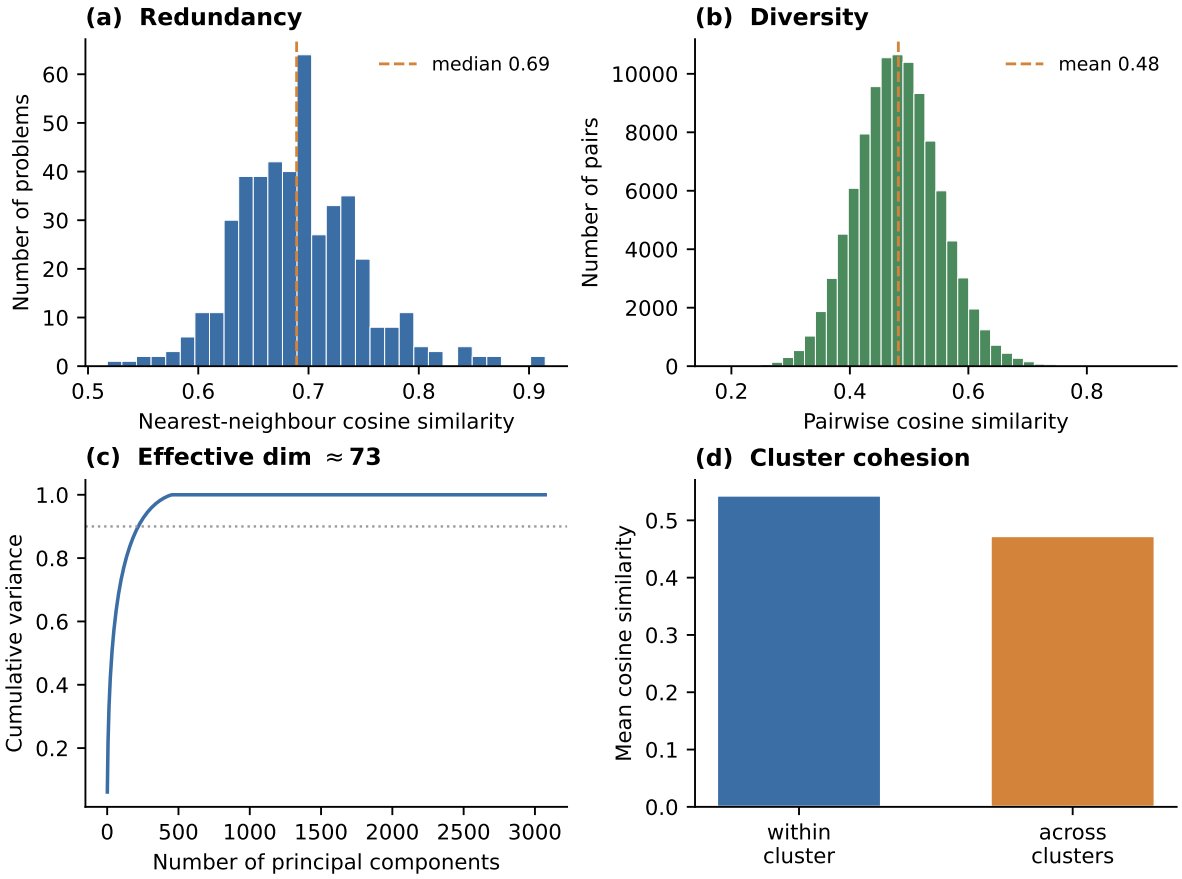

Figure 4: Embedding-based redundancy and diversity: (a) each problem's nearest-neighbour similarity; (b) all pairwise similarities; (c) cumulative variance of the embedding covariance (the dotted line marks 90%); and (d) mean within- vs across-cluster similarity.

Diversity. The embedding cloud has an effective dimensionality (participation ratio of its covariance spectrum) of about 73, despite all problems coming from a single judge, indicating that the editorials cover many largely independent directions rather than rephrasing a few templates. As a sanity check on the clusters of Appendix C, mean within-cluster similarity (0.54) exceeds mean across-cluster similarity (0.47); the gap is real but modest, consistent with the soft cluster structure noted above.

C Cluster Assignments

For completeness, the following lists every problem in each thematic cluster of Table 2 (clusters ordered by size, problems by number).

Basic Math (79): p100, p102, p105, p107, p113, p114, p115, p116, p117, p118, p119, p123, p126, p127, p130, p133, p137, p140, p144, p146, p148, p152, p154, p160, p163, p169, p171, p178, p180, p181, p184, p186, p193, p199, p203, p207, p231, p246, p254, p259, p275, p276, p296, p297, p299, p316, p347, p349, p353, p355, p365, p374, p375, p398, p403, p404, p405, p407, p415, p417, p443, p444, p455, p456, p460, p467, p473, p481, p486, p491, p492, p495, p497, p499, p524, p531, p533, p549, p551

Mixed Topics (67): p101, p111, p132, p147, p159, p161, p164, p187, p188, p205, p229, p235, p237, p245, p247, p256, p263, p268, p271, p273, p281, p287, p295, p298, p302, p306, p311, p317, p318, p319, p329, p330, p334, p339,

Table 3: The ten most similar editorial pairs by cosine similarity. The closest pairs are problem variants or shared-technique problems, not duplicates.

Problem A		Problem B		Cos.
p220	Little Bishops	p221	Big Bishops	0.91
p222	Little Rooks	p269	Rooks	0.87
p176	Flow construction	p194	Reactor Cooling	0.85
p122	The book	p156	Strange Graph	0.85
p233	The Greatest Angle	p332	Largest Circle	0.84
p233	The Greatest Angle	p440	Moles and Holes	0.82
p303	Great Berland Wall	p315	The Highway Belt	0.81
p247	Difficult Choice	p342	Reihenfolge	0.80
p251	Polymania	p430	Unit-distance graph	0.80
p224	Little Queens	p225	Little Knights	0.79

p342, p357, p360, p362, p366, p371, p372, p377, p382, p389, p393, p395, p399, p410, p411, p418, p419, p436, p447, p450, p453, p458, p475, p479, p480, p494, p504, p510, p516, p535, p540, p542, p547

Geometry & DP (67): p136, p158, p162, p165, p167, p204, p214, p240, p248, p250, p257, p265, p285, p300, p301, p308, p309, p313, p331, p335, p338, p343, p346, p363, p364, p367, p368, p376, p378, p380, p386, p390, p391, p394, p396, p400, p401, p408, p422, p423, p427, p429, p430, p433, p434, p445, p446, p452, p463, p464, p465, p466, p470, p471, p472, p477, p478, p482, p493, p501, p509, p521, p526, p532, p543, p545, p553

Diophantine & DP (56): p104, p106, p125, p135, p141, p142, p155, p168, p175, p177, p179, p190, p191, p196, p201, p202, p220, p221, p222, p223, p230, p238, p241, p249, p258, p262, p269, p274, p291, p292, p307, p310, p320, p328, p344, p350, p358, p361, p369, p379, p397, p406, p409, p425, p428, p437, p484, p488, p490, p496, p506, p517, p519, p523, p536, p538

Graph Theory (50): p103, p121, p122, p134, p143, p149, p156, p172, p174, p176, p185, p194, p195, p206, p212, p213, p216, p218, p226, p236, p242, p252, p272, p280, p286, p314, p321, p322, p323, p326, p352, p381, p384, p402, p413, p424, p438, p457, p462, p487, p503, p507, p513, p515, p518, p520, p525, p529, p534, p541

Constructive/DP (45): p108, p109, p112, p131, p138, p139, p153, p157, p170, p183, p197, p200, p210, p224, p225, p232, p239, p255, p261, p282, p288, p294, p304, p337, p354, p356, p370, p385, p416, p441, p448, p468, p476, p483, p485, p489, p498, p502, p508, p527, p537, p544, p546, p548, p552

Simulation / Algebra (45): p166, p173, p182, p189, p208, p211, p215, p219, p234, p243, p260, p264, p270, p279, p284, p289, p293, p305, p324, p325, p327, p336, p340, p341, p348, p359, p388, p392, p420, p421, p426, p431, p432, p439, p442, p449, p451, p454, p459, p461, p505, p511, p528, p530, p539

Computational Geometry (43): p110, p120, p124, p128, p129, p150, p151, p192, p198, p209, p217, p227, p228, p233, p244, p251, p253, p266, p267, p277, p278, p283, p290, p303, p312, p315, p332, p333, p345, p351, p373, p383, p387, p412, p414, p435, p440, p469, p474, p500, p512, p514, p522

D Accepted-Solution Counts

The acmsguru index reports, for each problem, the number of users with at least one accepted solution (as of 2026-06-08). We treat this as an engagement signal rather than a clean difficulty measure: it reflects difficulty in part, but it is also shaped by how many people *attempt* a problem, which is not the same thing. Counts span four orders of magnitude, from 4 (e.g. several problems in the 440–452 range) to 13,646 for problem 100 ($A+B$); the median is 29. The single problem with no accepted solution, number 145, is the broken problem excluded from the dataset, which is consistent with the failure described in Section 1.

The solver count correlates with the textual size of a problem's material (Figure 5): -0.75 with editorial length in tokens, -0.63 with reference-solution length in lines, and -0.45 with

problem-statement length in tokens. Longer editorials and solutions plausibly indicate genuinely harder problems, so part of this is a difficulty effect. The statement-length correlation, however, points at a confound: shorter, simpler-looking statements attract more attempts regardless of the underlying difficulty, so some of the signal is approachability rather than hardness. A second confound is position in the archive: solver count correlates -0.36 with problem number, consistent with users working through the set roughly in order and dropping off, so earlier problems accumulate more solvers independently of how hard they are.

For these reasons we do not present the count as a difficulty score. As a coarse popularity signal it is still informative: aggregated over the clusters of Table 2, the elementary number-theory and arithmetic cluster is by far the most-solved (median 156), while the geometry, geometry-with-DP, and simulation clusters have the fewest solvers (median $\approx$11) — a mix of those problems being both harder and less approachable than the introductory arithmetic ones.

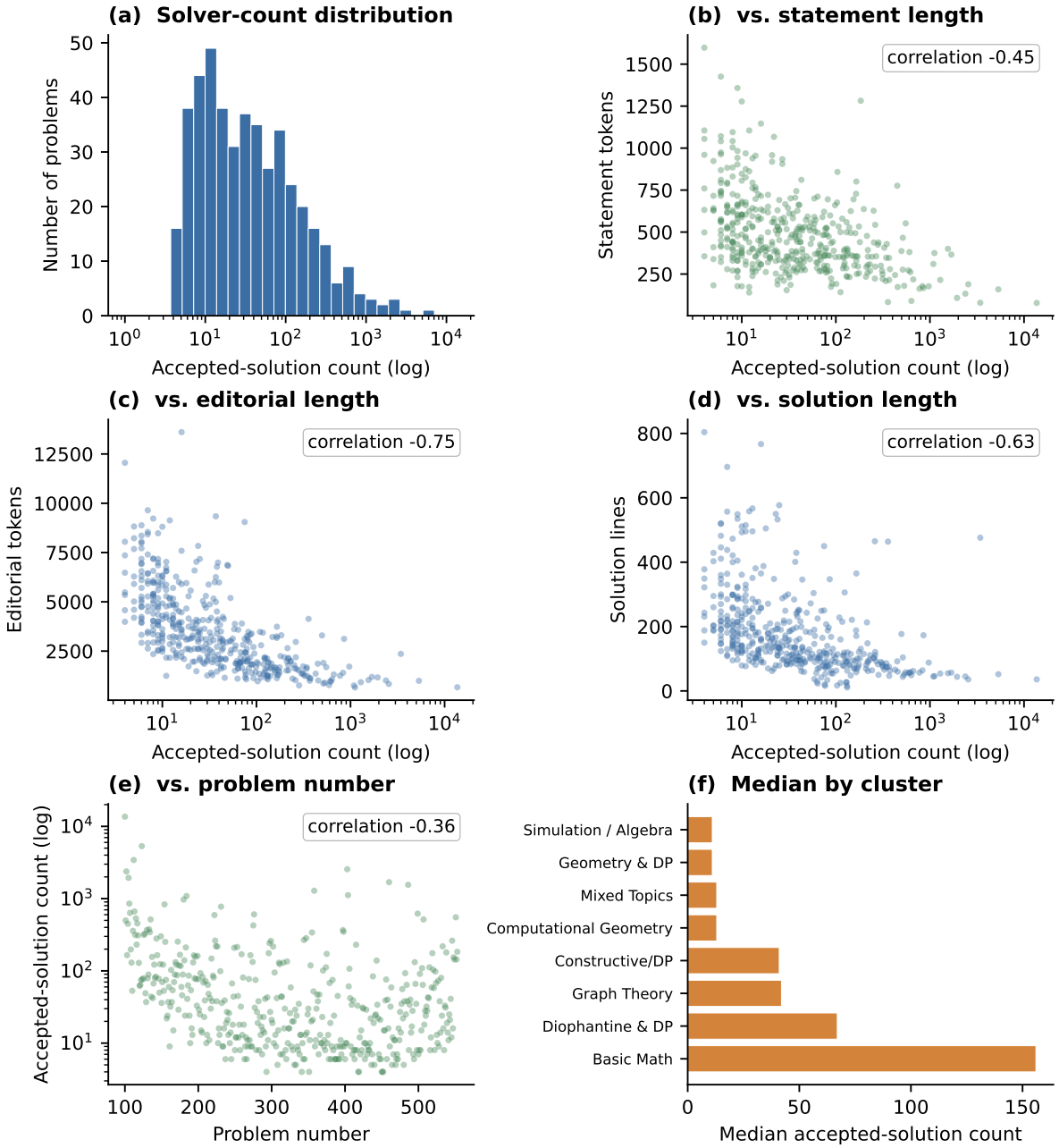

Figure 5: Accepted-solution (solver) counts: (a) distribution (log scale); (b–d) solver count against statement, editorial, and reference-solution length, each annotated with its correlation; (e) solver count vs. problem number; and (f) median solver count per thematic cluster. Counts mix difficulty with attempt rate, so shorter bars indicate fewer solvers, not necessarily harder problems.