TOPOMATH: A CONCEPT GRAPH TOPOLOGY-GUIDED APPROACH FOR LLM MATHEMATICAL REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Although large language models (LLMs) have shown immense potential for mathematical problem solving, their intermediate derivations frequently suffer from premise drift, computational errors, and exploratory "dead ends". A fundamental limitation is that models navigate an entirely implicit conceptual space without structural guidance on mathematical concepts and their interrelations. In this paper, we propose TOPOMATH, an inference-time framework that establishes a relationship graph of mathematical concepts to guide and verify LLM reasoning. Specifically, TOPOMATH builds a directed acyclic graph (DAG) from Wikidata's mathematical taxonomy and projects each problem onto a compact scaffold containing grounded anchors, taxonomic paths, shared ancestors, and local neighbors. At inference time, the scaffold can serve as a global navigational prior for solution generation, while a dual-verification mechanism combining SymPy's symbolic execution with topological concept-membership checks detects computational flaws and conceptual drifts to trigger targeted revision. Evaluated on six benchmarks spanning standard to Olympiad-level mathematics across three backbone models, TOPOMATH consistently outperforms direct inference and state-of-the-art inference-time baselines, exhibiting significant improvements on PASS@1 and perfect reasoning rates (PRRs). Process-level graph analysis further reveals that TOPOMATH eliminates 81.2% of irrelevant reasoning nodes and shortens proof depth by 34.5%, indicating that topological concept priors effectively foster concise, robust, and structurally closed mathematical reasoning.

1 INTRODUCTION

Large language models (LLMs) have achieved notable progress in mathematical problem solving through chain-of-thought (CoT) prompting, program execution, and self-consistency voting (Wei et al., 2022; Wang et al., 2023; Gao et al., 2023). Recent approaches also enhance LLM mathematical reasoning by retrieving knowledge to support problem solving (Das et al., 2024) and analyzing premise dependencies to identify errors in generated solutions (Mukherjee et al., 2025). Despite such advances, process-level evaluations reveal that generated solutions can contain invalid or irrelevant intermediate steps, and that correct final answers do not necessarily imply sound derivations (Xia et al., 2025; Yuan et al., 2026).

These findings motivate examining intermediate results and the evidence to support them to further improve reasoning. We observe that a primary cause of these pathologies is that models navigate an entirely implicit conceptual space. As such, we aim to resolve them by explicitly organizing the mathematical concepts to guide solution generation. Our hypothesis is that connecting concepts in a problem to potentially useful mathematical knowledge can provide a structural prior for multi-step reasoning, complementing answer aggregation, external computation, and post-generation verification. For example, consider the problem statement "a triangle ABC inscribed in a circle with diameter AB." Although the statement specifies only elementary geometric entities, identifying an optimal solution trajectory requires activating Thales's theorem, an implicit lemma connecting diameter conditions to right-angle properties. Without structural guidance, an LLM may easily overlook this bridging concept, instead pursuing convoluted coordinate or trigonometric calculations that rapidly derail. Exposing the relational path between stated conditions and necessary intermediate lemmas fundamentally constrains the reasoning space, transforming unguided exploration into goal-directed deduction.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

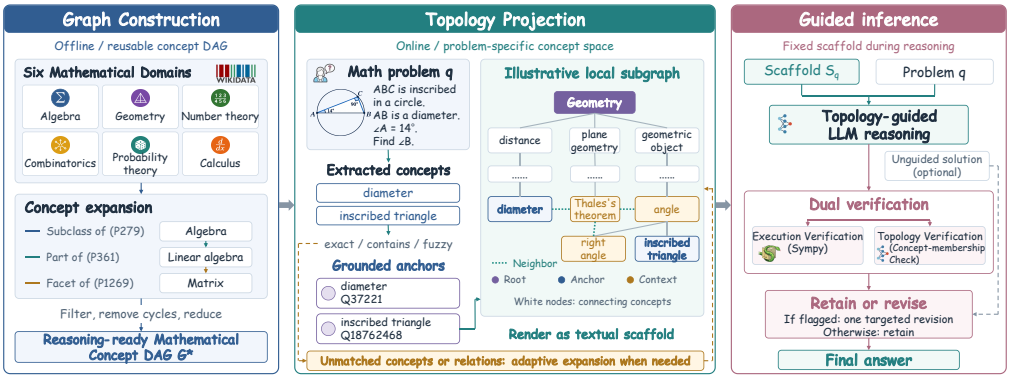

Figure 1: The overall framework of TOPOMATH.

Based on such observations, we explicitly represent structural guidance through the topology of mathematical concepts, understood from a graph-theoretic perspective as a directed acyclic graph (DAG) whose nodes represent mathematical concepts and edges encode taxonomic, compositional, and associative relations. Distinct from unstructured text retrieval, this parameter-free prior serves two complementary roles during inference: first, it provides a problem-specific scaffold that exposes potentially useful concepts before generation; and second, it serves as a verification boundary that cross-examines invoked concepts and symbolic claims to trigger targeted revision upon errors.

To implement the above ideas, we propose TOPOMATH, an inference-time framework that operationalizes concept topology across an offline-to-online pipeline, as shown in Figure 1. Specifically, TOPOMATH constructs a DAG of mathematical concepts from Wikidata, dynamically projects each problem onto a compact local scaffold to guide solution generation, and cross-checks the resulting derivation through execution-based and concept-membership dual-verification for targeted revision.

The main contributions of this work are summarized as follows:

- We formalize the topology of mathematical concepts as an explicit structural prior for LLM reasoning, introducing an offline-to-online projection mechanism that converts taxonomic relations into an actionable reasoning scaffold.
- We develop a verification-and-revision mechanism that couples deterministic symbolic execution via SymPy with topological concept-membership checks, detecting computational flaws and conceptual drift without costly neural verifiers.
- We evaluate the performance of TOPOMATH across six benchmarks spanning standard to Olympiad-level mathematics and three backbone models, demonstrating that TOPOMATH achieves consistent accuracy gains while largely eliminating irrelevant reasoning nodes and shortening proof depth.

2 RELATED WORK

LLM Mathematical Reasoning Enhancing the mathematical reasoning capability of LLMs has been actively investigated in recent years. Existing approaches can be broadly categorized into three paradigms: pre-training, post-training, and inference-time enhancement. Pre-training methods continually train LLMs on large-scale mathematical corpora. Representative methods, such as Llemma (Azerbayev et al., 2024), DeepSeekMath (Shao et al., 2024), and InternLM-Math (Ying et al., 2024), leverage textbooks, scientific documents, formal proofs, and Web resources to improve the mathematical ability of general-purpose LLMs. Building upon pretrained models, post-training approaches, e.g., WizardMath (Luo et al., 2025), MetaMath (Yu et al., 2024), Step-DPO (Lai et al., 2024), THOR (Chang et al., 2026), and rStar-Math (Guan et al., 2025), improve mathematical reasoning through supervised fine-tuning, preference optimization, or reinforcement learning. Although both types of methods achieve remarkable performance gains, they require additional training data, specialized optimization, and considerable computational resources.

Inference-time methods guide solution generation through prompting, sampling, and tool use. Chain-of-thought and least-to-most prompting structures intermediate reasoning and problem decomposition (Wei et al., 2022; Kojima et al., 2022; Zhou et al., 2023); Self-consistency aggregates answers from sampled trajectories (Wang et al., 2023); PAL and program-of-thought delegate computation to generated programs (Gao et al., 2023; Chen et al., 2023); SBSC generates successive subtask programs using earlier execution outputs (Singh et al., 2025); CodeSteer combines separately generated code-based and textual solutions (Chen et al., 2025). MATHSENSEI integrates mathematical knowledge retrieval, Python execution, WolframAlpha queries, and execution-error-driven code refinement within a modular inference pipeline (Das et al., 2024).

Conceptual and Structure-Guided Reasoning Recent work examines how mathematical concepts and explicit representations support reasoning. CounterMATH evaluates conceptual reasoning through university-level statements and example-based rationales and improves this ability through supervised fine-tuning on curated proofs (Li et al., 2025). CoMAT translates a problem into symbolic definitions, relations, facts, and a formalized question before prompting the same model to derive an answer (Leang et al., 2025). Structural approaches organize both external knowledge and intermediate computation. RAG and GraphRAG (Lewis et al., 2020; Edge et al., 2024) supply external context through passage retrieval and graph-based organization of document collections, respectively. Tree-of-thoughts, graph-of-thoughts, and adaptive graph-of-thoughts organize intermediate reasoning states for search and computation (Yao et al., 2023; Besta et al., 2024; Pandey et al., 2025). Inspired by the above methods, TOPOMATH constructs a DAG from Wikidata (Vrandečić & Krötzsch, 2014) and retrieves problem-specific scaffolds to provide contextual guidance for generation and references for checking.

Reasoning Verification and Evaluation Feedback and verification provide signals for refining or selecting reasoning trajectories. Self-Refine and Reflexion use feedback to revise outputs or guide subsequent attempts (Madaan et al., 2023; Shinn et al., 2023). Process supervision trains reward models to assess intermediate steps (Lightman et al., 2024), while SAFE combines formal verification outcomes with learned aggregation and process-reward scores for candidate ranking (Liu et al., 2025). PARC augments generated reasoning steps with explicit premise links and uses the resulting dependency graph to localize mathematical, logical, and propagated errors (Mukherjee et al., 2025). Rubric Rewards use problem-specific criteria to score complete derivations and guide reinforcement learning toward reducing correct-answer solutions with flawed reasoning (Yuan et al., 2026). Evaluation frameworks examine mathematical reasoning beyond standard answer accuracy. MATHCHECK combines problem solving, answerability judgment, outcome judgment, and process judgment with controlled variations of seed problems (Zhou et al., 2025). DAG-Math represents derivations as directed graphs and introduces a logical-closeness-based evaluation of reasoning trajectories (Zhang et al., 2026). TOPOMATH combines execution checks on explicit computational claims with scaffold-based concept-membership checks to support targeted revision. Its evaluation builds on DAG-Math to assess answer correctness together with structural closure.

3 METHODOLOGY

TOPOMATH is an inference-time framework that introduces an explicit mathematical concept space as a structural prior for reasoning. As illustrated in Figure 1, TOPOMATH consists of three stages. Stage I constructs a global mathematical concept DAG $\mathcal{G}^*$. Stage II projects each problem q onto a compact problem-specific subgraph G_q and renders it as a textual scaffold S_q. Stage III conditions the backbone model on this scaffold and applies execution-based and concept-membership verification to diagnose unreliable reasoning and support targeted revision. The central idea is to organize inference around a compact set of problem-relevant mathematical concepts rather than leaving the conceptual search space entirely implicit.

3.1 MATHEMATICAL CONCEPT GRAPH CONSTRUCTION

TOPOMATH first constructs a mathematical concept graph from Wikidata. Starting from six broad mathematical domains, we retrieve concepts and their hierarchical, compositional, and associative relations to obtain a field-level backbone $\mathcal{G}_{\text{field}} = (\mathcal{V}_{\text{field}}, \mathcal{E}_{\text{field}})$. We further augment this backbone

with additional concept relations and a small number of manually reviewed solution-level connections, yielding an initial graph $\mathcal{G}_0 = (\mathcal{V}_0, \mathcal{E}_0)$. Because $\mathcal{G}_0$ may contain redundant relations, cycles, or concepts disconnected from the predefined mathematical domains, we apply topology governance to reduce $\mathcal{G}_0$ to $\mathcal{G}^* = (\mathcal{V}, \mathcal{E})$. The resultant $\mathcal{G}^*$ is a directed acyclic graph (DAG) in which every retained concept is reachable from at least one root domain. It is constructed offline and remains unchanged during inference, serving as the global structural prior for Stage II. Details about relation extraction, graph governance, and topology statistics are provided in Appendix B.1.

3.2 Problem-Specific Topology Projection

The global topology contains substantially more concepts than are needed for solving an individual problem. Stage II therefore projects each problem onto a compact local concept subgraph. Let $\widetilde{\mathcal{G}}^{(t)}$ denote the working topology available when processing the current problem. For a fresh scaffold-generation run, $\widetilde{\mathcal{G}}^{(0)} = \mathcal{G}^*$, while accepted concept expansions may persist in the working topology across subsequent problems. The overall projection is formulated as $\Psi : (\widetilde{\mathcal{G}}^{(t)}, q) \to (G_q, S_q)$.

Concept Grounding and Adaptive Expansion TopoMath first extracts mathematical concept phrases from q and grounds them to the current working topology, producing grounded anchors A_q and unmatched concepts U_q, i.e., $(A_q, U_q) = \Gamma(q, \widetilde{\mathcal{G}}^{(t)})$. When an extracted concept cannot be grounded, TopoMath retrieves candidate concepts and their connections from Wikidata, expanding both the concept coverage and relational structure of the working topology. Predefined domain-bridging rules provide fallback connections when no suitable Wikidata path is found. Let (V_q^+, E_q^+) denote the accepted graph additions. The working topology is updated incrementally as

$$\widetilde{\mathcal{G}}^{(t+1)} = \widetilde{\mathcal{G}}^{(t)} \cup (V_q^+, E_q^+). \tag{1}$$

Accepted expansions remain available for subsequent problems, while the original offline DAG $\mathcal{G}^*$ is not modified.

Local Topology Induction and Scaffold Construction From the final anchor set A_q, TopoMath retrieves selected root-to-anchor paths Π_q, shared ancestors L_q, and a local concept neighborhood N_q. We denote the resulting compact representation by $\mathcal{T}_q = (A_q, \Pi_q, L_q, N_q)$, with a problem-specific node set $V_q = A_q \cup \mathrm{Nodes}(\Pi_q) \cup L_q \cup N_q$. The corresponding local topology is conceptually defined as $G_q = \widetilde{\mathcal{G}}^{(t+1)}[V_q]$. Here, L_q denotes shared ancestors selected according to stored topology depth rather than a general LCA operation on arbitrary DAGs. In implementation, the local structure is represented compactly through selected paths and neighborhood records rather than by serializing the complete typed induced subgraph. Finally, the structural representation is rendered as $S_q = \mathrm{Render}(\mathcal{T}_q)$. The scaffold summarizes grounded anchors, selected domain paths, shared ancestors, local neighborhood concepts, and available concept descriptions, and is supplied to the language model as an inference-time orientation prior.

Detailed grounding, adaptive expansion, topology induction, and rendering procedures are provided in Appendix B.2.

3.3 Topology-Guided Reasoning and Dual Verification

Given q and its scaffold S_q, the backbone model generates a topology-guided trajectory $r_q^s = \mathcal{M}_\theta(q, S_q)$. The scaffold acts as an orientation prior rather than a prescribed reasoning path: the model remains guided by the original problem statement and may ignore irrelevant scaffold information. The same backbone may additionally produce an unguided trajectory $r_q^0 = \mathcal{M}_\theta(q)$, which is used only as auxiliary diagnostic context and does not participate in answer voting or direct final-answer selection.

For execution verification, TopoMath extracts explicit machine-verifiable claims $X_q = \mathcal{C}(r_q^s)$. For topology verification, it extracts the mathematical concepts involved in the reasoning as $Z_q = \mathcal{Z}(r_q^s)$. The two verification signals are then computed as

$$e_q^{\mathrm{exec}} = \mathcal{V}_{\mathrm{exec}}(q, X_q), \qquad e_q^{\mathrm{topo}} = \mathcal{V}_{\mathrm{topo}}(Z_q, G_q). \tag{2}$$

Execution-based verification checks explicit symbolic and computational claims rather than attempting to verify the complete proof. Topology-based verification checks whether concepts invoked during reasoning are supported by the problem-specific scaffold.

TOPOMATH may compare the final answers of the guided and unguided trajectories, producing the auxiliary disagreement signal $d_q = \mathbb{I}[\neg\,\mathrm{Equivalent}(a(r_q^s), a(r_q^0))]$. This signal is used only as diagnostic evidence. Revision is triggered according to $\tau_q = \mathbb{I}[\mathrm{Fail}(e_q^{\mathrm{exec}}) \vee \mathrm{Fail}(e_q^{\mathrm{topo}})]$, and the diagnostic context is summarized by $\xi_q = (q, G_q, S_q, e_q^{\mathrm{exec}}, e_q^{\mathrm{topo}}, d_q, r_q^0)$. When verification identifies an unreliable component, TOPOMATH may apply the targeted revision operator $\mathcal{R}_\theta$ to obtain $\widetilde{r}_q = \mathcal{R}_\theta(r_q^s, \xi_q)$; otherwise, $\widetilde{r}_q = r_q^s$. The implementation applies at most one revision and retains the scaffold-guided trajectory as the primary solution. The final prediction is $\widehat{a}_q = a(\widetilde{r}_q)$.

Further details of executable-claim extraction, verification, revision, and answer equivalence are provided in Appendix B.3.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Datasets We evaluate TOPOMATH on six widely used mathematical reasoning benchmarks, including MATH-500 (Hendrycks et al., 2021), Omni-MATH (Gao et al., 2025), and AIME 2025, AIME 2026, BRUMO 2025, and HMMT 2026 from MathArena (Balunovic et al., 2025). These datasets cover standard mathematical reasoning, large-scale Olympiad-style problems, and high-difficulty competitions, providing a comprehensive evaluation of reasoning accuracy and robustness across different difficulty levels. For Omni-MATH, we evaluate on a fixed stratified subset of 500 problems sampled with a fixed random seed according to the joint distribution of mathematical domains and difficulty levels, denoted as Omni-500. More details are provided in Appendix C.

Metrics We evaluate TOPOMATH in terms of final-answer correctness and reasoning-process quality. **PASS@1** is the ratio of correctly answered problems, measuring final-answer accuracy. For process-level evaluation, we build on the logical-closeness framework of DAG-MATH (Zhang et al., 2026). In DAG-MATH, each trajectory r_i is parsed into a directed graph whose nodes represent reasoning steps and edges encode their dependencies. Let V_i^- denote all nodes except the final-answer node. Structural closure is defined as

$$\mathrm{Closure}(r_i) = \frac{\left|\left\{v \in V_i^- \mid \deg^+(v) > 0\right\}\right|}{|V_i^-|}, \tag{3}$$

where $\deg^+(v)$ denotes node outdegree. Full closure means that every non-final node is referenced by a later step. **Perfect Reasoning Rate (PRR)** measures the proportion of trajectories with both a correct final answer and full structural closure:

$$\mathrm{PRR} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}\left[\mathrm{correct}_i \wedge \mathrm{Closure}(r_i) = 1\right], \tag{4}$$

where N is the number of evaluated trajectory records and $\mathrm{correct}_i$ indicates final-answer correctness. **Closure-AUC** summarizes answer correctness under varying closure requirements. Let $A(\tau)$ denote the proportion of all evaluated trajectories that are answer-correct and satisfy $\mathrm{Closure}(r_i) \geq \tau$. We compute

$$\text{Closure-AUC} = \frac{1}{101}\sum_{k=0}^{100} A\left(\frac{k}{100}\right), \tag{5}$$

using thresholds from 0 to 1 in increments of 0.01, with $A(0) = \mathrm{PASS@1}$ and $A(1) = \mathrm{PRR}$. We additionally report **Mean Closure**, defined as $\mathrm{Mean\ Closure} = N^{-1}\sum_{i=1}^{N} \mathrm{Closure}(r_i)$, which measures average structural closure independently of final-answer correctness. Detailed evaluation procedures and dependency-annotation prompts are provided in Appendix D.

Baselines We evaluate TOPOMATH using the following three backbone models: DeepSeekV4-Flash (DeepSeek-AI, 2026), Qwen3.5-27B (Qwen Team, 2026), and Qwen3-8B (Qwen Team,

2025). By default, we use DeepSeekV4-Flash as the common backbone and compare TOPOMATH with seven inference-time baselines: CoT-ZeroShot (Kojima et al., 2022), Self-Consistency@5 (Wang et al., 2023), PAL (Gao et al., 2023), Least-to-Most (Zhou et al., 2023), Code + Text + Sum. (Chen et al., 2025), SBSC (Singh et al., 2025), and CoMAT (Leang et al., 2025). To assess cross-backbone generalization, we additionally compare direct inference with TOPOMATH on Qwen3.5-27B and Qwen3-8B across all six benchmarks. Detailed baseline descriptions and complete experimental results are provided in Appendix E.

Implementation Details We set the decoding temperature to 0.7 and the maximum solution-generation length to 4,096 tokens. Thinking mode is disabled for both DeepSeekV4-Flash and Qwen3.5-27B. For the four small competition benchmarks—AIME 2025, AIME 2026, BRUMO 2025, and HMMT 2026—we average results over ten runs with independent generations to reduce dependence on individual stochastic outputs. These benchmarks contain only approximately 30 problems each, meaning that the outcome of a single problem can change the score of one evaluation run by roughly 3.33 percentage points. Consequently, results from a single run may depend heavily on a small number of stochastic generation outcomes. This evaluation strategy is consistent with the multi-sampling protocol adopted in DAG-MATH (Zhang et al., 2026), which estimates reasoning performance from multiple independently generated trajectories for each problem. Post-hoc reasoning-graph parsing uses an output limit of 8,192 tokens, and execution-based verification is implemented with SymPy.

4.2 EXPERIMENTAL RESULTS

Overall Performance Table 1 compares TOPOMATH with representative inference-time reasoning methods on six mathematical reasoning benchmarks using DeepSeekV4-Flash as the common backbone. TOPOMATH achieves the highest macro-average score on all four metrics and ranks first in 19 of the 24 dataset–metric comparisons. With equal weight assigned to each benchmark, it improves PASS@1 from 0.575 to 0.708 and PRR from 0.223 to 0.521 over the backbone, yielding absolute gains of 13.2 and 29.8 percentage points, respectively. Closure-AUC increases from 0.536 to 0.687, and Mean Closure from 0.905 to 0.960.

Against Code + Text + Sum., the strongest baseline in macro-average PASS@1 and Closure-AUC, TOPOMATH improves the two metrics by 2.7 percentage points and 0.069, respectively. Its macro-average PRR reaches 0.521, compared with 0.225 for Code + Text + Sum. and 0.220 for Self-Consistency@5. TOPOMATH also achieves a macro-average Mean Closure of 0.960, exceeding the highest baseline value of 0.925 obtained by Least-to-Most.

The process-level advantages extend across benchmarks: TOPOMATH ranks first in Closure-AUC on all six benchmarks and in both PRR and Mean Closure on five of the six benchmarks. It achieves the highest scores on all four metrics on AIME 2025 and BRUMO 2025. On AIME 2026, TOPOMATH obtains PASS@1 of 0.760 and PRR of 0.617, compared with 0.800 and 0.267 for Code + Text + Sum., respectively. The PRR gains indicate that TOPOMATH more frequently produces answer-correct trajectories satisfying the full-closure criterion.

Cross-Backbone Generalization To assess cross-backbone generalization, we evaluate TOPOMATH with Qwen3.5-27B and Qwen3-8B alongside DeepSeekV4-Flash. Each backbone is compared with its direct inference baseline on all six benchmarks. For this experiment, we report only PASS@1 in the main text; complete results for all four metrics are provided in Appendix F.

As shown in Table 2, TOPOMATH improves PASS@1 in all 18 backbone–benchmark comparisons. Averaged equally across the six benchmarks, the gains are 13.2, 24.1, and 2.9 percentage points for DeepSeekV4-Flash, Qwen3.5-27B, and Qwen3-8B, respectively. Qwen3.5-27B exhibits the largest improvement, with macro-average PASS@1 increasing from 0.459 to 0.700 and gains of 33.4 and 38.7 percentage points on AIME 2025 and AIME 2026. Qwen3-8B also benefits across all six benchmarks, with improvements ranging from 1.3 to 5.0 percentage points. These results show that TOPOMATH's accuracy gains extend across the evaluated model families and parameter scales.

Scaffold Complexity and Irrelevant Reasoning Nodes We analyzed 93 problems from AIME 2026, BRUMO 2025, and HMMT 2026, measuring empirical difficulty as the baseline's error rate over ten runs. Difficulty was positively correlated with scaffold node counts (Spearman's $\rho = 0.286$,

Table 1: Answer correctness and process-level reasoning quality on six mathematical reasoning benchmarks. All metrics are reported on a scale of $[0, 1]$. The Δ columns report absolute differences relative to DeepSeekV4-Flash. The best result for each metric is highlighted in bold.

Dataset	Method	PASS@1		PRR		Closure-AUC		Mean Closure	
		Score	Δ	Score	Δ	Score	Δ	Score	Δ
AIME 2025	DeepSeekV4-Flash	0.480	–	0.170	–	0.453	–	0.911	–
	CoT-ZeroShot	0.500	+0.020	0.003	-0.167	0.464	+0.011	0.899	-0.012
	Self-Consistency@5	0.500	+0.020	0.100	-0.070	0.463	+0.010	0.919	+0.008
	PAL	0.367	-0.113	0.167	-0.003	0.351	-0.102	0.925	+0.014
	Least-to-Most	0.467	-0.013	0.200	+0.030	0.443	-0.010	0.936	+0.025
	Code + Text + Sum.	0.733	+0.253	0.167	-0.003	0.652	+0.199	0.895	-0.016
	SBSC	0.300	-0.180	0.067	-0.103	0.254	-0.199	0.884	-0.027
	CoMAT	0.667	+0.187	0.100	-0.07	0.551	+0.098	0.803	-0.108
	TopoMath	**0.753**	**+0.273**	**0.637**	**+0.467**	**0.743**	**+0.290**	**0.974**	**+0.063**
AIME 2026	DeepSeekV4-Flash	0.590	–	0.127	–	0.548	–	0.910	–
	CoT-ZeroShot	0.633	+0.043	0.133	+0.006	0.585	+0.037	0.907	-0.003
	Self-Consistency@5	0.700	+0.110	0.200	+0.073	0.659	+0.111	0.925	+0.015
	PAL	0.467	-0.123	0.133	+0.006	0.428	-0.120	0.881	-0.029
	Least-to-Most	0.533	-0.057	0.133	+0.006	0.503	-0.045	0.916	+0.006
	Code + Text + Sum.	**0.800**	**+0.210**	0.267	+0.140	0.743	+0.195	0.923	0.013
	SBSC	0.233	-0.357	0.067	-0.060	0.213	-0.335	0.892	-0.018
	CoMAT	0.633	+0.043	0.067	-0.060	0.538	-0.010	0.859	-0.051
	TopoMath	0.760	+0.170	**0.617**	**+0.490**	**0.748**	**+0.200**	**0.974**	**+0.064**
BRUMO 2025	DeepSeekV4-Flash	0.637	–	0.210	–	0.589	–	0.903	–
	CoT-ZeroShot	0.633	-0.004	0.267	+0.057	0.587	-0.002	0.892	-0.011
	Self-Consistency@5	0.700	+0.063	0.200	-0.010	0.654	+0.065	0.912	+0.009
	PAL	0.567	-0.070	0.300	+0.090	0.518	-0.071	0.909	+0.006
	Least-to-Most	0.500	-0.137	0.133	-0.077	0.467	-0.122	0.923	+0.020
	Code + Text + Sum.	0.667	+0.030	0.133	-0.077	0.605	+0.016	0.815	-0.088
	SBSC	0.467	-0.170	0.233	+0.023	0.437	-0.152	0.924	+0.021
	CoMAT	0.633	-0.004	0.067	-0.143	0.560	-0.029	0.846	-0.057
	TopoMath	**0.740**	**+0.103**	**0.440**	**+0.230**	**0.710**	**+0.121**	**0.955**	**+0.052**
HMMT 2026	DeepSeekV4-Flash	0.327	–	0.027	–	0.291	–	0.868	–
	CoT-ZeroShot	0.364	+0.037	0.000	-0.027	0.321	+0.030	0.853	-0.015
	Self-Consistency@5	**0.485**	**+0.158**	0.000	-0.027	0.430	+0.139	0.882	+0.014
	PAL	0.333	+0.006	0.091	+0.064	0.288	-0.003	0.871	+0.003
	Least-to-Most	0.424	+0.097	0.121	+0.094	0.379	+0.088	0.903	+0.035
	Code + Text + Sum.	0.455	+0.128	0.152	+0.125	0.398	+0.107	0.877	+0.009
	SBSC	0.212	-0.115	0.061	+0.034	0.174	-0.117	0.868	+0.000
	CoMAT	0.333	+0.006	0.030	+0.003	0.307	+0.016	0.772	-0.096
	TopoMath	0.476	+0.149	**0.342**	**+0.315**	**0.464**	**+0.173**	**0.947**	**+0.079**
MATH-500	DeepSeekV4-Flash	0.924	–	0.636	–	0.885	–	0.956	–
	CoT-ZeroShot	0.930	+0.006	0.598	-0.038	0.885	0.000	0.948	-0.008
	Self-Consistency@5	0.936	+0.012	**0.656**	**+0.020**	0.900	+0.015	**0.960**	**+0.004**
	PAL	0.798	-0.126	0.532	-0.104	0.751	-0.134	0.939	-0.017
	Least-to-Most	0.872	-0.052	0.570	-0.066	0.834	-0.051	0.957	+0.001
	Code + Text + Sum.	0.910	-0.014	0.448	-0.188	0.842	-0.043	0.923	-0.033
	SBSC	0.758	-0.166	0.508	-0.128	0.717	-0.168	0.935	-0.021
	CoMAT	0.902	-0.022	0.134	-0.502	0.754	-0.131	0.833	-0.123
	TopoMath	**0.944**	**+0.020**	0.652	+0.016	**0.902**	**+0.017**	0.955	-0.001
Omni-500	DeepSeekV4-Flash	0.494	–	0.166	–	0.449	–	0.884	–
	CoT-ZeroShot	0.500	+0.006	0.164	-0.002	0.453	+0.004	0.867	-0.017
	Self-Consistency@5	**0.582**	**+0.088**	0.162	-0.004	0.525	+0.076	0.885	+0.001
	PAL	0.412	-0.082	0.178	+0.012	0.374	-0.075	0.892	+0.008
	Least-to-Most	0.516	+0.022	0.188	+0.022	0.476	+0.027	0.914	+0.030
	Code + Text + Sum.	0.518	+0.024	0.180	+0.014	0.471	+0.022	0.878	-0.006
	SBSC	0.186	-0.308	0.054	-0.112	0.166	-0.283	0.621	-0.263
	CoMAT	0.488	-0.006	0.050	-0.116	0.410	-0.039	0.796	-0.088
	TopoMath	0.572	+0.078	**0.436**	**+0.270**	**0.556**	**+0.107**	**0.955**	**+0.071**

$P = 0.005$), edge counts ($\rho = 0.305$, $P = 0.003$), and branch counts ($\rho = 0.300$, $P = 0.004$; Figure 2(a)), indicating that problems with lower baseline accuracy tended to yield larger, more branched scaffolds.

We defined irrelevant nodes as those outside the final-answer node's ancestry in the parsed reasoning graph, excluding the final-answer node itself, which was identified as the last parsed step. Across 930 paired runs with valid parses for both methods, we computed each problem's mean irrelevant-node count and its reduction from the baseline to TopoMath. Scaffold node count was positively

Table 2: Cross-backbone generalization of TOPOMATH across six benchmarks. Δ denotes the gain over direct inference using the same backbone.

Backbone	Method	AIME25	AIME26	BRUMO25	HMMT26	MATH-500	Omni-500	Avg.
DeepSeekV4-Flash	Direct	0.480	0.590	0.637	0.327	0.924	0.494	0.575
	TOPOMATH	0.753	0.760	0.740	0.476	0.944	0.572	0.708
	Δ	+0.273	+0.170	+0.103	+0.149	+0.020	+0.078	**+0.132**
Qwen3.5-27B	Direct	0.333	0.400	0.470	0.242	0.924	0.386	0.459
	TOPOMATH	0.667	0.787	0.750	0.458	0.952	0.586	0.700
	Δ	+0.334	+0.387	+0.280	+0.216	+0.028	+0.200	**+0.241**
Qwen3-8B	Direct	0.200	0.123	0.320	0.130	0.798	0.326	0.316
	TOPOMATH	0.213	0.173	0.350	0.155	0.828	0.354	0.346
	Δ	+0.013	+0.050	+0.030	+0.025	+0.030	+0.028	**+0.029**

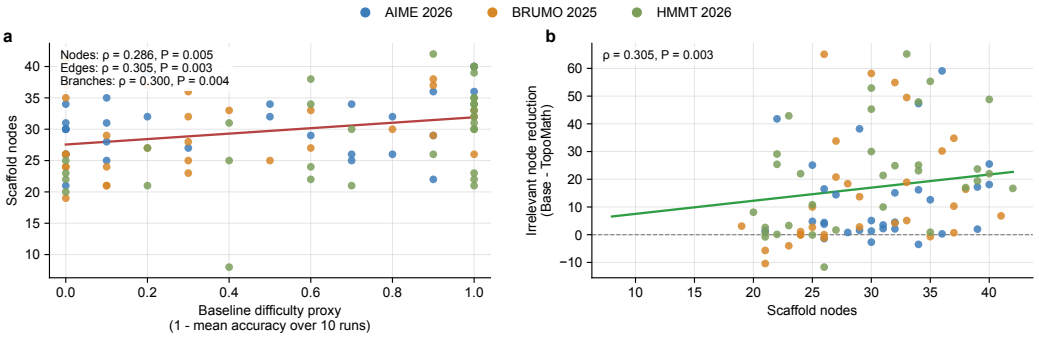

Figure 2: Scaffold complexity and irrelevant reasoning nodes. (a) Associations between baseline-derived difficulty and scaffold complexity. The vertical axis shows scaffold node counts; annotations additionally report correlations with scaffold edge and branch counts. (b) Association between scaffold node counts and the reduction in irrelevant reasoning nodes, defined as the baseline count minus the TOPOMATH count. An irrelevant node is neither the stated final-answer node nor one of its ancestors in the parsed reasoning graph. Solid lines show linear fits for visualization; annotations report two-sided Spearman correlations with unadjusted P values.

correlated with this reduction ($\rho = 0.305$, $P = 0.003$; Figure 2(b)). The overall mean count decreased from 20.72 to 3.89, an 81.2% reduction.

Reasoning Graph Structure and Logical Closure We compared final-output reasoning graphs across 93 problems using 930 valid paired runs, averaging graph statistics over matched runs within each problem. As shown in Figure 3(a), TOPOMATH reduced the mean node count from 52.64 to 19.95 and mean reasoning depth from 21.28 to 12.18. The mean number of irrelevant nodes—those outside the final-answer node and its dependency ancestry—decreased from 20.72 to 3.64 (Figure 3(b)).

Mean structural closure, measured as the fraction of non-final nodes with outgoing edges, increased from 0.894 to 0.958, while fully closed runs increased from 15.1% to 61.5% (Figure 3(c)). Node count, irrelevant-node count, and closure differences were significant under two-sided Wilcoxon signed-rank tests on paired problem-level means (all unadjusted $P < 0.0001$). These results show that TOPOMATH produced more compact final outputs with fewer irrelevant nodes and greater structural closure.

Ablation Study To analyze the contribution of each component in TOPOMATH, we conduct ablation studies by removing individual modules while keeping all other experimental settings unchanged. We consider the following variants:

- **w/o Topology Scaffold**: removes the topology scaffold while retaining the reasoning and verification modules. This variant removes the explicit structural prior induced by the mathematical concept topology.

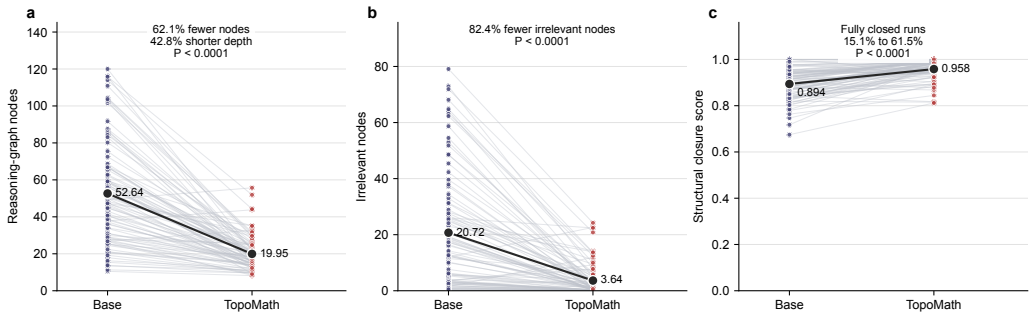

Figure 3: Reasoning-graph structure and closure across 93 problems. Gray lines connect paired problem-level means; black markers show means across problems. (a) node count; (b) irrelevant-node count; (c) structural closure. Annotations report depth reduction and fully closed run proportions. Displayed P values are unadjusted two-sided Wilcoxon signed-rank tests on paired problem-level means of the plotted metrics.

Table 3: Ablation results on Omni-500. The Δ columns report absolute changes relative to the complete TopoMath framework.

Method	PASS@1		PRR		Closure-AUC		Mean Closure	
	Score	Δ	Score	Δ	Score	Δ	Score	Δ
TopoMath	**0.572**	–	**0.436**	–	**0.556**	–	**0.955**	–
w/o Dual Verification	0.524	-0.048	0.406	-0.030	0.510	-0.046	0.941	-0.014
w/o Scaffold	0.556	-0.016	0.430	-0.006	0.548	-0.008	0.948	-0.007

- **w/o Dual Verification**: removes both execution-based verification and topology-based verification, and directly outputs the generated reasoning result without refinement.

All variants are evaluated under identical settings, including backbone model, prompting strategy, decoding configuration, and evaluation protocol.

As shown in Table 3, removing either component reduces all four evaluation metrics. Removing dual verification produces the larger decline across all four metrics. PASS@1 decreases from 0.572 to 0.524, and PRR falls from 0.436 to 0.406, corresponding to reductions of 4.8 and 3.0 percentage points. Closure-AUC and Mean Closure decrease by 0.046 and 0.014, respectively. Dual verification therefore contributes to both final-answer accuracy and the frequency of answer-correct, fully closed trajectories. The topology scaffold provides additional gains when the reasoning and verification modules are retained. Removing it reduces PASS@1 from 0.572 to 0.556 and PRR from 0.436 to 0.430, corresponding to decreases of 1.6 and 0.6 percentage points. Closure-AUC falls from 0.556 to 0.548, while Mean Closure decreases from 0.955 to 0.948.

To complement the quantitative evaluation, we provide a case study in Appendix F.1 (Figure 4) illustrating adaptive concept expansion, scaffold-guided reasoning, and dual verification on an AIME 2025 problem.

5 Conclusion

We presented TopoMath, an inference-time framework that organizes mathematical knowledge into problem-specific concept scaffolds. Built on a concept relation graph, the framework combines scaffold-guided generation with execution-based and concept-membership feedback for targeted revision. Experiments on six benchmarks and three backbones show consistent improvements in PASS@1 over direct inference. With DeepSeekV4-Flash, TopoMath achieves the highest macro-average scores on all four metrics among the compared methods, improving PASS@1 and PRR over direct inference by 13.2 and 29.8 percentage points, respectively. Analysis of final-output reasoning graphs further confirms fewer irrelevant nodes and greater structural closure.

AI USE STATEMENT

In this work, we used generative AI tools to polish the manuscript and prepare structured data for process-level reasoning evaluation. Specifically, for process-level reasoning evaluation based on the DAG-MATH protocol, we used DeepSeek-flash to parse the natural-language reasoning trajectories generated by the evaluated models into the structured representation required for subsequent reasoning-graph construction and metric computation. This AI-assisted parsing was used only as an intermediate data-processing step and was not used to modify the original model responses, determine final-answer correctness, or directly assign evaluation scores.

We have not used generative AI tools for help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, support qualitative and thematic data analysis, interpret results , and Generating synthetic datasets is not applicable to this work.

We have reviewed all AI-assisted work. The outputs produced during AI-assisted data processing were used only to support the predefined evaluation pipeline, while the reported metrics were computed according to the evaluation procedures described in the paper. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the TOPOMATH framework, experimental settings, evaluation metrics, and compared methods in the main text. The appendix further documents the implementation details of topology construction, problem-specific topology projection, reasoning and verification, the Omni-MATH-500 sampling protocol, and the reasoning-process evaluation procedure, including the corresponding prompts. Codes and datasets required to reproduce our experiments are available at `https://anonymous.4open.science/r/Topomath-B7E6/`. We release the implementation, evaluation scripts, prompts, and processed experimental data to facilitate reproduction of the reported results.

REFERENCES

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Qiaochu Jiang, Jia Deng, Stella R Biderman, and Sean Welleck. Llemma: An open language model for mathematics. In *International Conference on Learning Representations*, volume 2024, pp. 40622–40649, 2024. URL `https://openreview.net/forum?id=4WnqRR915j`.

Mislav Balunovic, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. MathArena: Evaluating llms on uncontaminated math competitions. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-0679.

Maciej Besta, Nils Blach, Aleš Kubíček, Robert Gerstenberger, Michał Podstawski, Lukáš Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 17682–17690, 2024. doi: 10.1609/aaai.v38i16.29720.

Qikai Chang, Zhenrong Zhang, Pengfei Hu, Jun Du, Jiefeng Ma, Yicheng Pan, Jianshu Zhang, Quan Liu, and Gao Jianqing. THOR: Tool-integrated hierarchical optimization via RL for mathematical reasoning. In *International Conference on Learning Representations*, volume 2026, pp. 21487–21510, 2026. URL `https://openreview.net/forum?id=0Af7UiJISU`.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*, 2023. URL `https://openreview.net/forum?id=YfZ4ZPt8zd`.

Yongchao Chen, Harsh Jhamtani, Srinagesh Sharma, Chuchu Fan, and Chi Wang. Steering large language models between code execution and textual reasoning. In *International Con-*

ference on Learning Representations, volume 2025, pp. 99676–99707, 2025. URL https://openreview.net/forum?id=5X5Z7Ffrjb.

Debrup Das, Debopriyo Banerjee, Somak Aditya, and Ashish Kulkarni. MATHSENSEI: A tool-augmented large language model for mathematical reasoning. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 942–966, 2024. doi: 10.18653/v1/2024.naacl-long.54.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence, 2026. URL https://arxiv.org/abs/2606.19348.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A GraphRAG approach to query-focused summarization, 2024. URL https://arxiv.org/abs/2404.16130.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, runxin xu, Zhengyang Tang, Wang Benyou, Daoguang Zan, Shanghaoran Quan, Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. Omni-MATH: A universal Olympiad level mathematic benchmark for large language models. In *International Conference on Learning Representations*, volume 2025, pp. 100540–100569, 2025. URL https://openreview.net/forum?id=yaqPf0KAlN.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 10764–10799, 2023. URL https://proceedings.mlr.press/v202/gao23f.html.

Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. rStar-Math: Small LLMs can master math reasoning with self-evolved deep thinking. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 20640–20661, 2025. URL https://proceedings.mlr.press/v267/guan25f.html.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://openreview.net/forum?id=7Bywt2mQsCe.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pp. 22199–22213, 2022. URL https://openreview.net/forum?id=e2TBb5y0yFf.

Xin Lai, Zhuotao Tian, Yukang Chen, Senqiao Yang, Xiangru Peng, and Jiaya Jia. Step-DPO: Step-wise preference optimization for long-chain reasoning of LLMs, 2024. URL https://arxiv.org/abs/2406.18629.

Joshua Ong Jun Leang, Aryo Pradipta Gema, and Shay B. Cohen. CoMAT: Chain of mathematically annotated thought improves mathematical reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20245–20274, 2025. doi: 10.18653/v1/2025.emnlp-main.1024.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf.

Yinghui Li, Jiayi Kuang, Haojing Huang, Zhikun Xu, Xinnian Liang, Yi Yu, Wenlian Lu, Yangning Li, Xiaoyu Tan, Chao Qu, Ying Shen, Hai-Tao Zheng, and Philip S. Yu. One example shown,

many concepts known! Counterexample-driven conceptual reasoning in mathematical LLMs. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 35168–35185, 2025. URL https://proceedings.mlr.press/v267/li25ax.html.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Chengwu Liu, Ye Yuan, Yichun Yin, Yan Xu, Xin Xu, Zaoyu Chen, Yasheng Wang, Lifeng Shang, Qun Liu, and Ming Zhang. Safe: Enhancing mathematical reasoning in large language models via retrospective step-aware formal verification. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12171–12186, 2025. doi: 10.18653/v1/2025.acl-long.594.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jian-Guang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, Yansong Tang, and Dongmei Zhang. WizardMath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. In *International Conference on Learning Representations*, volume 2025, pp. 49573–49609, 2025. URL https://openreview.net/forum?id=mMPMHWOdOy.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46534–46594, 2023. URL https://openreview.net/forum?id=S37hOerQLB.

Sagnik Mukherjee, Abhinav Chinta, Takyoung Kim, Tarun Anoop Sharma, and Dilek Hakkani Tur. Premise-augmented reasoning chains improve error identification in math reasoning with LLMs. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 45109–45128, 2025. URL https://proceedings.mlr.press/v267/mukherjee25a.html.

Tushar Pandey, Ara Ghukasyan, Oktay Goktas, and Santosh Kumar Radha. Adaptive graph of thoughts: Test-time adaptive reasoning unifying chain, tree, and graph structures, 2025. URL https://arxiv.org/abs/2502.05078.

Qwen Team. Qwen3 technical report. *arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652, 2023. URL https://openreview.net/forum?id=vAElhFcKW6.

Kunal Singh, Ankan Biswas, Sayandeep Bhowmick, Pradeep Moturi, and Siva Gollapalli. SBSC: Step-by-step coding for improving mathematical Olympiad performance. In *International Conference on Learning Representations*, volume 2025, pp. 31603–31653, 2025. URL https://openreview.net/forum?id=JgW2VGxvYV.

Denny Vrandečić and Markus Krötzsch. Wikidata: A free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014. doi: 10.1145/2629489.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022. URL https://openreview.net/forum?id=_VjQlMeSB_J.

Shichao Xia, Ximing Li, Yujiu Liu, Tong Wu, and Pingzhong Liu. Evaluating mathematical reasoning beyond accuracy. In *Proceedings of the 39th AAAI Conference on Artificial Intelligence*, volume 39, pp. 27723–27730, 2025. doi: 10.1609/aaai.v39i26.34987.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 11809–11822, 2023. URL https://openreview.net/forum?id=5Xc1ecxO1h.

Huaiyuan Ying, Shuo Zhang, Linyang Li, Zhejian Zhou, Yunfan Shao, Zhaoye Fei, Yichuan Ma, Jiawei Hong, Kuikun Liu, Ziyi Wang, Yudong Wang, Zijian Wu, Shuaibin Li, Fengzhe Zhou, Hongwei Liu, Songyang Zhang, Wenwei Zhang, Hang Yan, Xipeng Qiu, Jiayu Wang, Kai Chen, and Dahua Lin. InternLM-Math: Open math large language models toward verifiable reasoning, 2024. URL https://arxiv.org/abs/2402.06332.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. MetaMath: Bootstrap your own mathematical questions for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 45040–45061, 2024. URL https://openreview.net/forum?id=N8N0hgNDRt.

Youliang Yuan, Qiuyang Mang, Jingbang Chen, Hong Wan, Xiaoyuan Liu, Junjielong Xu, Jen-tse Huang, Wenxuan Wang, Wenxiang Jiao, and Pinjia He. Curing miracle steps in LLM mathematical reasoning with rubric rewards. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 18556–18577, 2026. doi: 10.18653/v1/2026.acl-long.844.

Yuanhe Zhang, Ilja Kuzborskij, Jason Lee, Chenlei Leng, and Fanghui Liu. DAG-Math: Graph-of-thought guided mathematical reasoning in LLMs. In *International Conference on Learning Representations*, volume 2026, pp. 74347–74389, 2026. URL https://openreview.net/forum?id=ylr6WArKQN.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WZH7099tgfM.

Zihao Zhou, Shudong Liu, Maizhen Ning, Wei Liu, Jindong Wang, Derek Wong, Xiaowei Huang, Qiufeng Wang, and Kaizhu Huang. Is your model really a good math reasoner? evaluating mathematical reasoning with checklist. In *International Conference on Learning Representations*, volume 2025, pp. 34238–34281, 2025. URL https://openreview.net/forum?id=nDvgHIBRxQ.

A NOTATION SUMMARY

Tables 4 and 5 summarize the notations used in Stages I, II, and III.

Table 4: Main notation used in Stages I and II.

Symbol	Description
$\mathcal{G}_{\text{field}}$	Field-level mathematical backbone constructed from the predefined root domains through bounded breadth-first expansion over the Wikidata *subclass* and *part-of* relations.
$\mathcal{G}_0$	Initial mathematical concept graph obtained after offline concept-layer enrichment of $\mathcal{G}_{\text{field}}$, before topology governance.
$\Phi(\cdot)$	Topology-governance operator that transforms $\mathcal{G}_0$ through concept filtering, cycle elimination, transitive reduction, and reachability-based pruning.
$\mathcal{G}^*$	Final reasoning-ready global mathematical concept topology obtained after topology governance and used as the structural prior for problem-specific projection.
Ψ	Problem-specific topology projection operator that maps the global topology and an input problem to its local topology and textual scaffold.
$\widetilde{\mathcal{G}}^{(t)}$	Dynamically evolving working topology used when processing the t-th problem, initialized from $\mathcal{G}^*$ at the beginning of a scaffold-generation run and incrementally augmented by accepted concept expansions.
Γ	Concept-grounding operator that maps mathematical concept phrases extracted from the problem to nodes in the working topology.
A_q	Set of grounded concept anchors successfully mapped to nodes in the working topology.
U_q	Set of concept phrases that cannot be grounded to the current working topology.
V_q^+	Set of newly accepted concept nodes introduced through adaptive topology expansion.
E_q^+	Set of newly accepted concept relations introduced through adaptive topology expansion.
Π_q	Set of selected root-to-anchor paths connecting mathematical root domains to grounded concept anchors.
L_q	Set of deepest common ancestors shared by multiple grounded anchors.
N_q	Bounded set of locally connected concepts associated with the grounded anchors.
$\mathcal{T}_q$	Compact structural representation of the problem-specific topology, defined as $\mathcal{T}_q = (A_q, \Pi_q, L_q, N_q)$.
V_q	Node set of the problem-specific topology, consisting of anchors, root-to-anchor path nodes, common ancestors, and local neighbors.
G_q	Problem-specific local mathematical topology induced from the working topology using V_q.
$\text{Render}(\cdot)$	Rendering function that converts the problem-specific topology structure into a textual topology scaffold.
S_q	Textual topology scaffold supplied to the backbone language model as inference-time structural guidance.

B DETAILED IMPLEMENTATION OF TOPOMATH

This appendix details the three stages of TOPOMATH. We first describe the offline construction of the mathematical concept DAG from Wikidata, followed by concept grounding, adaptive topology expansion, and problem-specific scaffold generation. We then explain how the scaffold guides solution generation and how execution-based and concept-membership verification provide diagnostic feedback for targeted revision.

B.1 DETAILED TOPOLOGY CONSTRUCTION

Stage I constructs the offline mathematical concept DAG $\mathcal{G}^*$. Its edges represent taxonomic, compositional, or associative relations, rather than theorem prerequisites or proof dependencies.

B.1.1 ROOT DOMAINS AND RELATION EXTRACTION

Extraction starts from six Wikidata domains: algebra (Q3968), geometry (Q8087), number theory (Q12479), combinatorics (Q76592), probability theory (Q5862903), and calculus (Q149972). These roots delimit the extraction scope without assuming a complete or mutually exclusive classification of mathematics.

The initial backbone is extracted by breadth-first traversal over P279 (*subclass of*) and P361 (*part of*), with roots at depth zero and a maximum traversal depth of $D_{\max} = 3$. For each parent u, a retrieved claim (v, p, u) is stored as $u \to v$, orienting edges from superclass to subclass or from whole to component. If both properties connect the same ordered pair during initial extraction, P279 takes precedence. The backbone is then enriched with P1269 (*facet of*) connections, one-hop P279 descendants of the retrieved facet concepts, and 15 manually reviewed seed connections. This offline enrichment is separate from Stage II online expansion.

Table 5: Main notation used in Stage III.

Symbol	Description
$\mathcal{M}_\theta$	Backbone language model used to generate mathematical reasoning trajectories, where θ denotes the model parameters.
r_q^s	Topology-guided reasoning trajectory generated by conditioning the backbone model on problem q and its topology scaffold S_q.
r_q^0	Unguided reference reasoning trajectory generated from problem q without topology guidance and used only as auxiliary diagnostic evidence.
$\mathcal{C}(\cdot)$	Executable-claim extraction operator that extracts explicit machine-verifiable mathematical claims from the topology-guided reasoning trajectory.
X_q	Set of executable mathematical claims extracted from r_q^s for execution-based verification.
$\mathcal{Z}(\cdot)$	Concept extraction operator that identifies the mathematical concepts involved in the generated reasoning trajectory.
Z_q	Concept trajectory or concept set extracted from r_q^s for topology-based verification.
$\mathcal{V}_{\text{exec}}$	Execution-based verifier that checks symbolic operations and intermediate mathematical computations.
e_q^{exec}	Execution-verification result, including the verification decision and diagnostic information for detected computational inconsistencies.
$\mathcal{V}_{\text{topo}}$	Topology-based verifier that checks whether the mathematical concepts invoked during reasoning are supported by the problem-specific topology.
e_q^{topo}	Topology-verification result indicating whether the concepts involved in the reasoning remain supported by the problem-specific topology.
$a(\cdot)$	Final-answer extraction function that extracts the answer from a generated reasoning trajectory.
$\text{Equivalent}(\cdot,\cdot)$	Answer-equivalence function used to determine whether two extracted final answers are mathematically equivalent.
d_q	Diagnostic indicator of disagreement between the final answers produced by the topology-guided and unguided trajectories.
$\text{Fail}(\cdot)$	Function indicating whether a verification result contains a detected failure.
τ_q	Binary revision indicator that determines whether targeted revision is triggered by execution-based or topology-based verification.
ξ_q	Collection of diagnostic information supplied to the targeted revision module, including verifier feedback and auxiliary reference information.
$\mathcal{R}_\theta$	Targeted revision operator that corrects unreliable portions of the topology-guided trajectory using diagnostic feedback.
$\widetilde{r}_q$	Final accepted reasoning trajectory, corresponding to either the original topology-guided trajectory or its once-revised version.
$\widehat{a}_q$	Final answer returned by TopoMath, extracted from the accepted reasoning trajectory $\widetilde{r}_q$.

B.1.2 Filtering and DAG Construction

Nodes are deduplicated by Wikidata QID. Non-root entities lacking both Chinese and English labels are removed. A manually reviewed gateway blocklist excludes clearly irrelevant branches. Self-loops, dangling references, and nodes unreachable from all roots are also removed. Human curation is limited to the blocklist and seed connections.

Cycles are checked using NetworkX. When a cycle is found, edge removal prioritizes `part_of` relations, then edges pointing toward shallower nodes, with QID ordering used to break ties. After acyclicity is established, transitive reduction removes reachability-redundant edges from the untyped graph. This preserves reachability, not the semantics of every removed relation. Filtering and structural checks are applied to the backbone and repeated after enrichment.

The final graph contains 2,346 nodes and 2,424 edges: 1,613 subclass, 499 part-of, 298 facet, and 14 seed edges. Every retained node is reachable from at least one root. Node depth is recomputed as the shortest distance from any root; the maximum is seven. Thus, $D_{\max} = 3$ bounds only the initial backbone traversal, not the depth of the enriched graph.

B.1.3 Storage and Retrieval Assets

The graph is stored in `skeleton_en.json`. Node records contain QIDs, English labels and descriptions, depths, and root indicators; edge records contain parent and child QIDs and relation types. Derived assets include adjacency lists, ancestor and root-path caches, and a concept index mapping normalized English labels and non-Chinese aliases to candidate QIDs. Normalization applies lowercasing, character-width normalization, and removal of whitespace and punctuation. Ambiguous keys retain all candidate QIDs for subsequent grounding.

B.1.4 REPRODUCIBILITY AND SCOPE

Raw extraction responses are cached in `fetch_raw.json`, supporting reconstruction without repeating the initial Wikidata queries. The offline graph $\mathcal{G}^*$ remains frozen during inference. Stage II expansions are stored in separate dynamic assets and do not overwrite this offline snapshot.

B.2 DETAILED PROBLEM-SPECIFIC PROJECTION

Stage II constructs a problem-specific subgraph scaffold through concept extraction, grounding, adaptive expansion, and local structure retrieval. It operates on a dynamic working graph stored separately from the frozen offline DAG $\mathcal{G}^*$. Accepted expansions can be reused by subsequent problems, and the dynamic assets can be reset to $\mathcal{G}^*$ before a new run. Once generated, each problem's scaffold remains fixed during subsequent reasoning.

B.2.1 CONCEPT EXTRACTION AND GROUNDING

Concept extraction uses DeepSeekV4-Flash with a temperature of 0.7 and a maximum output length of 512 tokens. The original problem is supplied as the user message, with the following system instruction:

> You are a mathematical concept extractor. Extract every relevant mathematical concept, object, operation, and theorem name from the problem using standard English terminology. Return only a JSON array of strings with no explanation; for example: ["derivative", "maxima and minima"].

No explicit limit is imposed on the number of extracted phrases. If extraction fails or returns an empty result, a lexical fallback searches the normalized problem text for indexed concept names, prioritizes longer matches, and removes shorter matches that are contained in already selected phrases.

Extracted phrases are normalized by lowercasing, character-width normalization, and removal of whitespace and punctuation, including hyphens, slashes, and brackets. Optional traditional-to-simplified Chinese conversion is supported, but general lemmatization and singular–plural normalization are not applied.

Grounding follows an exact–containment–fuzzy cascade. Exact matching requires equality between normalized strings. Otherwise, bidirectional substring matching is attempted: an indexed key contained in the query receives a score of $|k|/|x|$, while a query contained in an indexed key receives a score of $|x|/|k|$ and is accepted only if this ratio is at least 0.5. Candidates are ranked by score, key length, and lexical key order. If containment fails, the highest-scoring RapidFuzz `WRatio` match is accepted when its score is at least 88 on a 0–100 scale. Semantic matching is disabled in the reported experiments.

If a matched key corresponds to multiple QIDs, all candidates are retained. Anchors are deduplicated by QID while preserving their first occurrence in the extracted sequence. Unmatched phrases are passed to adaptive expansion.

B.2.2 ADAPTIVE TOPOLOGY EXPANSION

Adaptive topology expansion extends both the concept coverage and relational structure of the working topology. In addition to P279 (*subclass of*), P361 (*part of*), and P1269 (*facet of*) used in offline construction, online connection search also considers P31 (*instance of*) and P2579 (*studied by*). These additional relation types provide further ways to connect previously unmatched concepts to the existing topology.

For each unmatched phrase, TOPOMATH searches Wikidata using the original phrase and manually defined singular variants, retaining at most eight candidates per query variant. Candidate metadata and outgoing claims are retrieved, and connection search follows P279, P361, P1269, P31, and P2579 in that priority order. A breadth-first search of at most three hops seeks a connection to an existing working-graph node.

Candidates must pass entity filters, and new connections must preserve acyclicity. A retrieved multi-hop path is stored as connection evidence, while the candidate is attached to the existing graph

through a single expansion edge that summarizes the retrieved connection. Thus, graph updates incorporate accepted concept nodes together with their connections to the working topology.

If no path is found, a keyword-based fallback may attach the candidate to a predefined mathematical domain concept. These heuristic connections are marked as `domain_bridge` edges. Duplicate and cycle-producing edges are rejected; no online transitive reduction is applied. After expansion, unmatched phrases are grounded again. Any remaining unmatched phrases are excluded from local structure retrieval.

B.2.3 Local Subgraph Construction

For the final anchor set A_q, TopoMath constructs

$$\mathcal{T}_q = (A_q, \Pi_q, L_q, N_q), \tag{6}$$

where Π_q contains selected root-to-anchor paths, L_q contains maximum-depth common ancestors, and N_q contains neighboring concepts.

For each anchor and each reachable root, one shortest directed path is retained. An anchor without a valid root path is retained as a singleton path. When at least two anchors are available, common ancestors are selected using the cached ancestor information, retaining those with the greatest stored node depth, including ties. If no common ancestor exists or fewer than two anchors are available, $L_q = \varnothing$. This depth-based selection is not a general DAG lowest-common-ancestor computation.

Neighborhood retrieval considers direct children and siblings sharing a parent in the working graph. Candidates follow anchor order and graph-edge order, and the first 12 distinct neighbors are retained. The resulting node set is

$$\mathcal{V}_q = A_q \cup \mathrm{Nodes}(\Pi_q) \cup L_q \cup N_q. \tag{7}$$

The local topology is stored through selected paths and neighborhood provenance in $\mathcal{T}_q$, rather than a complete serialized edge list of the induced subgraph.

B.2.4 Scaffold Rendering and Caching

The textual scaffold is obtained as $S_q = \mathrm{Render}(\mathcal{T}_q)$. It lists grounded concepts, domain paths, common ancestors, neighboring concepts, available anchor definitions, and unmatched phrases marked as ignored. Anchors and common ancestors include labels and QIDs; paths use labels only. Neighborhood entries distinguish children from siblings. Wikidata edge types are not exposed in the rendered prompt.

Anchors retain extraction order, paths follow anchor and root order, duplicate textual paths are removed, common ancestors are ordered by QID, and neighbors retain retrieval order. No separate scaffold-specific token limit is imposed. The solver is instructed to treat the scaffold as an orientation aid, ignore irrelevant concepts, prioritize the problem statement, and avoid mentioning the graph in its solution.

Scaffolds are cached per problem and reused across repeated reasoning runs without repeating extraction or expansion. If no concept can be grounded, an empty-anchor scaffold record is retained; the system does not automatically switch to direct inference. Anchors from disconnected root domains retain their individual paths even when no shared ancestor is available.

B.3 Reasoning and Verification Details

This section describes the dual-verification configuration of Stage III, comprising scaffold-guided generation, execution checks, concept-membership checks, and targeted revision.

B.3.1 Topology-Guided Reasoning

Given a problem q and its textual scaffold S_q, the backbone model generates a primary solution $r_q^s = \mathcal{M}_\theta(q, S_q)$. The scaffold serves as an orientation aid: the model is instructed to ignore irrelevant concepts, prioritize the problem statement, and avoid mentioning the graph in its solution.

An unguided solution from the same backbone may provide auxiliary diagnostic evidence during revision. It does not participate in majority voting or direct final-answer selection.

B.3.2 Execution and Concept-Membership Verification

Execution Verification A post-hoc extractor converts explicit mathematical claims in the generated solution into structured executable checks. Each retained claim must include a source span found in the solution and a check conforming to the supported schema. The extractor is instructed not to correct claims or introduce new statements.

SymPy executes the extracted checks using 16 supported verification types. These include `equality`, `evaluate`, and `solve` for algebra; `derivative`, `integral`, and `limit` for calculus; `summation`, `binomial`, and `factorial` for series and combinatorial computations; `gcd`, `mod`, `is_prime`, and `factorint` for number-theoretic operations; and `matrix_det`, `matrix_eigenvalues`, and `matrix_equal` for basic linear algebra. Symbolic equalities are checked by simplifying expression differences, while solution sets and eigenvalues are compared without assuming a fixed ordering. For indefinite integrals, the claimed antiderivative is validated by differentiating it and checking whether the original integrand is recovered.

Concept-Membership Verification The allowed concept set is

$$\mathcal{V}_q = A_q \cup \mathrm{Nodes}(\Pi_q) \cup L_q \cup N_q. \tag{8}$$

Concepts associated with reasoning steps are matched against the scaffold using normalized names. If needed, they are grounded through the concept index and checked for QID membership in $\mathcal{U}_q$. Semantic matching is disabled.

Each concept receives one of three labels: `in_scaffold`, `off_scaffold`, or `ungrounded`. The latter two indicate missing scaffold membership or unresolved grounding, rather than mathematical incorrectness.

B.3.3 Targeted Revision

Let e_q^{exec} and e_q^{topo} denote the execution and concept-membership reports. In this configuration, revision is triggered when either report flags a failure:

$$\tau_q = \mathbb{I}\left[\mathrm{Fail}(e_q^{\mathrm{exec}}) \vee \mathrm{Fail}(e_q^{\mathrm{topo}})\right]. \tag{9}$$

When revision is triggered, the model receives the problem, scaffold, candidate solution, and diagnostic feedback, including source claims and execution results where available. It is instructed to address the flagged issues, preserve sound reasoning, and return a complete solution ending in the standard `Answer:` format. An unguided solution, when available, serves only as diagnostic context. If no revision is triggered, the primary solution is retained.

B.3.4 Final-Answer Extraction and Equivalence

The final answer is extracted from the retained or revised scaffold-guided solution. For AIME-style and MATH-500 evaluations, extraction first searches for the last `Answer:` line, then the last balanced `\boxed{...}` expression, and finally the last nonempty line.

Answer comparison uses normalized literal matching, component-wise comparison for structured answers, SymPy simplification, and numerical comparison with tolerance 10^{-6}. Conservative textual matching is used as a final fallback when symbolic comparison is inapplicable.

C Omni-MATH Data Setting and Sampling Protocol

C.1 Data Source and Evaluation Scope

To control the computational cost of both reasoning generation and process-level evaluation while preserving coverage across mathematical domains and difficulty levels, we construct a fixed evaluation subset of 500 problems from a local snapshot of Omni-MATH, referred to as **Omni-MATH-500**. The sampling population contains 4,428 records from `Omni-Math.jsonl`, corresponding to approximately 11.29% of the available instances. Therefore, all Omni-MATH results reported in this paper are based on this fixed subset rather than the full Omni-MATH benchmark.

Each original record contains the problem statement, reference solution, reference answer, mathematical domain annotation, difficulty score, and competition source. The sampled subset preserves these fields and additionally records the original position `original_idx` and subset identifier `sample_idx` to support instance tracing and paired comparison across methods. Problem statements and original annotations are not rewritten. Reference answers are used only for post-generation scoring, while reference solutions are not provided to the solving model.

C.2 Joint Stratified Sampling

We adopt proportional stratified random sampling over the joint space of *primary mathematical domain* and *difficulty interval*, rather than simple random sampling or equal-size sampling across domains. This design aims to preserve the domain and difficulty composition of the original dataset as closely as possible.

The original domain annotation may contain multiple hierarchical labels. For sampling purposes, we use the first annotation path and select the category immediately following *Mathematics* as the primary domain. For example, *Mathematics* $\rightarrow$ *Algebra* $\rightarrow$ *Other* is assigned to *Algebra*. This rule is used only to define a unique sampling stratum and does not imply that a problem belongs exclusively to that domain. All original domain labels are retained in the sampled records. Instances without a valid domain label are assigned to *Unknown*.

Difficulty values are taken directly from the original dataset and are not re-estimated using model accuracy, reasoning length, or any model-derived signal. We partition the difficulty scores into four non-overlapping intervals:

$$[1, 3], \qquad (3, 5], \qquad (5, 8], \qquad (8, 10]. \tag{10}$$

The observed difficulty range in the local source snapshot is 1.0 to 9.5. The joint domain–difficulty partition yields 30 non-empty strata.

Let stratum h contain N_h problems, with total population size $N = 4428$ and target sample size $n = 500$. The ideal proportional allocation for stratum h is

$$a_h = n\frac{N_h}{N}. \tag{11}$$

We first assign $\lfloor a_h \rfloor$ samples to each stratum and then distribute the remaining slots according to the fractional remainder $a_h - \lfloor a_h \rfloor$ in descending order. Ties are resolved by preferring the larger source stratum; if a tie remains, the order of first occurrence in the source file is retained. The final stratum quotas therefore sum exactly to 500.

Sampling within each stratum is performed without replacement using Python's standard pseudo-random number generator with a fixed random seed of 2026. Strata are processed in sorted order by domain name and difficulty-bin label while sharing the same random number generator. Selected records are then sorted by their original source positions and assigned `sample_idx` values starting from 1. Given the same source file, record ordering, and sampling rules, the resulting 500-instance subset is fully reproducible.

C.3 Sample Composition

Table 6 summarizes the number of source instances for each primary domain and the corresponding sampled distribution across difficulty intervals.

Table 7 compares the overall difficulty distribution of the source population and the sampled subset.

The final subset covers seven primary domains and 23 populated joint strata. No minimum quota of one instance per stratum is enforced. Consequently, extremely small strata may receive zero samples after proportional allocation and integer rounding. This occurs for the *Other* and *Unknown* categories, each of which contains only one source instance. These omissions result from the sampling rule rather than any manual filtering based on model performance.

C.4 Fixed-Subset Evaluation and Scoring

All compared methods are evaluated on the same fixed set of 500 problems. The subset is not resampled across methods, and no problem is replaced according to model correctness, concept-

Table 6: Domain and difficulty composition of the Omni-MATH-500 subset.

Primary Domain	Source	[1,3]	(3,5]	(5,8]	(8,10]	Sample Total
Algebra	1581	45	65	60	9	179
Applied Mathematics	733	19	50	14	0	83
Calculus	63	0	2	5	0	7
Discrete Mathematics	613	1	31	32	5	69
Geometry	792	10	51	28	1	90
Number Theory	594	4	31	27	5	67
Precalculus	50	0	2	3	0	5
Other	1	0	0	0	0	0
Unknown	1	0	0	0	0	0
Total	**4428**	**79**	**232**	**169**	**20**	**500**

Table 7: Difficulty distribution of the Omni-MATH source population and Omni-MATH-500.

Difficulty	Source	Source %	Sample	Sample %
$[1, 3]$	699	15.79	79	15.80
$(3, 5]$	2050	46.30	232	46.40
$(5, 8]$	1492	33.69	169	33.80
$(8, 10]$	187	4.22	20	4.00
Total	**4428**	**100.00**	**500**	**100.00**

grounding success, scaffold construction, or any other model-dependent outcome. The sampling procedure uses only the original domain annotations, difficulty values, and the fixed random seed.

For Omni-MATH-500, each method performs one complete run per problem. Each method therefore produces 500 final outputs. Self-Consistency@5 is an exception at the internal inference level: each problem generates five candidate solutions and returns one output through majority voting among the candidates. Reference answers do not participate in this voting process. Thus, one method-level run per problem does not imply identical model-call budgets across all compared methods.

Final-answer evaluation uses a local rule-based scoring procedure. The evaluator first extracts the final answer from the generated output and then applies formatting normalization and numerical or symbolic equivalence checks for selected textual answers. Process-level metrics are computed separately after reasoning generation by parsing the generated reasoning trajectories.

D Reasoning-Process Evaluation

D.1 Evaluation Protocol

Our reasoning-process evaluation builds on the logical-closeness framework introduced by DAG-MATH (Zhang et al., 2026). We use DeepSeek-flash to parse saved model solutions into atomic reasoning steps and their direct dependencies. The resulting reasoning graphs are processed by deterministic scripts to compute PRR, Closure-AUC, and Mean Closure. Final-answer correctness is obtained from the labels saved by the answer-evaluation procedure.

Dependency Annotation For each solution, the parser receives the target problem, candidate solution, and one reference annotation selected from `DAG-MATH-Formatted-CoT`. Reference selection uses available domain information and IDF-weighted lexical overlap, excluding references whose normalized problem statements exactly match the target problem. The reference illustrates the intended step granularity and direct-dependency representation. The parser is instructed to preserve the candidate's claims and logical order, identify the immediately required earlier steps for each claim, and include the stated final answer as the last step.

Graph Construction The parser returns a JSON object containing a list of steps, each with a `step_id`, `text`, and `direct_dependent_steps`. Each step becomes a node, and an edge $i \to j$ is added when step j lists step i as a direct dependency. Valid records contain at least

two steps with consecutive identifiers starting from 1. Dependency lists are normalized into sorted, unique references to earlier steps; invalid references are removed, and empty dependency lists are represented by `null`. The final step is designated as the final-answer node.

Parsing failures are handled through a format-repair attempt followed, when necessary, by a compact retry. Records that remain unresolved are retained for metric aggregation with a closure ratio of zero and their original answer-correctness labels.

D.2 Metric Definitions and Aggregation

We adopt the perfect-reasoning criterion and threshold-based evaluation approach of DAG-MATH, explicitly quantify partial closure using a closure ratio, and additionally report its mean across trajectories. Let N denote the total number of evaluated trajectory records, including unresolved parsing failures, and let $\mathrm{correct}_i$ indicate whether the final answer associated with record i is correct. PASS@1 is computed as the proportion of these records with correct final answers.

Structural Closure For a successfully parsed reasoning trajectory r_i, let V_i^- denote the set of all nodes except the final-answer node. A non-final node is considered closed when it is referenced by at least one subsequent reasoning step. We define the closure ratio as

$$\mathrm{Closure}(r_i) = \frac{\left|\left\{v \in V_i^- \mid \deg^+(v) > 0\right\}\right|}{|V_i^-|}, \tag{12}$$

where $\deg^+(v)$ is the outdegree of node v in the parsed graph. A trajectory is fully closed when $\mathrm{Closure}(r_i) = 1$, meaning that every non-final node is referenced by a later step. This implements the logical-closeness criterion of DAG-MATH on the parsed graph. Intermediate values quantify the proportion of non-final reasoning nodes referenced by subsequent steps.

Perfect Reasoning Rate (PRR) Following the perfect-reasoning criterion of DAG-MATH, PRR measures the proportion of trajectories that both produce a correct final answer and achieve full structural closure:

$$\mathrm{PRR} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}\left[\mathrm{correct}_i \wedge \mathrm{Closure}(r_i) = 1\right]. \tag{13}$$

Here, $\mathbb{I}[\cdot]$ is the indicator function. This empirical estimate adds a full-closure requirement to final-answer correctness.

Closure-AUC Following the threshold-based evaluation approach of DAG-MATH, we evaluate answer correctness under increasingly strict closure requirements. For a threshold $\tau \in [0, 1]$, the threshold-specific score is

$$A(\tau) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}\left[\mathrm{correct}_i \wedge \mathrm{Closure}(r_i) \geq \tau\right]. \tag{14}$$

The resulting accuracy–closure curve starts at $A(0) = \mathrm{PASS@1}$ and ends at $A(1) = \mathrm{PRR}$. Our implementation reports the arithmetic mean over 101 uniformly spaced thresholds as Closure-AUC:

$$\text{Closure-AUC} = \frac{1}{101}\sum_{k=0}^{100} A\left(\frac{k}{100}\right). \tag{15}$$

The thresholds range from 0 to 1 with an interval of 0.01. A higher score indicates a higher average proportion of answer-correct trajectories satisfying the specified structural-closure requirements.

Mean Closure We additionally report Mean Closure to summarize structural closure independently of final-answer correctness:

$$\text{Mean Closure} = \frac{1}{N}\sum_{i=1}^{N} \mathrm{Closure}(r_i). \tag{16}$$

This metric averages closure ratios across trajectory records rather than pooling individual reasoning nodes. It therefore assigns equal weight to each evaluated trajectory, regardless of its number of steps or final-answer correctness.

Aggregation and Parsing Failures All evaluated records enter the aggregation denominator. An unresolved parsing failure has a closure ratio of zero and contributes zero to PRR and Mean Closure. Its stored answer-correctness label is preserved: when the final answer is correct, the record satisfies the zero-threshold condition and contributes only at that threshold to Closure-AUC. Parsed annotations and per-record statistics are saved alongside the aggregate results.

D.3 DEPENDENCY-ANNOTATION PROMPT

The following template is supplied as a single user message, without a system message. Placeholders are populated with the selected reference metadata, reference problem, annotation, target problem, and candidate solution.

```
You are an external mathematical dependency annotator.
Your task is to convert the CANDIDATE SOLUTION into a directed acyclic
graph of atomic reasoning steps. The REFERENCE DAG is a gold-standard
annotation from the DAG-MATH dataset derived from Omni-MATH. Use it only
to learn the intended granularity and the meaning of direct dependencies.

Do not solve the target problem. Do not correct, complete, or improve the
candidate. Preserve incorrect claims and omitted justifications exactly
as properties of the candidate reasoning. Do not transfer mathematical
content from the unrelated reference problem.

Annotation rules:
- Each step contains one atomic claim faithfully copied or condensed from
  the candidate solution.
- Preserve the candidate's logical order.
- direct_dependent_steps lists only the minimal immediately required
  earlier steps, not all ancestors.
- Use null when the claim comes directly from the target problem or is
  invoked as standalone background knowledge.
- Every dependency must refer to a smaller step_id.
- Include the candidate's stated final answer as the final step.
- Do not invent dependencies merely to make the graph closed.

Return one English JSON object only:
{
  "steps": [
    {
      "step_id": 1,
      "text": "one atomic claim from the candidate",
      "direct_dependent_steps": null
    }
  ]
}

[GOLD REFERENCE METADATA]
problem_id: {reference_problem_id}
sample_id: {reference_sample_id}
domain: {reference_domain}

[GOLD REFERENCE PROBLEM]
{reference_problem}

[GOLD REFERENCE DAG]
{reference_dag}

[TARGET PROBLEM]
{problem}

[CANDIDATE SOLUTION]
{solution}
```

Table 8: Summary of the evaluated inference-time methods and their external execution tools.

Method	Primary Mechanism	External Tool
CoT-ZeroShot	Zero-shot elicitation of step-by-step reasoning.	None
Self-Consistency@5	Five sampled reasoning trajectories with answer majority voting.	None
PAL	Program-based intermediate reasoning with interpreter execution.	Python
Least-to-Most	Sequential subproblem solving using earlier intermediate answers.	None
Code + Text + Sum.	Separate textual and code-based solutions followed by LLM synthesis.	Code interpreter
SBSC	Successive subtask and program generation conditioned on previous execution outputs.	Code interpreter
CoMAT	Symbolic problem conversion followed by reasoning within the LLM.	None
TOPOMATH	Problem-specific concept scaffolding with verification-guided revision.	SymPy

E BASELINE CONFIGURATIONS

The reproduced baselines cover complementary approaches to inference-time mathematical reasoning. Their mechanisms are described below and summarized in Table 8.

CoT-ZeroShot CoT-ZeroShot (Kojima et al., 2022) prompts the backbone model to generate intermediate reasoning steps before producing the final answer. It uses a zero-shot reasoning instruction without task-specific demonstrations or external execution tools.

Self-Consistency@5 Self-Consistency@5 (Wang et al., 2023) samples five reasoning trajectories for each problem and selects the final answer through majority voting among the resulting answers. Its inference mechanism aggregates multiple candidate answers rather than relying on a single generated trajectory.

Program-Aided Language Models (PAL) PAL (Gao et al., 2023) uses the language model to represent intermediate reasoning as executable programs. The generated code is executed by a Python interpreter, which performs the computations used to obtain the final answer.

Least-to-Most Least-to-Most prompting (Zhou et al., 2023) first decomposes a complex problem into simpler subproblems and then solves them sequentially. Answers obtained for earlier subproblems are included in the context for later ones, allowing intermediate results to support subsequent reasoning.

Code + Text + Sum. We use the Code + Text + Summarization variant from Chen et al. (2025), abbreviated as Code + Text + Sum. This method obtains separate textual and code-based solutions, with generated code evaluated by an interpreter. The language model then combines and summarizes the two outputs into a final response.

Step-by-Step Coding (SBSC) SBSC (Singh et al., 2025) performs mathematical reasoning through successive subtasks and executable programs. At each turn, the model generates an intermediate subtask and a corresponding program, using previous programs and execution outputs as context. The process proceeds through multiple coding turns to obtain the final answer.

Chain of Mathematically Annotated Thought (CoMAT) CoMAT (Leang et al., 2025) separates problem solving into symbolic conversion and reasoning execution. It first converts the natural-language problem into a structured mathematical representation and then performs step-by-step reasoning over that representation. Both stages are carried out within the language model without an external symbolic solver.

F ADDITIONAL EXPERIMENTAL RESULTS

This appendix describes the reproduced inference-time baselines and reports complete experimental results across three backbone models: DeepSeekV4-Flash, Qwen3.5-27B, and Qwen3-8B. The main method comparison evaluates all seven inference-time baselines and TOPOMATH using DeepSeekV4-Flash without thinking mode. The additional backbone experiments compare direct

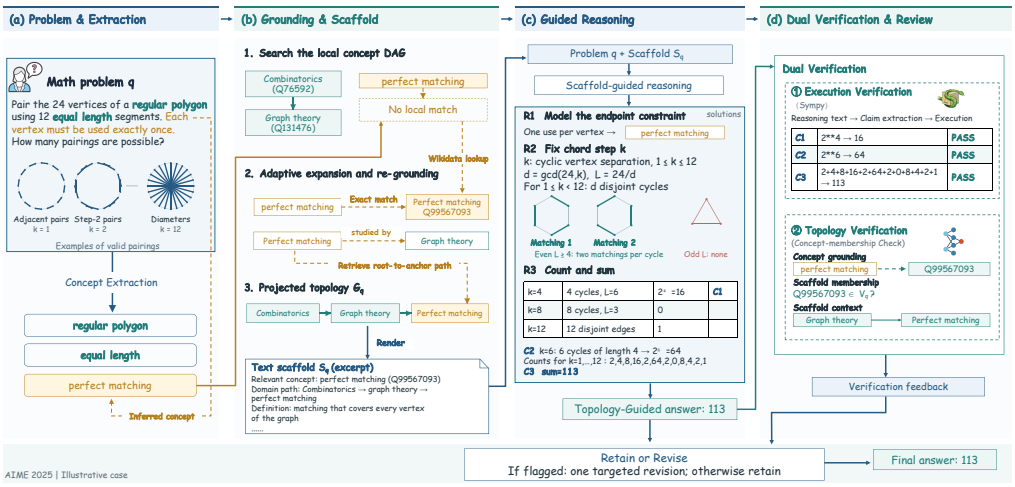

Figure 4: Adaptive concept expansion and scaffold-guided reasoning on an AIME 2025 problem.

inference with TopoMath separately on Qwen3.5-27B and Qwen3-8B across all six benchmarks, focusing on within-backbone performance changes. We also present a case study in Figure 4 illustrating adaptive concept expansion, scaffold-guided reasoning, and dual verification on an AIME 2025 problem.

F.1 Case Study

Figure 4 illustrates adaptive concept expansion and scaffold-guided reasoning on an AIME 2025 problem that pairs the vertices of a regular 24-gon using equal-length segments. From the requirement that each vertex be used exactly once, the concept extractor infers *perfect matching*, alongside the explicitly stated concepts of regular polygon and equal length. Since perfect matching is initially unmatched in the working topology, adaptive expansion retrieves its Wikidata entity and connects it to graph theory through a *studied-by* relation. The resulting path from combinatorics through graph theory to perfect matching is rendered as part of the textual scaffold S_q. The guided solution then models the admissible pairings as perfect matchings in a graph defined by a fixed cyclic vertex separation k. For $1 \le k < 12$, this graph decomposes into $d = \gcd(24, k)$ cycles of length $L = 24/d$. Each even cycle admits two perfect matchings, giving 2^d pairings, whereas odd cycles admit none. The diameter case $k = 12$ contributes one pairing, and summing across all step sizes yields 113. Execution verification checks the selected arithmetic claims, while concept-membership verification checks whether the invoked perfect-matching concept is supported by the scaffold. The resulting feedback informs the retain-or-revise decision. This case illustrates how a concept inferred from the problem can be grounded through adaptive expansion, organized into a problem-specific scaffold, and used in subsequent reasoning and diagnostic verification.

F.2 Complete Experimental Results

Table 9 reports the experimental results of Qwen3.5-27B with and without TopoMath on the six evaluation benchmarks, including both final-answer accuracy and process-level reasoning quality.

Table 10 reports the experimental results for Qwen3-8B with and without TopoMath on the six evaluation benchmarks, including both final-answer accuracy and process-level reasoning quality.

Table 9: Answer correctness and process-level reasoning quality with Qwen3.5-27B as the backbone on six mathematical reasoning benchmarks. All scores are reported on a scale of $[0, 1]$. The higher score for each dataset–metric pair is shown in bold.

Dataset	Method	PASS@1	PRR	Closure-AUC	Mean Closure
AIME 2025	Qwen3.5-27B	0.333	0.080	0.305	0.847
	TopoMath	**0.667**	**0.370**	**0.643**	**0.943**
AIME 2026	Qwen3.5-27B	0.400	0.070	0.367	0.859
	TopoMath	**0.787**	**0.430**	**0.759**	**0.958**
BRUMO 2025	Qwen3.5-27B	0.470	0.177	0.435	0.855
	TopoMath	**0.750**	**0.467**	**0.724**	**0.949**
HMMT 2026	Qwen3.5-27B	0.242	0.061	0.227	0.836
	TopoMath	**0.458**	**0.218**	**0.437**	**0.927**
MATH-500	Qwen3.5-27B	0.924	0.386	0.868	0.934
	TopoMath	**0.952**	**0.666**	**0.919**	**0.966**
Omni-500	Qwen3.5-27B	0.386	0.104	0.348	0.831
	TopoMath	**0.586**	**0.334**	**0.560**	**0.938**

Table 10: Answer correctness and process-level reasoning quality with Qwen3-8B as the backbone on six mathematical reasoning benchmarks. All scores are reported on a scale of $[0, 1]$. The higher score for each dataset–metric pair is shown in bold.

Dataset	Method	PASS@1	PRR	Closure-AUC	Mean Closure
AIME 2025	Qwen3-8B	0.200	0.123	0.195	0.901
	TopoMath	**0.213**	**0.180**	**0.208**	**0.931**
AIME 2026	Qwen3-8B	0.123	0.103	0.120	0.907
	TopoMath	**0.173**	**0.133**	**0.168**	**0.947**
BRUMO 2025	Qwen3-8B	0.320	0.120	0.305	0.897
	TopoMath	**0.350**	**0.273**	**0.334**	**0.922**
HMMT 2026	Qwen3-8B	0.130	0.042	0.117	0.830
	TopoMath	**0.155**	**0.121**	**0.150**	**0.937**
MATH-500	Qwen3-8B	0.798	0.464	0.754	0.930
	TopoMath	**0.828**	**0.720**	**0.811**	**0.975**
Omni-MATH-500	Qwen3-8B	0.326	0.124	0.296	0.863
	TopoMath	**0.354**	**0.262**	**0.338**	**0.939**