Adaptive Tournament Answer Pooling for Test-Time Scaling

Terry Ma*
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
terryma@cs.cmu.edu

Aidan Zhang
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
aidanz@andrew.cmu.edu

Abstract

Self-consistency improves large language model (LLM) reasoning by sampling multiple solution traces and aggregating their final answers. However, on difficult problems with highly branching reasoning, majority voting and confidence-weighted voting can fail even when the correct answer has already been sampled. In these answer-present selection failures, the correct solution appears in the pool but is suppressed by a more frequent or higher-confidence spurious answer.

We introduce **Adaptive Tournament Answer Pooling (ATAP)**, a lightweight, training-free, inference-only framework for rescuing correct-minority answers. ATAP groups traces by normalized final answer, synthesizes the supporting traces for each candidate into an answer-level rationale, and adjudicates among candidate answers using tournament-style pairwise voting. ATAP focuses verification on contested choices where frequency-based aggregation is unreliable. Settled pools are handled by cheap consensus, while tournament rescue is applied to ambiguous pools. On challenging math and reasoning benchmarks including AIME, HMMT, and GPQA Diamond, ATAP improves over majority voting and confidence-weighted self-consistency, especially on high-entropy instances where correct minority answers are most likely to be suppressed.

1 Introduction

Large language models (LLMs) have demonstrated strong reasoning capabilities that can be further enhanced by scaling inference-time compute. A widely used inference-time strategy is self-consistency, where the model samples multiple reasoning traces and aggregates their final answers, typically by majority vote [1]. The underlying intuition is that multiple reasoning attempts reduce the variance of a single sample, so repeated convergence to the same answer provides evidence that the answer is reliable.

This frequency-based view is effective when the sampled answer distribution is sharply peaked, as in easier problems where most traces converge to the same solution. However, difficult reasoning problems often induce substantial branching: different traces may explore different decompositions, case splits, algebraic transformations, or intermediate assumptions, leading to a diffuse distribution over final answers. In this regime, the majority answer may not correspond to the correct answer. Crucially, the failure is not always that the model never found the solution: the correct answer may already appear in the sampled pool, but remain a low-frequency candidate overwhelmed by a more common spurious answer. We refer to these cases as *answer-present selection failures*, where the bottleneck is not answer discovery but answer selection among competing hypotheses.

*Use footnote for providing further information about author (webpage, alternative address)—*not* for acknowledging funding agencies.

Preprint.

This limitation of frequency-based aggregation has been recognized in recent work. AoR observes that answer-frequency ensembling can fail when correct answers are in the minority and proposes a hierarchical aggregation framework based on reasoning-chain evaluation [2]. AggLM similarly treats solution aggregation as a learned reasoning skill, training an aggregator to reconcile candidate solutions and recover minority-but-correct answers [3]. In parallel, adaptive self-consistency methods use sample agreement, confidence, or answer-pool uncertainty to allocate test-time sampling budgets or stop early on settled instances [4, 5]. These works suggest two complementary lessons: answer frequency alone can be unreliable on difficult reasoning problems, and answer-pool statistics provide useful model-relative signals about uncertainty.

Our work builds on these observations but focuses on a targeted rescue question: once a sampled pool contains multiple competing answers, how should we spend limited adjudication compute to recover an under-supported correct candidate? Rather than applying judge-based comparison to every answer, we argue that adjudication should operate on grouped answers. Traces that reach the same final answer often contain redundant evidence, while the real selection problem is to decide which final-answer hypothesis is best supported. This motivates an answer-level tournament: collapse traces by normalized final answer, synthesize the support for each candidate, and compare the canonicalized evidence rather than every summary.

We introduce **Adaptive Tournament Answer Pooling (ATAP)**, a training-free inference procedure that combines cheap consensus with targeted answer-level rescue. ATAP first samples a warm-up pool of reasoning traces and computes statistics of the induced answer distribution. Settled pools are handled by majority or confidence-weighted consensus. Contested warm-up pools enter a higher-budget route with additional sampling and stability-based filtering. Normalized examples then undergo answer-level tournament rescue over the top retained answer clusters.

In rescue mode, ATAP groups traces by their normalized final answer, compresses each answer group into a representative rationale, and runs a seeded tournament over the resulting answer-level hypotheses. This design treats judge calls as a scarce selection resource. A tournament over raw traces can spend many comparisons on redundant solutions that reach the same final answer, and a noisy early comparison can eliminate a correct trace before later rounds. ATAP instead compares a smaller set of candidate answers, focusing adjudication on the regime where consensus is least reliable: high-entropy answer pools in which the correct answer may have been generated but remains under-supported by frequency or confidence-weighted voting.

A detailed pipeline of ATAP is illustrated in Figure 1. On challenging competitive math and reasoning benchmarks such as AIME, HMMT, and GPQA Diamond, ATAP improves over fixed self-consistency and confidence-weighted aggregation, with the largest gains on high-entropy instances where standard aggregation is most likely to suppress correct minority answers.

2 Related Work

Self-consistency and adaptive test-time sampling. Self-consistency improves reasoning by sampling multiple chains of thought and selecting the most frequent final answer [1]. Because fixed-budget sampling can be expensive, subsequent work has explored adaptive allocation of inference-time compute. Adaptive-Consistency dynamically adjusts the number of samples per question using a lightweight stopping criterion based on agreement among generated samples [4]. ESC similarly reduces the cost of self-consistency by stopping once a sufficiently confident answer window has emerged [5]. Confidence-based methods use model probabilities, verbalized confidence, or stability scores to weight votes, prune low-confidence traces, or halt generation [6–8]. Our implementation uses the same token-level stability instantiation as DeepConf [6] for retained-pool construction and online early stopping, followed by a separate answer-level tournament rescue stage; details are given in Appendix B.4.

ATAP additionally uses answer-pool statistics to separate routing from answer-level adjudication. Low-uncertainty warm-up pools are handled by cheap consensus, while high-uncertainty warm-up pools enter a higher-budget route followed by tournament rescue over the top retained answer clusters.

Answer selection and correct-minority aggregation. Majority voting can fail when the correct answer appears only as a minority candidate. AoR identifies this limitation and argues that answer-frequency aggregation is insufficient because the relevant evidence may lie in the reasoning chains

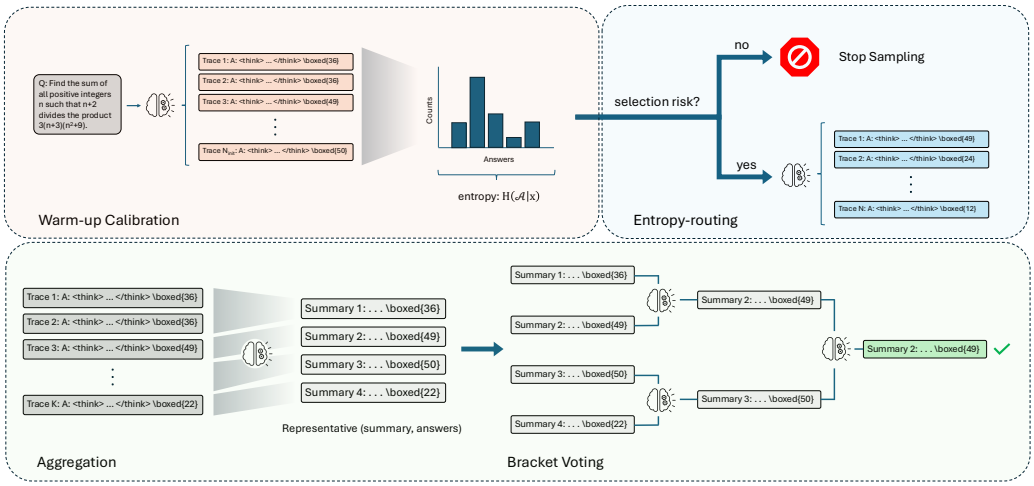

Figure 1: **Framework overview:** ATAP begins by sampling N_{init} warm-up traces and computing answer entropy over the valid warm-up answer distribution. An inference router either stops early and returns a consensus answer when the warm-up pool is settled, or samples additional traces when uncertainty remains. For examples routed to the higher-budget regime, ATAP aggregates retained traces by final answer, summarizes representative reasoning for each candidate, and applies bracket-style tournament adjudication to select the final answer.

rather than the final answer counts alone [2]. It proposes a hierarchical reasoning aggregation framework that evaluates sampled reasoning chains and incorporates dynamic sampling. AggLM instead treats aggregation as an explicit learned skill, training an aggregator with reinforcement learning from verifiable rewards to review, reconcile, and synthesize candidate solutions, including cases where the correct answer is in the minority [3].

Our work shares the goal of improving answer selection in correct-minority regimes, but differs in its assumptions and mechanism. ATAP is training-free and does not require learning a new aggregator. It also performs routed answer-level adjudication: traces are first grouped by final answer, then summarized into candidate-specific rationales, and only then compared by a judge. This separates the question of *whether* adjudication is warranted from the question of *which* answer hypothesis should win.

Judge-based aggregation, revision, and tournament selection. A separate line of work uses LLMs as judges, verifiers, or revisers to select among candidate solutions, aggregate rationales, or refine sampled traces. Universal Self-Consistency (USC) uses the model itself to select the most consistent answer among sampled candidates, extending self-consistency beyond settings where exact answer extraction and majority voting are straightforward [9]. Other work similarly studies judge-based or generative selection procedures for choosing among candidate outputs [10, 11]. Pairwise and tournament-style selection methods can improve over simple answer counting by comparing candidate solutions more directly [12]. However, judge-based comparison also introduces new failure modes: comparisons can be sensitive to position, verbosity, self-preference, and other evaluation biases [13, 14]. In addition, trace-level tournaments may waste judge calls comparing multiple traces that support the same final answer.

ATAP uses judge-based comparison only after routing and only at the answer level. By collapsing redundant traces into a single representative packet per candidate answer, ATAP reduces unnecessary comparisons and focuses judge compute on competing hypotheses rather than individual sampled traces. Appendix C lists the prompt templates used for packet synthesis and tournament judging.

3 Preliminaries

3.1 Self-Consistency as Answer Selection

Let x be an input problem with correct final answer $a^\star$. A stochastic reasoning model generates a trace $\tau \sim \pi(\cdot \mid x)$, from which an answer extractor returns a normalized final answer $a = g(\tau)$. Given a sampling budget n, self-consistency draws traces

$$\mathcal{T}_n = \{\tau_i\}_{i=1}^n, \qquad a_i = g(\tau_i).$$

Because self-consistency aggregates final answers rather than full reasoning traces, we summarize the sampled pool by answer counts. Let $\mathcal{U}_n$ be the set of unique answers among $\{a_i\}_{i=1}^n$ with corresponding supports N_j and empirical frequencies $\hat{p}_j$:

$$\mathcal{U}_n = \{u_1, \ldots, u_m\}, \quad N_j = \big|\{i : a_i = u_j\}\big|, \quad \hat{p}_j = \frac{N_j}{\sum_{\ell=1}^m N_\ell},$$

where we further restrict $\mathcal{U}_n$ to non-empty, valid answers ($u_i \neq \varnothing$). Standard self-consistency returns the most frequent sampled answer:

$$\hat{a}_{\mathrm{MV}} = u_{j^\star}, \qquad j^\star = \arg\max_{j \in [m]} N_j.$$

This estimator is effective when the correct answer is the dominant mode of the model's sampled answer distribution. However, as noted in prior works, majority voting can fail even when the model has already generated a correct trace: the correct answer may appear in the pool but receive less support than a coherent wrong answer.

3.2 Answer Discovery and Answer Selection Regimes

To organize our analysis, we separate failures of candidate generation from failures of candidate selection. An *answer discovery failure* occurs when no sampled trace produces the correct answer:

$$a^\star \notin \mathcal{U}_n.$$

In this case, no aggregation rule restricted to the sampled candidates can recover $a^\star$, requiring a stronger model, additional trace sampling, or external information to improve.

An *answer selection failure* occurs when the correct answer is present in the candidate pool but the aggregation rule selects a different answer:

$$a^\star \in \mathcal{U}_n \quad \text{and} \quad \hat{a} \neq a^\star.$$

Our method focuses on this answer-present regime: cases where the sampled pool already contains the correct answer, but the current aggregation rule mis-ranks it relative to competing hypotheses.

A common instance is the *correct-minority* case:

$$a^\star \in \mathcal{U}_n, \qquad N(a^\star) < \max_{u \in \mathcal{U}_n,\, u \neq a^\star} N(u),$$

where $N(u) = |\{i : a_i = u\}|$ denotes the support of answer u. Majority voting cannot recover the correct answer on such examples without additional information beyond raw frequency. Although confidence-weighted voting can help when correct traces receive higher stability scores, it can still fail when the correct answer has lower aggregate weight than an incorrect competing answer.

3.3 Answer-Pool Statistics

We use simple statistics of the sampled answer distribution to estimate whether additional sampling and answer-level adjudication are likely to be useful. Let $\mathcal{A}_n$ denote the multiset of valid normalized answers extracted from the sampled traces, after clustering mathematically equivalent answers. Let $U_n = \{u_1, \ldots, u_m\}$ be the set of distinct answer clusters, with counts

$$N_j = |\{i : g(\tau_i) = u_j\}|, \qquad \hat{p}_j = \frac{N_j}{\sum_{\ell=1}^m N_\ell}.$$

We compute the Shannon entropy of the clustered valid answer distribution, measured in bits:

$$H(U_n) = -\sum_{j=1}^{m} \hat{p}_j \log_2 \hat{p}_j.$$

When there is only one valid answer cluster, this entropy is zero. If no valid answer is extracted, we treat the pool as maximally uncertain and route the instance to the higher-budget regime.

Low entropy indicates that the model repeatedly returns the same answer, while high entropy indicates a contested pool with multiple plausible answer hypotheses. As in adaptive self-consistency methods, these quantities should be interpreted as model-relative diagnostics rather than intrinsic measures of problem difficulty: a low-entropy pool may reflect either an easy problem solved consistently or a hard problem on which the model confidently collapses to the same wrong answer.

In our setting, answer-pool entropy is used as a lightweight routing signal. Concentrated warm-up pools are routed to cheap consensus, while dispersed pools enter the higher-budget route, where additional sampling and answer-level rescue may be useful.

In our implementation, each trace τ_i receives a scalar stability score s_i using the same token-level confidence instantiation as DeepConf [6]; see Appendix B.4 for details. Entropy is computed only over valid normalized answers. The stability score is then used in the higher-budget regime for online early stopping, conservative trace retention, confidence-weighted support, and candidate seeding. Invalid-answer traces are excluded from both entropy and support calculations; additional implementation details are given in Appendix B.2.

4 Adaptive Tournament Answer Pooling

4.1 Overview

We propose ATAP, a training-free inference procedure that decides when to trust cheap consensus and when to spend additional compute on answer-level adjudication. The method has three stages:

1. cheap warm-up sampling of reasoning traces;
2. warm-up routing conditioned on answer-pool entropy;
3. answer-level tournament adjudication over competing candidate answers

The goal is not to replace self-consistency with tournament judging on every problem. Instead, ATAP uses the warm-up distribution to identify cases where consensus appears reliable, and reserves adjudication for contested pools where selection failures are more likely. In our experiments, the higher-budget route uses stability for retained-pool construction and online early stopping, after which ATAP applies answer-level rescue over the top retained answer clusters. This routing step is central: the tournament is treated as a targeted rescue mechanism rather than a default aggregation rule. Algorithms 1 and 2 summarize the routing/retention stage and the consensus/rescue stage, respectively. Appendix B gives the implementation details and Appendix C lists the prompts used for synthesis and judging.

4.2 Warm-Up Sampling

For each problem x, we first sample n_0 warm-up traces

$$\mathcal{T}_0 = \{\tau_i\}_{i=1}^{n_0}, \qquad a_i = g(\tau_i).$$

The extracted answers are normalized by g and summarized as a set of distinct valid answers $\mathcal{U}_0$. From the empirical distribution over these answers, we compute the warm-up answer entropy $H(\mathcal{U}_0)$.

Each warm-up trace may also receive a reliability score s_i. Our experiments instantiate s_i with DeepConf-style stability, but the framework only requires a completed-trace score for filtering, confidence-weighted support, and candidate ranking. Online early stopping additionally requires an online version of that score that can be computed during generation. We keep the main text abstract and describe the concrete entropy estimator, stability score, and filtering rule in Appendix B.

4.3 Entropy Routing

The router maps warm-up answer-pool statistics to one of two inference modes:

$$r(x) \in \{\text{consensus}, \text{higher-budget}\}.$$

A simple implementation uses the warm-up answer entropy, entering the higher-budget route when the pool is sufficiently dispersed:

$$r(x) = \begin{cases} \text{higher-budget}, & H(\mathcal{U}_0) > c, \\ \text{consensus}, & H(\mathcal{U}_0) \leq c. \end{cases}$$

This routing rule captures the main intuition: when the warm-up pool is concentrated, additional adjudication is unlikely to change the answer; when the pool is split across competing answers, the method should spend more compute constructing a stronger retained pool.

4.4 Consensus Mode

In consensus mode, ATAP returns the leading answer from the retained pool. The default rule is majority vote:

$$\hat{a}_{\text{consensus}} = \arg \max_{u \in \mathcal{U}(\mathcal{P})} N(u),$$

where $\mathcal{P}$ is the retained trace pool, $\mathcal{U}(\mathcal{P})$ is the set of distinct valid answers in that pool, and $N(u) = |\{\tau_i \in \mathcal{P} : g(\tau_i) = u\}|$.

If trace confidence scores are available, we instead use confidence-weighted support:

$$W(u) = \sum_{\tau_i \in \mathcal{P}: g(\tau_i) = u} s_i, \qquad \hat{a}_{\text{consensus}} = \arg \max_{u \in \mathcal{U}(\mathcal{P})} W(u).$$

Consensus mode is deliberately cheap: it avoids spending judge or synthesis calls on answer pools that can be resolved by direct aggregation.

4.5 Rescue Mode

In contrast, rescue mode is intended for examples with sufficiently dispersed warm-up answer distributions at risk for answer selection failure.

Specifically, the warm-up-derived retained pool $\mathcal{P}$ is augmented with additional sampled traces up to a fixed budget B. From this augmented pool, ATAP selects a small candidate set $\mathcal{V} \subseteq \mathcal{U}(\mathcal{P})$, ranked by cumulative stability-weighted support. In our experiments we use $K = 8$ retained answer clusters; the appendix ablation in Figure 4 shows that ground-truth retention largely saturates near this value. For each candidate answer $u \in \mathcal{V}$, we collect its supporting traces

$$\mathcal{T}(u) = \{\tau_i \in \mathcal{P} : g(\tau_i) = u\},$$

and synthesize them into an answer-level packet

$$R(u) = \text{Summarize}(x, u, \mathcal{T}(u)).$$

Each packet contains the candidate answer and the main reasoning evidence supporting it, collapsing redundant traces that reach the same final answer.

The final rescue answer is obtained by a pairwise tournament over these answer-level packets:

$$\hat{a}_{\text{rescue}} = \text{Tournament}\big(\{(u, R(u)) : u \in \mathcal{V}\}, J\big).$$

We seed tournament candidates using their cumulative stability-weighted support, in which high-support candidates are paired against low-support candidates in the initial bracket. In our implementation, each match is judged using repeated verifier samples; the candidate with majority judge support advances, and ties are broken in favor of the higher-seeded candidate; prompt templates are listed in Appendix C.

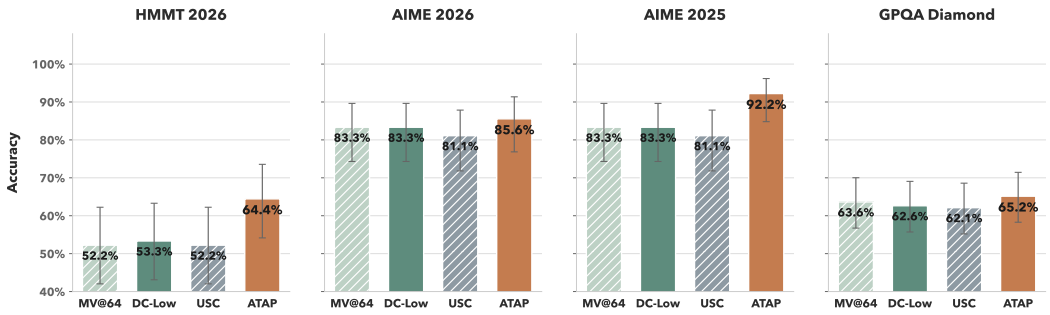

Figure 2: Mean accuracy across benchmarks. For HMMT 2026, AIME 2026, and AIME 2025, accuracies are averaged over three random seeds; GPQA Diamond is run with one seed. MV@64 denotes simple majority voting over 64 traces. DeepConf-Low refers to DeepConf [6] with 10% filtering. Error bars denote 95% Wilson confidence intervals for the reported accuracy estimates.

5 Results

We evaluate adaptive test-time compute pipelines on competition mathematics and multiple-choice scientific reasoning benchmarks. All experiments use DeepSeek-R1-0528-Qwen3-8B as the base model [15]. Our evaluation includes HMMT February 2026 subject tests [16], AIME 2025 and AIME 2026 from the American Invitational Mathematics Examination series [17], and GPQA Diamond [18].

For tournament adjudication, we use the base model as the judge. Each pairwise match is evaluated with three independent judge calls, and the candidate preferred by the majority advances. The tournament packet contains the top eight candidate answers.

Implementation details are summarized in Appendix B.2. Solver traces are sampled with temperature 0.6, top-$p = 0.95$, top-20 log probabilities, and maximum generation length 131072. The controller uses $n_0 = 16$, $B = 64$, $h_{\text{stop}} = 0.5$, and $\eta = 0.1$. Tournament packets contain the top $K = 8$ retained answer clusters ranked by stability-weighted support.

Figure 2 reports the main accuracy results. Error bars denote 95% Wilson confidence intervals for accuracy. For the math benchmarks, accuracies are averaged over three inference seeds; GPQA Diamond is reported from a single seed. We compare ATAP against majority voting over 64 sampled traces (MV@64), DeepConf-Low (DC-Low), and Universal Self-Consistency (USC). Across all evaluated benchmarks, ATAP outperforms each baseline. The largest gains occur on HMMT 2026 and AIME 2025. On HMMT 2026, ATAP reaches 64.4%, compared with 52.2% for MV@64, 53.3% for DC-Low, and 52.2% for USC. On AIME 2025, ATAP improves to 92.2%, compared with 83.3% for MV@64 and DC-Low and 81.1% for USC. Gains on AIME 2026 are smaller, with ATAP reaching 85.6% versus 83.3% for MV@64 and DC-Low and 81.1% for USC. On GPQA Diamond, ATAP obtains a modest improvement, reaching 65.2% compared with 63.6% for MV@64, 62.6% for DC-Low, and 62.1% for USC.

The smaller improvement on AIME 2026 reflects limited headroom, the baseline already solves 25.0 of 30 problems on average. Remaining errors include both candidate-coverage failures and answer-present cases where tournament judging does not recover the correct answer.

5.1 Efficiency

We next analyze the compute cost of ATAP. Table 1 compares fixed 64-trace parallel sampling, ATAP without tournament judging, and the full ATAP pipeline. All wall-clock measurements are collected on a single NVIDIA H200 GPU under the same serving setup.

The adaptive controller reduces solver-side compute by stopping low-entropy instances after the warm-up stage and allocating additional traces only to more contested examples. Without tournament judging, ATAP reduces token usage on all benchmarks: from 63.6M to 51.4M on HMMT 2026, from 50.8M to 36.0M on AIME 2026, from 52.2M to 38.5M on AIME 2025, and from 156.7M to 99.0M on GPQA Diamond. These correspond to token reductions of 19.2%, 29.1%, 26.2%, and 36.8%, respectively.

Table 1: Compute cost across adaptive inference configurations from a chosen representative seed. ATAP w/o Tournament refers to using ATAP without the final tournament round.

Benchmark	Approach	Traces	Tokens	Wall Clock Time
HMMT 2026	64-Trace Parallel Sampling	1,920	63.6M	18.5h
HMMT 2026	ATAP w/o Tournament	1,392	51.4M	17.5h
HMMT 2026	ATAP	1,749	58.4M	22.6h
AIME 2026	64-Trace Parallel Sampling	1,920	50.8M	13.6h
AIME 2026	ATAP w/o Tournament	1,104	36.0M	10.9h
AIME 2026	ATAP	1,253	37.8M	12.9h
AIME 2025	64-Trace Parallel Sampling	1,920	52.2M	9.8h
AIME 2025	ATAP w/o Tournament	1,104	38.5M	7.2h
AIME 2025	ATAP	1,385	44.0M	11.4h
GPQA Diamond	64-Trace Parallel Sampling	12,672	156.7M	49.8h
GPQA Diamond	ATAP w/o Tournament	8,297	99.0M	27.8h
GPQA Diamond	ATAP	9,496	114.2M	36.9h

Table 2: Math benchmark values are averaged over three seeds; GPQA Diamond is reported for one seed. GT denotes cases where the ground-truth answer appears in the retained candidate pool before tournament judging. Fractional values arise from averaging over seeds.

Benchmark	n	Baseline	GT	Rescued	Missed	Regress	GT Absent
HMMT 2026	30	16.0	21.3	3.3	2.0	0.0	8.7
AIME 2026	30	25.0	26.3	1.0	0.3	0.3	3.7
AIME 2025	30	25.0	27.7	2.7	0.0	0.0	2.3
GPQA Diamond	198	124	180	20	36	15	18

Adding tournament rescue increases cost relative to ATAP without the final tournament stage, since unsettled examples require repeated pairwise judge calls. Nevertheless, the full pipeline remains below the token cost of fixed 64-trace sampling on every benchmark. Full ATAP uses 58.4M tokens on HMMT 2026, 37.8M on AIME 2026, 44.0M on AIME 2025, and 114.2M on GPQA Diamond, corresponding to reductions of 8.2%, 25.6%, 15.7%, and 27.1% relative to fixed sampling.

Wall-clock time shows the tradeoff between reduced token generation and sequential adjudication. ATAP without tournament is faster than fixed sampling on all benchmarks. The full pipeline is also faster on AIME 2026 and GPQA Diamond, but slower on HMMT 2026 and AIME 2025 despite using fewer tokens. This reflects the sequential structure of synthesis and pairwise tournament judging, which can increase latency even when total token usage decreases.

5.2 Tournament judging improves answer-present failures

On the math benchmarks, tournament judging primarily helps in the intended answer-present regime. On HMMT 2026, the ground truth appears in 21.3 of 30 pools on average; the baseline solves 16.0 problems, while the tournament rescues 3.3 additional cases with no regressions. The AIME benchmarks have less headroom because the baseline already solves 25.0 of 30 problems. Even so, the tournament rescues 1.0 cases on AIME 2026 and 2.7 on AIME 2025, with near-zero regressions. GPQA Diamond shows both the promise and limitation of this approach. The ground truth is present in 180 of 198 pools, and the tournament rescues 20 baseline errors. However, it also misses 36 answer-present errors and introduces 15 regressions, indicating that high candidate coverage alone is insufficient when the judge is less reliable. Overall, the rescue accounting supports ATAP as a selection mechanism rather than a generation mechanism. It improves when correct candidates are already present but under-selected, while GT-absent cases require better sampling, prompting, tools, or search rather than better aggregation alone.

6 Conclusion

We introduced Adaptive Tournament Answer Pooling (ATAP), a training-free inference-time method for improving answer selection in self-consistency-style reasoning. ATAP separates answer discovery failures, where the correct answer is absent from the sampled pool, from answer selection failures, where the correct answer is present but not chosen by the aggregation rule. The method uses answer-pool entropy to route settled warm-up cases to cheap consensus and applies answer-level tournament rescue to higher-budget routed cases. Across competition mathematics and GPQA Diamond, ATAP improves over majority voting, DeepConf-Low, and Universal Self-Consistency, with the largest gains on HMMT 2026 and AIME 2025. The rescue analysis shows that these gains often arise in the intended answer-present regime: the correct answer is already generated and retained, but direct consensus selects a competing candidate. The efficiency results further show that adaptive routing can reduce token usage relative to fixed 64-trace sampling, although sequential tournament adjudication can increase wall-clock time on some benchmarks. The broader implication is that test-time scaling should distinguish between generating candidates and selecting among them. More samples help only when they improve candidate coverage or produce reliable consensus. When the correct answer is already present but under-selected, targeted answer-level adjudication can recover solutions that frequency-based aggregation misses.

7 Limitations

ATAP is a targeted answer-selection mechanism, not a complete solution to reasoning failures. If the correct answer is absent from the sampled or retained pool, tournament adjudication cannot recover it without changing the solver, prompt, sampling distribution, or generation budget. This limitation is reflected in the rescue accounting in Table 2, where ground-truth answers absent from the candidate pool remain unrecoverable by selection alone.

The method also depends on answer extraction, answer normalization, and equivalence clustering. Errors in these components can distort the measured entropy of the answer pool, merge distinct hypotheses, split equivalent hypotheses, or construct incomplete candidate packets. The tournament stage further inherits limitations of LLM-based judging: pairwise decisions may be sensitive to prompt wording, packet completeness, candidate order, seeding, and verifier sampling noise. Grouping by answer reduces redundant trace-level comparisons, but it does not guarantee that the synthesized packet contains every decisive step or that the judge will identify the correct answer.

Our empirical scope is limited. The reported experiments use a single reasoning backbone and a small set of competition mathematics and multiple-choice scientific reasoning benchmarks. Additional evaluations across model scales, domains, prompt families, and longer-form tasks are needed before treating the routing thresholds or tournament prompts as generally calibrated. Finally, although the adaptive controller can reduce token usage on settled instances, the synthesis and pairwise tournament stages introduce sequential judge calls. As shown in Table 1, this can reduce total tokens while still increasing wall-clock time on some benchmarks.

8 References

References

[1] Xuezhi Wang, Jason Wei, Dale Schuurmans, Le Quoc, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

[2] Zhangyue Yin, Qiushi Sun, Qipeng Guo, Zhiyuan Zeng, Xiaonan Li, Tianxiang Sun, Cheng Chang, Qinyuan Cheng, Ding Wang, Xiaofeng Mou, Xipeng Qiu, and Xuanjing Huang. Aggregation of reasoning: A hierarchical framework for enhancing answer selection in large language models. In Nicoletta Calzolari, Min-Yen Kan, Veronique Hoste, Alessandro Lenci, Sakriani Sakti, and Nianwen Xue, editors, *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 609–625, Torino, Italia, May 2024. ELRA and ICCL. URL `https://aclanthology.org/2024.lrec-main.53/`.

[3] Wenting Zhao, Pranjal Aggarwal, Swarnadeep Saha, Asli Celikyilmaz, Jason Weston, and Ilia Kulikov. The majority is not always right: Rl training for solution aggregation, 2025. URL https://arxiv.org/abs/2509.06870.

[4] Pranjal Aggarwal, Aman Madaan, Yiming Yang, and Mausam. Let's sample step by step: Adaptive-consistency for efficient reasoning and coding with llms, 2023. URL https://arxiv.org/abs/2305.11860.

[5] Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning, 2024. URL https://arxiv.org/abs/2401.10480.

[6] Yichao Fu, Xuewei Wang, Yuandong Tian, and Jiawei Zhao. Deep think with confidence. *arXiv preprint arXiv:2508.15260*, 2025.

[7] Amir Taubenfeld, Tom Sheffer, Eran Ofek, Amir Feder, Ariel Goldstein, Zorik Gekhman, and Gal Yona. Confidence improves self-consistency in llms. In *Findings of the Association for Computational Linguistics: ACL 2025*, page 20090–20111. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.1030. URL http://dx.doi.org/10.18653/v1/2025.findings-acl.1030.

[8] Zhi Zhou, Tan Yuhao, Zenan Li, Yuan Yao, Lan-Zhe Guo, Xiaoxing Ma, and Yu-Feng Li. Bridging internal probability and self-consistency for effective and efficient llm reasoning, 2025. URL https://arxiv.org/abs/2502.00511.

[9] Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. Universal self-consistency for large language model generation, 2023. URL https://arxiv.org/abs/2311.17311.

[10] Yingchuan Zhang, Terry Ma, Wenxuan Zhong, and Ping Ma. A single revision step improves token-efficient llm reasoning, 2026. URL https://arxiv.org/abs/2602.02828.

[11] Shubham Toshniwal, Ivan Sorokin, Aleksander Ficek, Ivan Moshkov, and Igor Gitman. Genselect: A generative approach to best-of-n, 2025. URL https://arxiv.org/abs/2507.17797.

[12] Yantao Liu, Zijun Yao, Rui Min, Yixin Cao, Lei Hou, and Juanzi Li. Pairwise rm: Perform best-of-n sampling with knockout tournament. *arXiv e-prints*, pages arXiv–2501, 2025.

[13] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

[14] Lin Shi, Chiyu Ma, Wenhua Liang, Weicheng Ma, and Soroush Vosoughi. Judging the judges: a systematic study of position bias in llm-as-a-judge (2024). *URL https://arxiv.org/abs/2406.07791*.

[15] DeepSeek-AI. Deepseek-r1-0528-qwen3-8b. https://huggingface.co/deepseek-ai/DeepSeek-R1-0528-Qwen3-8B, 2025. Hugging Face model card.

[16] Harvard-MIT Mathematics Tournament. Hmmt february 2026 problems and solutions. https://www.hmmt.org/www/archive/292, 2026. Official HMMT archive.

[17] Mathematical Association of America. American invitational mathematics exam. https://maa.org/maa-invitational-competitions/, 2026. MAA Invitational Competition Sequence.

[18] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

A Ethics, Broader Impacts, and Asset Information

A.1 Broader Impacts

This work aims to make inference-time reasoning systems more reliable and compute-aware by distinguishing between candidate generation and candidate selection. Potential positive impacts include improved auditing of self-consistency pipelines, reduced unnecessary sampling on settled problems, and better diagnosis of cases where a model has already generated the correct answer but a simple aggregation rule fails to select it.

The same capability can have negative or unintended uses. More accurate automated solution of contest-style mathematics and scientific reasoning problems could facilitate academic dishonesty if deployed in educational settings without controls. Judge-based rescue can also create unwarranted confidence in a wrong answer if users treat the tournament outcome as verified truth rather than a heuristic selection decision. More generally, stronger and cheaper inference-time reasoning may be useful in both benign and harmful planning contexts. We do not release a new foundation model or a new high-risk dataset, and our experiments are limited to public benchmark-style tasks. Responsible use should include disclosure of uncertainty, human oversight in high-stakes settings, and domain-specific validation before deployment beyond benchmark evaluation.

A.2 Asset, Data, and Code Availability

We use existing models and benchmarks and do not introduce a new dataset or new model weights. The base model used in the experiments is DeepSeek-R1-0528-Qwen3-8B, which is released under the MIT License. GPQA Diamond is used under its CC BY 4.0 license and associated access conditions, including the request not to reveal examples in plain text or images online. HMMT February 2026 problems are credited to the official HMMT archive; when using the MathArena-converted version of HMMT February 2026, we follow its CC BY-NC-SA 4.0 license. AIME 2025 and AIME 2026 are credited to the Mathematical Association of America; because the AIME problem statements are copyrighted, we use them only for internal evaluation and report aggregate results rather than redistributing problem text.

We release the source code for ATAP, evaluation scripts, prompt templates, and reproduction instructions under the MIT License. For benchmark assets whose terms restrict redistribution, the code expects users to obtain the data from the original source or an authorized benchmark distribution rather than bundling restricted problem statements directly.

B Implementation Details and Ablations

This appendix collects the concrete implementation choices used in our experiments. The main paper gives the method definitions; here we provide an algorithmic summary and specify the controller settings, the DeepConf-style stability instantiation, and the candidate-pool ablation used to choose the tournament candidate cap.

B.1 Algorithmic Summary

Algorithms 1 and 2 provide pseudocode for the two stages of ATAP. Algorithm 1 constructs the retained valid trace pool from warm-up routing and stability-based online sampling. Algorithm 2 then either returns the confidence-weighted consensus answer for warm-up-stop instances or runs answer-level tournament rescue for higher-budget instances.

B.2 Controller and Decoding Configuration

We use a two-regime controller. Each problem first receives $n_0 = 16$ warm-up traces. We extract and cluster valid answers from the warm-up pool, then compute the Shannon entropy of the resulting answer distribution. If the warm-up entropy is at most $h_{\text{stop}} = 0.5$, the instance stops after warm-up and returns the consensus answer from the warm-up pool. Otherwise, the instance enters the higher-budget regime and samples additional traces up to a maximum total budget of $B = 64$.

Algorithm 1 ATAP Routing and Retained-Pool Construction

Require: Problem x, model π, answer extractor g, stability scorer s
Require: Warm-up budget n_0, maximum total budget B, entropy threshold c, drop fraction η
Ensure: Retained valid trace pool $\mathcal{P}$, warm-up-stop flag z

Sample warm-up traces $\mathcal{T}_0 = \{\tau_i\}_{i=1}^{n_0} \sim \pi(\cdot \mid x)$
$\mathcal{T}_0^{\mathrm{val}} \leftarrow \{\tau \in \mathcal{T}_0 : g(\tau) \neq \varnothing\}$
$H_0 \leftarrow \mathrm{Entropy}(\{g(\tau) : \tau \in \mathcal{T}_0^{\mathrm{val}}\})$
if $\mathcal{T}_0^{\mathrm{val}} \neq \varnothing$ **and** $H_0 \leq c$ **then**
    $\mathcal{P} \leftarrow \mathcal{T}_0^{\mathrm{val}}$
    $z \leftarrow \mathrm{true}$
else
    $\theta \leftarrow \mathrm{Percentile}_{100\eta}\big(\{s(\tau) : \tau \in \mathcal{T}_0\}\big)$
    $\mathcal{T}_{\mathrm{on}} \leftarrow \mathrm{OnlineSample}(\pi, x, B - n_0, \theta)$
    $\mathcal{P} \leftarrow \{\tau \in \mathcal{T}_0^{\mathrm{val}} : s(\tau) \geq \theta\} \cup \{\tau \in \mathcal{T}_{\mathrm{on}} : g(\tau) \neq \varnothing,\ \tau \text{ completed}\}$
    $z \leftarrow \mathrm{false}$
end if
return $(\mathcal{P}, z)$

Algorithm 2 ATAP Consensus and Answer-Level Rescue

Require: Problem x, retained valid trace pool $\mathcal{P}$, answer extractor g, stability scorer s
Require: Warm-up-stop flag z, candidate count K
Ensure: Final answer $\hat{a}$

Let $\mathcal{U}(\mathcal{P})$ be the clustered distinct answers in $\{g(\tau) : \tau \in \mathcal{P}\}$
for $u \in \mathcal{U}(\mathcal{P})$ **do**
    $W(u) \leftarrow \sum_{\tau \in \mathcal{P} : g(\tau) = u} s(\tau)$
end for
$\hat{a}_{\mathrm{conf}} \leftarrow \arg\max_{u \in \mathcal{U}(\mathcal{P})} W(u)$
if z **then**
    return $\hat{a}_{\mathrm{conf}}$
end if
$\mathcal{V} \leftarrow$ top K answers in $\mathcal{U}(\mathcal{P})$ ranked by $W(u)$
for $u \in \mathcal{V}$ **do**
    $\mathcal{T}(u) \leftarrow \{\tau \in \mathcal{P} : g(\tau) = u\}$
    $R(u) \leftarrow \mathrm{Summarize}(x, u, \mathcal{T}(u))$
end for
return $\mathrm{Tournament}\big(\{(u, R(u)) : u \in \mathcal{V}\}\big)$

Unless otherwise specified, our default configuration is

$$n_0 = 16, \qquad B = 64, \qquad h_{\mathrm{stop}} = 0.5, \qquad \eta = 0.1, \qquad K = 8.$$

The parameter $\eta = 0.1$ sets the DeepConf-style retention threshold to the 10th percentile of warm-up stability scores, corresponding to a conservative bottom-10% stability cut. In the higher-budget regime, warm-up traces are retained only if they have a valid answer and their stability score exceeds this threshold. Online traces are generated with the same stability threshold and are retained only if they complete and produce a valid normalized answer.

Solver traces are generated with temperature 0.6, top-p $= 0.95$, top-20 log probabilities, and maximum generation length 131072. Experiments are run with vLLM 0.13.0 on NVIDIA H200 GPUs using tensor parallel size 1 and FP8 KV cache. For the three-seed math experiments, we use independent inference seeds while keeping the benchmark questions fixed.

Tournament rescue is applied only after the retained pool has been constructed. When rescue is triggered, we rank answer clusters by cumulative stability-weighted support and keep the top $K = 8$ candidate answers for answer-level packet construction and tournament adjudication.

All answer-pool statistics are computed only over traces with valid normalized answers. We cluster mathematically equivalent extracted answers before computing answer counts, entropy, support, and

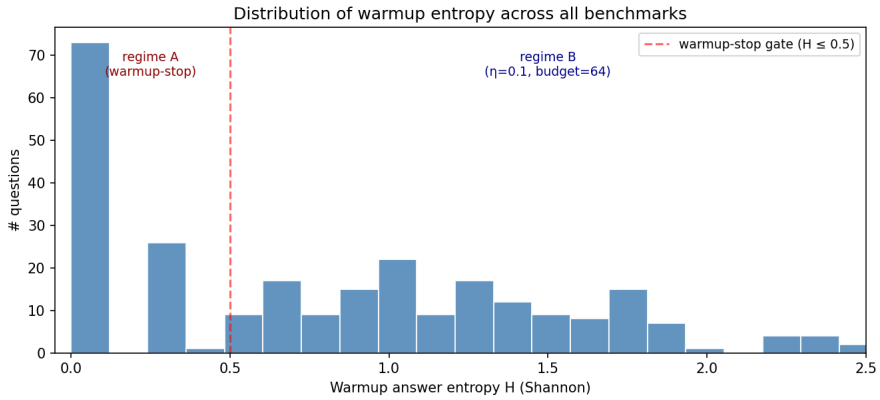

Figure 3: Distribution of warm-up answer entropy across benchmarks. The dashed line marks the warm-up stopping threshold $h_{\text{stop}} = 0.5$. Instances to the left of the threshold are highly concentrated after 16 traces and are routed to cheap consensus; instances to the right enter the higher-budget regime.

candidate rankings. If no valid answer is extracted from the warm-up pool, the instance is treated as maximally uncertain and routed to the higher-budget regime.

B.3 Warm-Up Entropy Cutoff Selection

Figure 3 shows the distribution of warm-up answer entropy across the evaluated benchmarks. The distribution has a large low-entropy mass near zero, corresponding to instances where the model repeatedly generates the same normalized answer during the 16-trace warm-up. We use $h_{\text{stop}} = 0.5$ as a conservative warm-up stopping threshold: only pools with very concentrated answer distributions are routed to the cheap consensus regime.

Empirically, instances below this threshold were already settled after warm-up. Expanding these examples to the full 64-trace budget did not change the final selection outcome in our pilot sweeps, suggesting that additional sampling mostly reinforces the same answer rather than producing useful new candidate diversity. Thus, the cutoff is intended to identify cases where the model's sampled answer distribution is effectively committed, not to certify that the leading answer is correct. Low-entropy pools may still be wrong, but they are unlikely to benefit from answer-level rescue because there are few competing hypotheses to adjudicate.

Instances above the threshold are routed to the higher-budget regime, where additional sampling and stability-based retention can construct a richer candidate pool. Tournament rescue is then performed over this candidate pool.

B.4 DeepConf-Style Stability and Retained-Pool Construction

We instantiate trace reliability using the same token-level stability instantiation as DeepConf [6]. For a completed trace τ_i, the decoder returns the top-k log-probabilities at each generation step. In our experiments, $k = 20$. Let $\ell_{i,t,r}$ denote the r-th returned log-probability at token position t. We compute the token-level confidence score as

$$c_{i,t} = -\frac{1}{k} \sum_{r=1}^{k} \ell_{i,t,r}.$$

The trace-level stability score is the minimum sliding-window mean over these token-level scores:

$$s_i = \min_j \frac{1}{w} \sum_{t=j}^{j+w-1} c_{i,t},$$

with window size $w = 2048$. For traces shorter than w, we use the mean over available token-level confidence scores; if no token log-probabilities are available, we set $s_i = 0$.

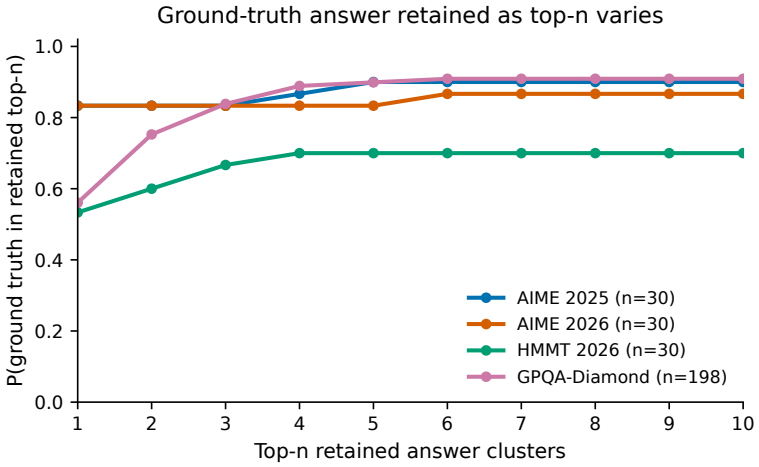

Figure 4: Ground-truth answer retention as a function of the number of retained answer clusters considered by the rescue stage. Retention increases quickly for small candidate pools and largely saturates by roughly $n = 8$ across the evaluated datasets. This supports using a small candidate cap while preserving most of the attainable answer-selection headroom.

In the higher-budget route, we compute the retention and early-stopping threshold

$$\theta = \text{Percentile}_{100\eta}\big(\{s_i : \tau_i \in \mathcal{T}_0\}\big).$$

With $\eta = 0.1$, θ is the 10th percentile of completed warm-up trace stability scores. Warm-up traces are generated to completion and retained if they have a valid answer and satisfy $s_i \geq \theta$. Online traces are generated with θ passed to the decoding backend as a generation-time stability threshold. The backend applies the threshold to an online windowed confidence statistic during autoregressive generation. A trace is retained only if it produces a valid normalized answer and completes without being stopped early. Thus the retained pool is

$$\mathcal{P} = \{\tau_i \in \mathcal{T}_0 : g(\tau_i) \neq \varnothing,\ s_i \geq \theta\} \cup \{\tau_i \in \mathcal{T}_{\text{on}} : g(\tau_i) \neq \varnothing,\ \tau_i \text{ completed}\}.$$

Our framework can use other confidence or reliability scores for post-hoc filtering, confidence-weighted voting, and candidate ranking. However, generation-time early stopping requires an online version of the score that can be computed on partial generations. If a confidence formulation provides only a completed-trace score, it can replace DeepConf-style stability for retained-pool filtering and weighted support, but not for online stopping.

B.5 Consensus Scoring and Candidate Ranking

For each retained valid answer u, we compute stability-weighted support

$$W(u) = \sum_{\tau_i \in \mathcal{P} : g(\tau_i) = u} s_i.$$

The consensus answer is the answer with maximum stability-weighted support. We rank answer clusters by cumulative stability-weighted support $W(u)$ and select the top K candidates for packet construction and tournament judging.

B.6 Candidate-Pool Size Ablation

Tournament rescue can only recover an error when the correct answer has already been generated and retained among the candidate answer clusters. We therefore ablate the number of retained answer clusters considered by the rescue stage. For each value of top-n, we measure whether the ground-truth answer appears among the top-n retained clusters ranked by stability-weighted support.

Based on this ablation, we set the rescue candidate cap to $K = 8$ in our experiments. Larger candidate pools provide little additional ground-truth coverage while increasing packet synthesis and

judge-comparison cost. This ablation measures answer-selection headroom: when the ground-truth answer is absent even from larger retained pools, the failure is due to answer discovery or retention rather than tournament selection.

B.7 Answer-Level Packet Construction and Tournament Procedure

For each selected candidate answer u, we gather the supporting retained traces

$$\mathcal{T}(u) = \{\tau_i \in \mathcal{P} : g(\tau_i) = u\}$$

and convert them into a candidate packet

$$R(u) = \mathrm{Summarize}(x, u, \mathcal{T}(u)).$$

The packet contains the final answer, the main reasoning evidence supporting that answer, concrete intermediate values, critical assumptions, and likely failure points. This converts the selection problem from trace-level comparison to answer-hypothesis comparison.

Candidate packets are seeded by cumulative stability-weighted support. We use a bracket-style pairwise tournament. Each match compares two answer-level packets using a verifier prompt. The judge is sampled multiple times per match; the candidate with majority judge support advances. If the verifier samples tie or cannot be parsed, the higher-seeded candidate advances. The final tournament winner is returned as $\hat{a}_{\mathrm{rescue}}$.

C Packet Synthesis

The following prompt templates were used to generate the packet.

C.1 Trace Generation Summary Instruction

This instruction is appended to the original problem when generating reasoning traces. It asks the model to provide a structured public summary after its reasoning, which is later used for answer-level packet construction.

```
Solve this step by step. After your reasoning, provide a structured summary
outside of your thinking. Use this format:

SUMMARY:
- Method: <one short phrase describing the main approach>
- Key steps:
  1. <first decisive step with its result>
  2. <second decisive step with its result>
  3. <third decisive step with its result>
  ... (continue as needed)
- Critical claims or assumptions:
  - <most important claim, formula, or assumption the solution depends on -
    state the actual value or result, not just a description>
  - ... (list all critical ones)
- Main uncertainty or possible failure point:
  - <the step or assumption most likely to be wrong if the solution is incorrect>
  - ... (list if multiple)
- Final answer: <answer>

Put your final answer within \boxed{}
```

C.2 Candidate Packet Synthesis Prompt

For each candidate answer, we collect summaries from traces that produced that answer and synthesize them into a single answer-level packet. We find that oftentimes the model does not adhere well to the initial system prompt to generate the structured summary meaning. Thus we need the synthesis prompt. Additionally, the model sometimes gets stuck in thinking mode thus we prefill SUMMARY: to skip the often obtuse thinking.

```
Below are {num_summaries} independent solution traces that all conclude the
answer is: {ans}

{options_text}

Using the best evidence from ALL traces above, write a single structured
summary in this format:

SUMMARY:
- Method: <one short phrase>
- Key steps:
  1. <first step with its numerical result>
  2. <second step with its numerical result>
  ... (continue as needed)
- Critical claims or assumptions:
  - <most important claim - state the actual value, not just a description>
  ... (list all critical ones)
- Main uncertainty or possible failure point:
  - <the step most likely to be wrong>

Important: Include ALL concrete intermediate values (numbers, probabilities,
formulas) from any trace. Do not omit numerical results. Do not add claims not
supported by the traces. Do not write a <think> block or private analysis.
Start directly with the structured summary.
```

D Adjudication Prompt

We tested several tournament-prompt variants on the math benchmarks and selected D.1 because it produced the fewest regressions. Due to compute constraints, GPQA Diamond was evaluated with a single adjudication prompt.

D.1 Default Pairwise Tournament Prompt

```
You are a careful verifier of candidate solutions.

Your job is to find which candidate is correct by checking their key claims,
not by picking the most polished or familiar-sounding answer.

Here is the problem:

{question}

[Solution A]: {ans_a}
{summary_a}

[Solution B]: {ans_b}
{summary_b}

Verification procedure:

Step 1 - Find where the candidates disagree.
Look at the key numerical values, probabilities, or intermediate results that
differ between the candidates. These disagreement points are where errors hide.

Step 2 - For each disagreement point, independently verify which value is
correct. Work through the calculation yourself from first principles. Do NOT
accept any candidate's claimed value just because it is stated confidently or
called a "standard result." A standard result may not apply if the problem has
```

```
constraints that violate the result's assumptions.

Step 3 - Eliminate the candidate whose key values you found to be wrong in
Step 2.

Step 4 - If both remain, choose the one whose reasoning is most internally
consistent and whose intermediate values you verified.

Important:
- If the candidates cite different values for the same quantity, at most one
  can be right. Verify it yourself.
- A claim like "by symmetry," "it is well known," or "the standard probability
  is X" is not a proof. Check whether the stated conditions actually hold.

Which solution is correct, A or B? Reply with just A or B, then put the winning
answer in \boxed{}.
```

D.2 Adjudication-Style Pairwise Prompt

In some experiments, we used a more explicit adjudication prompt.

```
You are an adjudicator deciding between two candidate solutions.

Problem:
{question}

[Candidate A]: {ans_a}
{summary_a}

[Candidate B]: {ans_b}
{summary_b}

Adjudication rules:

1. Do not start by trusting either candidate. First build your own verification
   target from the problem statement:
   - What exactly must be computed, proved, selected, or explained?
   - What are the hard constraints?
   - What would count as a decisive error?

2. Independently verify the answer before comparing candidates.
   Use the strongest available checking method for the task:
   - For finite/counting/search problems: check whether the cases are
     exhaustive. If the space is small enough, do a direct enumeration mentally,
     algebraically, or with pseudocode.
   - For algebraic/numeric problems: recompute the decisive quantities from
     first principles and check edge cases.
   - For logical/proof problems: test each implication, converse, and excluded
     case.
   - For empirical/factual/domain questions: verify against the provided
     evidence or reliable sources if available; do not invent facts.

3. Pay special attention to exclusion claims.
   Any claim of the form "no cases," "only these cases," "must be," "uniform by
   symmetry," "independent," or "standard result" must be explicitly tested.
   Try to construct a counterexample before accepting it.

4. Make a compact audit table:
   - decisive issue
```

```
   - independently verified result
   - whether Candidate A matches it
   - whether Candidate B matches it

5. Then decide:
   - Prefer the candidate whose final answer matches the independently verified
     result.
   - If neither final answer is verified, choose the one with fewer decisive
     false claims.
   - Do not reward polish, confidence, popularity, trace count, or
     familiar-sounding methods.
   - Both candidates may be wrong; still choose the better one if forced.

6. Your final line must be exactly:
FINAL: A \boxed{{ans_a}}
or
FINAL: B \boxed{{ans_b}}
```

D.3 Strict Pairwise Prompt

We also implemented a stricter completeness-oriented verifier prompt.

```
You are a careful verifier. Compare these two candidate solutions to the
problem below and determine which one is correct.

Problem:
{question}

[Solution A]: {ans_a}
{summary_a}

[Solution B]: {ans_b}
{summary_b}

Before selecting a candidate, perform a completeness check.

For each candidate, identify:
1. What it claims as the final answer.
2. What assumptions it relies on.
3. What evidence, derivation, or reasoning supports it.
4. What parts of the original problem/request it does not address.
5. Whether its reasoning only establishes a partial result rather than the full
   conclusion.

Do not choose a candidate merely because it:
- sounds plausible,
- explains one important issue correctly,
- matches a common heuristic,
- is close to another answer,
- has more supporting traces,
- or identifies one constraint while ignoring others.

A candidate is verified only if:
- it satisfies all stated constraints,
- its reasoning covers the full scope of the question,
- analogous or repeated cases have been checked,
- edge cases have been considered,
- and no unresolved gap remains between the evidence and the final answer.
```

```
Which solution is correct, A or B? Reply with just A or B, then put the winning
answer in \boxed{}.
```

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the scope of ATAP as a training-free, inference-only answer-selection method for self-consistency, and the results in Section 5 report the benchmark settings supporting those claims. The limitations discussion in Section 7 clarifies that the method cannot recover answers absent from the sampled candidate pool.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 7 discusses selection-only failures, dependence on answer normalization and judge reliability, the limited empirical scope, and the token versus wall-clock tradeoff introduced by tournament adjudication.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: This work does not provide theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The main text describes the algorithmic pipeline and benchmarks in Sections 4 and 5; Appendix B gives the controller hyperparameters, answer normalization, stability filtering, packet construction, and tournament procedure. Appendix C provides the prompt templates used for trace summarization and pairwise judging.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Appendix A.2 describes the released source code, evaluation scripts, prompt templates, reproduction instructions, and code license. For benchmarks with redistribution restrictions, the code directs users to obtain the assets from their original or authorized sources rather than bundling restricted problem text.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 5 specifies the base model, evaluated benchmarks, main baselines, token counts, wall-clock measurements, and evaluation tables. Appendix B specifies the inference budgets, thresholds, answer normalization, stability scoring, retention rule, and tournament configuration.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Figure 2 reports 95% Wilson confidence intervals for accuracy. The caption states which benchmarks are averaged over three inference seeds and which are run with one seed.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section 5.1 reports token usage and wall-clock time for the evaluated configurations and states that the wall-clock measurements were collected on a single H200 GPU under the same serving setup.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This work conforms to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: Appendix A.1 discusses potential positive impacts for reliable and compute-aware reasoning, as well as risks such as academic dishonesty, overconfidence in judge-selected answers, and broader misuse of improved reasoning systems.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: We do not release new models or datasets that are at risk for misuse; the new asset released by the paper is source code for an inference-time aggregation method.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: Appendix A.2 credits the existing model and benchmark assets and lists the relevant available licenses or copyright terms: MIT for DeepSeek-R1-0528-Qwen3-8B, CC BY 4.0 for GPQA, CC BY-NC-SA 4.0 for the MathArena HMMT conversion, and MAA copyright for AIME.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: Appendix A.2 describes the released source code, prompt templates, and evaluation scripts. The repository includes installation instructions, usage examples, hyperparameter settings, license information, and documentation for reproducing the main experiments, while restricted benchmark assets must be obtained from their original sources.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: Our work does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: Our work does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: We use DeepSeek-R1-0528-Qwen3-8B as the LLM backbone for ATAP in our experiments, and Appendix C reports the prompts used for packet synthesis and pairwise tournament judging.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.