Joint Consistency: A Unified Test-Time Aggregation Framework via Energy Minimization

Yunzhen Yao [*1] **Hongye Wang** [*2] **Yahong Wang** [2] **Michael Gastpar** [1] **Bo Jiang** [2] **Lie He** [2]

Abstract

This paper studies test-time aggregation, an approach that generates multiple reasoning traces and aggregates them into a final answer. Most existing methods rely on evaluation signals collected from candidate traces in isolation or answer frequencies, while ignoring comparative interactions among candidates. We propose *Joint Consistency* (JC), formulated as a constrained Ising-type energy minimization problem, where independent evaluation signals act as external fields and pairwise comparisons act as interactions. JC provides a unified framework for test-time aggregation that subsumes existing voting and weighted aggregation methods as special cases. Our construction of the interaction matrix leverages LLM-as-a-judge comparisons, and admits a theoretical interpretation under answer-level homogeneity assumptions. Moreover, we develop an efficient approximation strategy that makes interaction modeling practical for large-scale test-time aggregation. Experiments on math and code reasoning benchmarks show that JC consistently outperforms existing baselines across tasks, judge models, trace budgets, and trace-generation settings.

1. Introduction

Scaling test-time computation is an effective approach for improving the performance of large language models (LLMs) on complex reasoning tasks. Given a specific query, Test-Time Scaling (TTS) methods allocate additional computation at inference time to improve performance. This is typically achieved by eliciting additional reasoning steps within a single response (Wei et al., 2022) or by aggregating multiple candidate responses into a final decision (Snell et al., 2024; Welleck et al., 2024; Balachandran et al., 2025). This additional computation can be executed in parallel (Wang et al., 2023) or sequentially (Madaan et al., 2023; Qu et al., 2024), and may incorporate evaluation signals at different stages of the reasoning process (Yao et al., 2023; Fu et al., 2025).

For a wide range of TTS methods, the central question is *how to effectively aggregate diverse candidate reasoning traces into a single, reliable output*. We formulate this post-hoc decision problem as *Test-Time Aggregation* (TTA). Existing approaches to TTA include Self-Consistency (Wang et al., 2023), Weighted Self-Consistency (Guo et al., 2025b), Best-of-N selection (Jinnai et al., 2025), Self-Certainty (Kang et al., 2025), and a broad class of evaluation-based TTS methods (Cobbe et al., 2021), where candidate solutions are first fully generated and subsequently evaluated, compared, and/or aggregated.

Most TTA methods evaluate reasoning traces *in isolation*: the score of each candidate is *independent* of others, disregarding comparative information across traces. This requires the evaluator to produce well-calibrated absolute quality scores, which are often difficult to obtain. In contrast, comparative judgments between candidate traces can be more reliable, as they focus on *relative* quality rather than *absolute* scores, a principle well-established in preference learning and LLM alignment (Christiano et al., 2017; Ouyang et al., 2022).

Nevertheless, a key question remains open: how can we incorporate pairwise judgments jointly with independent evaluations into test-time aggregation in a principled and computationally efficient manner? It requires us to design a principled aggregation rule that leverages multiple comparative judgments, and make it scalable beyond the naive $O(N^2)$ cost of collecting all-pair comparisons.

Contributions. To address the above challenges, we propose *Joint Consistency* (JC) for Test-Time Aggregation (TTA), formulated as a *constrained energy minimization problem* with an Ising-type structure. Our contributions are threefold:

- **Incorporating comparative signals into test-time ag-**

[1]LINX, EPFL, Switzerland [2]Shanghai University of Finance and Economics, China. Correspondence to: Bo Jiang <syebojiang@gmail.com>, Lie He <helie@sufe.edu.cn>.

The 3rd *AI for Math Workshop* at the 43rd International Conference on Machine Learning (ICML), Seoul, South Korea, 2026.
Copyright 2026 by the author(s).

gregation. Our proposed constrained energy minimization model enables JC to incorporate pairwise comparative signals through an *interaction* matrix, while also allows independent evaluation signals to enter through an *external field* vector. We instantiate the interaction matrix using LLM-as-a-judge comparisons, and provide a theoretic interpretation under answer-level homogeneity assumptions.

- **A unifying and scalable formulation.** JC recovers evaluation-free and independent-evaluation-based aggregation methods as special cases, and additionally incorporates pairwise comparative signals through the interaction term. By varying the hyperparameter μ, JC interpolates between independent-evaluation-only and interaction-only aggregation, placing these methods within a broader family of aggregation rules and enabling systematic exploration of stronger alternatives. We further develop an efficient approximation strategy that retains the benefits of interaction modeling while making JC computationally scalable at test time.

- **Consistent gains across math and code reasoning benchmarks.** We evaluate JC on large-scale reasoning traces on math and code reasoning tasks, covering both heterogeneous crowdsourced settings such as MathArena and controlled homogeneous settings. Across extensive experiments, JC *consistently* outperforms existing baselines across tasks, judge models, trace budgets, and trace-generation settings, with especially large gains on challenging problems and heterogeneous trace pools. Moreover, JC scales effectively with the number of candidate traces, remains robust to hyperparameter choices, and achieves the benefits of interaction modeling at costs comparable to WSC through efficient approximation.

2. Prior Work and Preliminaries

In this section, we formulate the test-time aggregation problem and review prior work most relevant to Joint Consistency. A more comprehensive discussion of related work is provided in Appendix A.

2.1. Problem Formulation of Test-Time Aggregation

Consider a collection of N reasoning traces $\{y_i\}_{i=1}^N$, where each trace $y_i = (z_i, a_i)$ consists of a *reasoning chain* z_i and an *answer* a_i. Let $\mathcal{A} = \{a^{(1)}, \ldots, a^{(K)}\}$ denote the set of K distinct candidate answers appearing in $\{a_i\}_{i=1}^N$. The goal of Test-Time Aggregation (TTA) is to infer a single *aggregated answer* $\hat{a} \in \mathcal{A}$.

We partition the set of traces according to their answers. Formally, let $\{\mathcal{I}_k\}_{k=1}^K$ be a partition of $[N] := \{1, \ldots, N\}$ defined by

$$\mathcal{I}_k := \{\, i \in [N] \mid a_i = a^{(k)} \,\}. \tag{1}$$

Let $\mathbf{x} \in \{0,1\}^N$ be an indicator vector, where $\mathbf{x}_i = 1$ indicates that the aggregated answer $\hat{a} = a_i$. Then, the feasible set of $\mathbf{x}$ is $\{\mathbb{1}_{\mathcal{I}_k} \mid k \in [K]\}$, where $\mathbb{1}_{\mathcal{I}_k}$ denotes the indicator vector of the set $\mathcal{I}_k$.

2.2. Evaluation-Free TTA

Self-Consistency (SC). Self-Consistency (Wang et al., 2023; Brown et al., 2024; Schaeffer et al., 2025; Zhou et al., 2025b) is a classic *evaluation-free* method that aggregates $\{y_i = (z_i, a_i)\}_{i=1}^N$ via majority vote over final answers $\{a_i\}_{i=1}^N$, treating all candidates with equal weight. It can be equivalently formulated as

$$\min_{\mathbf{x} \in \{\mathbb{1}_{\mathcal{I}_k} \mid k \in [K]\}} \quad -\langle \mathbf{h}, \mathbf{x} \rangle, \tag{2}$$

where $\mathbf{h} = \mathbf{1}_N \in \mathbb{R}^N$ is the all-one vector. Let $\hat{\mathbf{x}} = \mathbb{1}_{\mathcal{I}_{\hat{k}}}$ be a minimizer of (2); the aggregated answer is then $\hat{a} = a^{(\hat{k})}$.

2.3. TTA with Independent Evaluation Signals

In *evaluation-based* TTA methods, evaluation is typically applied to the complete reasoning trace, using *outcome reward models* (ORMs).

Evaluation signals may be *intrinsic*, derived directly from the generator model, such as sequence log-probability, token-level certainty (Kadavath et al., 2022; Kang et al., 2025; Taubenfeld et al., 2025), or verbalized confidence (Tian et al., 2023; Xiong et al., 2024). However, such intrinsic evaluation signals are often not directly comparable across heterogeneous generators, and are usually unavailable in crowdsourced settings, limiting their applicability. Alternatively, *extrinsic* evaluation signals can be obtained from separate verifiers (Cobbe et al., 2021), trained reward models (Huang et al., 2025; Thatikonda et al., 2025; Wan et al., 2025), or from prompting-based evaluators such as LLM-as-a-judge (Zheng et al., 2023; Gu et al., 2024b; Zhou et al., 2025a; Liu et al., 2025).

Weighted Self-Consistency (WSC). Weighted Self-Consistency extends SC (2) by encoding evaluation signals in the weight vector $\mathbf{h}$, where $\mathbf{h}_i$ is the weight of trace y_i.

Self-Certainty. Self-Certainty (Kang et al., 2025) aggregates traces based on the relative ranking of their confidence scores. However, these scores are computed *independently* for each trace, without reference to other candidates. As a result, Self-Certainty does not leverage pairwise comparison signals and can be viewed as a special case of WSC, where $\mathbf{h}_i$ is given by a Borda-style rank-based score

$\mathbf{h}_i = (N - r_i + 1)^q$, where r_i denotes the rank induced by the average log-probability of trace y_i, and $q > 0$ is a hyperparameter.

2.4. TTA with Comparative Evaluation Signals

The key distinction between *independent* and *comparative* evaluation signals lies in whether the judge model evaluates candidates *jointly*. For instance, when employing LLM-as-a-judge, independent evaluation signals are elicited by assessing each trace in a separate query, whereas comparative evaluation signals are elicited by presenting two candidate traces together within one query context.

Knockout Tournament (KT). Unlike most TTA methods relying solely on independent evaluation signals, knockout tournament (Liu et al., 2025) operates purely on comparative judgments. It performs aggregation through iterative pairwise comparisons, where pairwise judges determine relative correctness between candidates from distinct answer groups and progressively eliminate inferior ones until a single survivor remains. By avoiding exhaustive pairwise comparisons, this stochastic elimination procedure reduces the computational complexity to $O(N \log N)$. Rather than comparing two candidates at a time as in KT, GenSelect (Toshniwal et al., 2025) performs an N-ary comparison, placing a handful of candidate traces within a single context to obtain the index of the best candidate.

V_1-Infer. Closely related to knockout tournament, V_1-Infer (Singh et al., 2026) also aggregates candidates purely through pairwise comparisons. It first compares candidates until each has participated in at least a specified number of comparisons to obtain initial quality scores, and then refines the ranking by comparing similarly-ranked, near-tie pairs in a Swiss-tournament fashion.

3. Joint Consistency

We propose Joint Consistency (JC), a *constrained energy minimization* approach for test-time aggregation inspired by the *Ising* model:

$$\min_{\mathbf{x} \in \{\mathbb{1}_{\mathcal{I}_k} \mid k \in [K]\}} \quad H(\mathbf{x}) := -\mu \langle \mathbf{h}, \mathbf{x} \rangle - \mathbf{x}^\top \mathbf{J} \mathbf{x}, \tag{3}$$

where $\mathbf{h} \in \mathbb{R}^N$ encodes independent evaluation signals as described in Section 2.3, $\mathbf{J} \in \mathbb{R}^{N \times N}$ encodes comparative evaluation signals, and $\mu > 0$ controls the contribution of the former.

This formulation corresponds to the Ising model augmented with an additional constraint: $\mathbf{h}$ acts as *external fields* derived from independent evaluation signals, while $\mathbf{J}$ specifies *interactions* induced by comparative evaluation signals. Intuitively, each candidate behaves like an interacting particle whose selection depends not only on its own quality estimate (external field), but also on its compatibility with other candidates (pairwise interactions). The constraint in (3) restricts $\mathbf{x}$ to select exactly one answer, where $\mathcal{I}_k$ denotes the index set of traces producing answer $a^{(k)}$, as defined in (1).

JC covers the existing TTA methods in Section 2.2 and 2.3 as special cases by setting $\mathbf{J} \equiv 0$. Figure 1 illustrates a representative example where JC outperforms existing TTA methods.

The rest of this section is organized as follows. Section 3.1 presents our design of constructing $\mathbf{J}$ from pairwise comparative judgments, together with its theoretical motivation. Section 3.2 discusses the computational complexity of Joint Consistency and introduces an efficient approximation method.

3.1. Interaction Modeling via LLM-as-a-Judge

We design the interaction matrix $\mathbf{J}$ as a positive semi-definite matrix that encodes comparative evaluation signals elicited from an LLM-as-a-judge. To proceed, we first introduce the necessary notation.

Pairwise preference probabilities. Given two reasoning traces y_i and y_j, the judge model is asked to estimate the probability that y_i is better than y_j. Let π_θ denote the LLM-as-a-judge model. We define

$$p_\theta(y_i \succeq y_j) := \mathbb{E}\left[P_\theta(y_i \succeq y_j)\right] \tag{4}$$

where $P_\theta(y_i \succeq y_j)$ denotes the probability value output by π_θ of the event that y_i is preferred to y_j, and the expectation is taken over the stochasticity of the judge model's responses. Concrete examples of the prompts used to query the judge model are provided in Appendix E.1.2 and E.2.2.

Positive semi-definite construction. We construct $\mathbf{J}$ as a positive semi-definite matrix:

$$\mathbf{J} = \mathbf{C}\mathbf{C}^\top \tag{5}$$

where $\mathbf{C} \in \mathbb{R}^{N \times N}$ is defined entrywise by

$$\mathbf{C}_{ij} := \sqrt{\frac{p_\theta(y_i \succeq y_j)}{n_i^2 n_j}}. \qquad \forall i, j \in [N].$$

Here, n_i denotes the number of traces that share the same final answer as y_i, that is, if $i \in \mathcal{I}_k$, then $n_i = |\mathcal{I}_k|$.

Intuitively, each row of $\mathbf{C}$ can be viewed as a normalized *comparative profile* of a trace: it represents how strongly the trace is preferred to each candidate. Consequently, $\mathbf{J}_{ij} = \langle \mathbf{C}_{i,:}, \mathbf{C}_{j,:} \rangle$ measures the similarity between the comparative profiles of y_i and y_j. Thus, two traces are assigned a strong interaction $\mathbf{J}_{ij}$ when they are both preferred over similar sets of competing traces. JC tends to favor an answer

Answer	Gen Model
2	grok-4.1-fast
254 ★	gpt-5-mini
128	gpt-5-nano
128	grok-4
0	glm-4.6
4	gpt-5.1

Judge Model

$P_\theta(y_i)$	$P_\theta(y_i \succcurlyeq y_j)$				
0.1		0.01	0.90	0.90	0.99	0.95
0.1			0.40	0.80	0.99	0.95
0.2					0.73	0.95
0.1					0.14	0.00
1.0						0.33
0.0						

Answer	Test-Time Aggregation				
	SC(↑)	BoN(↑)	WSC(↑)	KT(↑)	JC-H (↓)
2	1	0.1	0.1	4	-6.24
254	1	0.1	0.1	2	-6.56 ★
128	2	0.2	0.3	3	-6.20
0	1	1.0	1.0	1	-4.54
4	1	0.0	0.0	1	-5.26

Figure 1. TTA methods on crowdsourced traces for Q28 from the HMMT 2025 (Nov) dataset, where the ground-truth answer is `254`. Yellow cells denote the final aggregated answer selected by each method. Blue and green cells represent independent evaluation signals and comparative signals, respectively. SC, BoN, and WSC favor incorrect majority or high-scored answers. KT eliminates the correct answer `254` in Round 2 after losing to `128` with a pairwise score of 0.4. In contrast, JC with $\mu = 0.5$ successfully identifies the correct answer by aggregating global pairwise interactions.

group when its traces exhibit coherent pairwise preference patterns against the remaining candidates.

Theoretically, our interaction construction admits a natural interpretation through the following result. If we omit the linear term and consider only the quadratic term in (3), then the resulting objective admits a simple closed-form characterization: it selects the answer that is collectively preferred over competing candidate answers, as formalized in Proposition 3.1.

Proposition 3.1 (Answer-level interpretation of **J**). *Assume that for any $k, k' \in [K]$, there is a constant $\beta_\theta(a^{(k)} \succeq a^{(k')})$ such that for every $i \in \mathcal{I}_k$ and every $j \in \mathcal{I}_{k'}$,*

$$p_\theta(y_i \succeq y_j) = \beta_\theta\left(a^{(k)} \succeq a^{(k')}\right). \tag{6}$$

Then, for any feasible $\mathbf{x}$ of (3), i.e., $\mathbf{x} = \mathbb{1}_{\mathcal{I}_k}$,

$$\mathbf{x}^\top \mathbf{J}\mathbf{x} = \sum_{k'=1}^{K} \beta_\theta(a^{(k)} \succeq a^{(k')}). \tag{7}$$

If taking $\mathbf{h} \equiv \mathbf{0}$, an optimizer of (3) satisfies

$$\hat{k} = \arg\max_{k \in [K]} \sum_{k'=1}^{K} \beta_\theta(a^{(k)} \succeq a^{(k')}), \tag{8}$$

where $p_\theta(y_i \succeq y_j)$ is defined in (4) and $\mathcal{I}_k$ is defined in (1).

The proof of Proposition 3.1 is provided in Appendix D. Proposition 3.1 shows that interaction-only JC converts trace-level comparisons into an answer-level preference aggregation rule. Moreover, this result also motivates our approximation strategy in Section 3.2, where interactions can be estimated at the answer-group level rather than over all trace pairs.

We denote $P_\theta(y_i)$ as the independent evaluation score of trace y_i elicited by LLM-as-a-judge π_θ. Concrete prompt examples are provided in Appendix E.1.1 and E.2.1. The resulting Joint Consistency pipeline is summarized in Algorithm 1.

Algorithm 1 Joint Consistency for Test-Time Aggregation

1: **Input:** Question q, number of traces N, judge model π_θ, μ
2: Generate or collect N reasoning traces $\{y_i = (z_i, a_i)\}_{i=1}^N$ for question q.
3: Partition traces by their extracted answers: $\{\mathcal{I}_k\}_{k=1}^K$, where $\mathcal{I}_k = \{i : a_i = a^{(k)}\}$.
4: Estimate $\mathbf{h}$ using independent trace-level evaluation scores $P_\theta(y_i)$ from π_θ.
5: Estimate the interaction matrix $\mathbf{J}$ using pairwise comparative judgments $P_\theta(y_i \succeq y_j)$ from π_θ.
6: $\hat{\mathbf{x}} = \mathbb{1}_{\mathcal{I}_{\hat{k}}} \leftarrow$ Solve the constrained Ising objective in (3)
7: **Output:** Aggregated answer $\hat{a} = a^{(\hat{k})}$.

3.2. Computational Complexity and Efficient Approximation

From trace-level to answer-level comparisons. A naive implementation of JC constructs **J** by evaluating pairwise comparisons over all trace pairs, incurring an $O(N^2)$ computational cost. This quickly becomes prohibitive when the number of traces N is large. However, Proposition 3.1 shows that, under the answer-level homogeneity assumption in (6), the interaction part of the objective depends only on the answer-level preferences $\beta_\theta(a^{(k)} \succeq a^{(k')})$. This motivates an efficient approximation strategy: instead of exhaustively comparing all individual traces, we estimate interactions between answer groups by sampling only a constant number of traces from each group. The complexity is thus reduced from $O(N^2)$ to $O(K^2)$, where K denotes the number of distinct candidate answers.

Top-κ approximation for scalability. For challenging prob-

lems, K can still be large; see Table 8. We can therefore further restrict pairwise evaluation to the top-κ most frequent answers, where κ acts as a *consistency budget*. This reduces the pairwise comparison cost to $O(\kappa^2)$. Empirically, Figure 2 and Table 9 in Section 4.1.1 show that a small fixed κ already achieves strong performance.

Cost of LLM evaluation for one trace pair. LLM-as-a-judge evaluation of a trace pair places both traces in the input prompt while requiring only a small number of output tokens to produce the judgment. As input tokens are typically substantially cheaper than output tokens, pairwise evaluation is relatively cost-efficient. For example, on OpenRouter, output tokens are approximately $4.6\times$ more expensive than input tokens when querying GPT-OSS-120B (\$0.18 versus \$0.039 per million).[1] In contrast, reasoning trace generation often incurs a much larger number of output tokens than input tokens. Empirically, Table 10 shows that the cost of JC can be as low as 1% of the generation cost, suggesting that JC remains practical even for large-scale test-time aggregation.

Computational tractability for solving (3). In practice, the cost for solving (3) is *negligible* compared to the cost of generating reasoning traces and querying the judge model. Minimizing $H(\mathbf{x})$ over $\mathbf{x} \in \{0,1\}^N$ is NP-hard in general, since it is a quadratic unconstrained binary optimization (QUBO) problem, which subsumes classical NP-hard problems such as Max-Cut (Lucas, 2014). However, under our answer-level constraints, the feasible set is restricted to only K configurations, each corresponding to a candidate answer group. Consequently, solving (3) reduces to evaluating the objective over K candidates, yielding a time complexity of $O(K)$ and making exact optimization tractable at inference time. With the top-κ approximation, the complexity further reduces to $O(\kappa)$.

4. Experiments

We evaluate Joint Consistency on both math and code reasoning tasks. Our experiments cover two complementary settings. In Section 4.1, we study the crowdsourced settings where solutions to the same problem are generated by diverse models using MathArena (Balunović et al., 2025). In Section 4.2, we investigate the homogeneous settings where traces are generated by the same model so that token-level intrinsic evaluation signals, such as log-probabilities, are available. This enables comparisons with intrinsic-evaluation-based TTA baselines. Throughout the experiments in this paper, we use LLM-as-a-judge as an extrinsic evaluator for obtaining both independent and comparative evaluation signals. Experiments are repeated over 10 independent trials.

[1] https://openrouter.ai/openai/gpt-oss-120b/pricing

4.1. Crowdsourced Tasks

We evaluate Joint Consistency on heterogeneous traces from MathArena, where candidate solutions to the same problem are generated by diverse models. This setting captures a realistic crowdsourced scenario, in which traces differ in generator quality and typically lack shared intrinsic evaluation signals such as token-level log-probabilities or process rewards. Thus, intrinsic-evaluation-based methods such as Self-Certainty, DeepConf, or PRM-based aggregation are not directly applicable.

We consider four competition-level math benchmarks, namely AIME'25 (Art of Problem Solving, 2025a;b), HMMT'25-Feb, HMMT'25-Nov, and Brumo'25. These benchmarks are challenging even for strong models; for instance, GPT-4o achieves only 14% Pass@1 accuracy on one HMMT setting, as shown in Fig. 5. The MathArena traces are available on Hugging Face.[2]

To examine performance under different levels of difficulty and generator quality, we consider three MathArena variants:

- **MathArena-Full**: dataset including all problems and all available models.
- **MathArena-C**: dataset including the most challenging problems in MathArena-Full, identified by low Pass@1 accuracy and high answer diversity, i.e., a small average number of traces per answer. Dataset statistics are provided in Table 8.
- **MathArena-W**: dataset including only traces generated by the weakest models in MathArena.

4.1.1. Scalable κ-Approximation of the Interaction Matrix $\mathbf{J}$

We investigate how the consistency budget κ affects the cost and accuracy of Joint Consistency on MathArena-C. Specifically, we vary the consistency budget $\kappa \in \{2, 4, 6, 8, 10\}$ and use $\{20\%, 40\%, 60\%, 80\%\}$ of the available reasoning traces for test-time aggregation. We set $\mathbf{h} \equiv 0$ to isolate the effect of the interaction matrix $\mathbf{J}$, and use GPT-OSS-20B as the judge for both JC and WSC. For a fair cost comparison, we report the actual API usage for both independent and comparative evaluations.

The results are shown in Figure 2. We observe that **a small κ is sufficient for JC to scale beyond WSC.** JC surpasses the performance plateau of WSC and achieves an accuracy gain of over 20%, even with a small budget of $\kappa = 4$. Moreover,

[2] Traces are available at https://huggingface.co/datasets/MathArena. We use a revision where the four datasets are evaluated by 58, 38, 57, and 15 models, respectively. Each model provides four traces for every question.

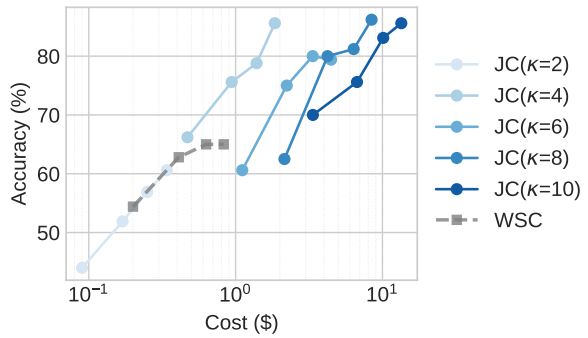

Figure 2. **Accuracy–cost trade-off under κ-approximation on MathArena-C.** Only evaluation costs are included.

across different values of κ, JC continues to improve as the evaluation budget increases, suggesting that a small subset of frequent answers is sufficient to capture most of the benefit of JC. Moreover, the results show that **the interaction J can be approximated efficiently**. Detailed results are provided in Table 9 in the appendix.

4.1.2. Robustness to Diverse Judge Models

We evaluate Joint Consistency with diverse judge models on MathArena-C. We consider five judge model families: Qwen3.5-Flash (Qwen Team, 2026), DeepSeek-V4-Flash (DeepSeek-AI, 2026), GPT-5.2 (OpenAI, 2025), GPT-OSS-20B, and GPT-OSS-120B (OpenAI, 2025). For Qwen, DeepSeek, and the default GPT-5.2 setting, we disable reasoning. For GPT-OSS-20B and GPT-OSS-120B, we use low reasoning effort. We also include GPT-5.2 with low reasoning effort, denoted as GPT-5.2(R).

The optimal μ is stable across judge models. For each judge model, we evaluate Joint Consistency over $\mu \in \{0, 0.01, 0.05, 0.1, 0.5, 1, 5, 10, 20\}$. As shown in Figure 3, the best performance is generally achieved between 0.5 and 1, across different judges. This suggests that μ does not require careful tuning for a specific judge model.

Although GPT-5.2 with low reasoning effort achieves the strongest overall performance across most values of μ, it is $19\times$ more expensive than GPT-OSS-120B in our setting and provides only modest additional gains. This indicates that frontier closed-source models are not necessary for effective judging: open-weight GPT-OSS models offer competitive performance at substantially lower cost.

JC improves over WSC and its ablations for every judge model. As shown in Figure 4, JC outperforms the **h**-only (WSC) and **J**-only variants across all judge models, and improves over WSC by $+1.8\%$ to $+11.5\%$. While reasoning improves pairwise evaluation and makes the **J**-only variant more competitive, combining **h** and **J** remains the most reliable choice, confirming that independent and comparative evaluation signals provide complementary information for TTA.

4.1.3. MathArena-Full: Complete Heterogeneous Trace Pools

We now evaluate aggregation performance on MathArena-Full, the complete heterogeneous trace pool containing all available problems and model-generated traces. We compare Joint Consistency against five baselines: Pass@1, Best-of-N (BoN), SC ($\mathbf{h} \equiv \mathbf{1}$ and $\mathbf{J} \equiv \mathbf{0}$), WSC ($\mathbf{J} \equiv \mathbf{0}$), and Knockout Tournament (KT) (Liu et al., 2025). KT also uses trace-level pairwise comparisons, but aggregates them through knockout rounds. For a fair comparison, we match the pairwise-comparison budget of KT to the sampling budget used for **J** and use the same non-finetuned judge model, with the cost specified in Table 10.

Full JC provides the strongest aggregation on MathArena-Full. As shown in Table 1, JC achieves the best performance across datasets and trace budgets. KT performs substantially worse despite also using pairwise comparisons, suggesting that the stochastic knockout procedure tends to discard globally consistent candidates based on locally noisy comparisons.

Overall, these results indicate that **the Ising-based formulation provides a more principled and robust way to incorporate pairwise comparisons into test-time aggregation.** Notably, the evaluation cost is less than 1% of the generation cost, as listed in Table 10.

4.1.4. Robustness to Lower-Tier Trace Generators

To assess whether JC relies on strong solution generators, we evaluate aggregation performance using only traces from lower-tier models. Specifically, we rank models by their Pass@1 accuracy on AIME 2025 and retain the R lowest-performing models. We vary $R \in \{3, 5, 10, 15, 20, 30\}$ while keeping all other settings identical to the main MathArena-Full experiment. **JC remains effective even with lower-tier trace generators.** As shown in Table 2, the interaction-only variant of JC achieves the best performance for most values of R and remains competitive even when full JC is strongest. This indicates that comparative coherence can provide a reliable aggregation signal even when individual traces are generated by lower-tier models. Notably, when R is small, independent evaluation scores are less reliable due to low quality of the candidate pool, while comparative evaluation signals can better distinguish relatively stronger solutions among weak traces.

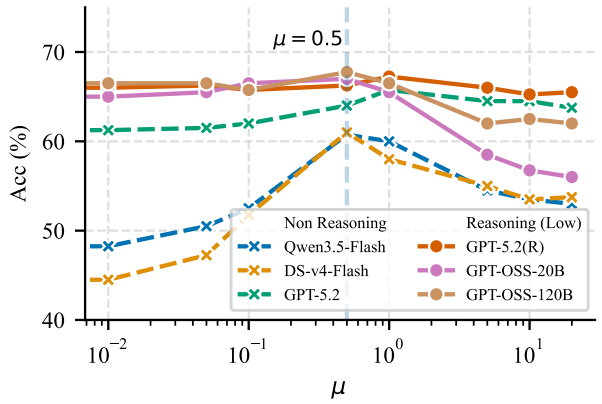

Figure 3. Sensitivity of Joint Consistency to μ on MathArena-C dataset. Across diverse judge models, JC performs best with a moderate $\mu \in [0.5, 1]$, suggesting that μ **does not require judge-model-specific tuning**.

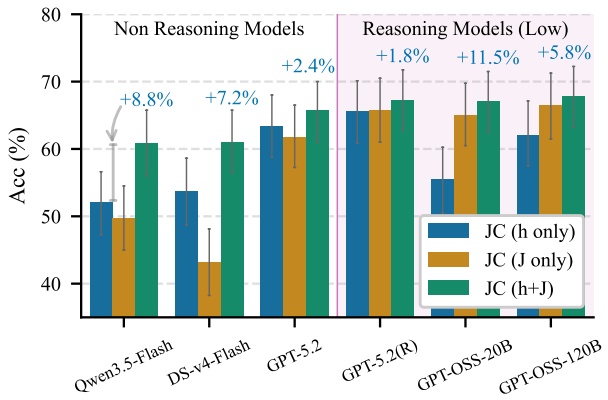

Figure 4. Accuracy of Joint Consistency across different judge models on MathArena-C dataset. Bars compare three variants of JC, namely **h** only, **J** only, and **h+J**, where the **h** only variant is WSC. The annotated values (+X%) show the accuracy gain of JC (**h+J**) over WSC.

Table 1. **Accuracies of Joint Consistency on MathArena (Full)**. The GPT-OSS-20B model with low reasoning effort is used to score both **h** and **J** with cost specified in Table 10. The number of traces N is chosen proportional to available traces in each dataset.

Dataset	N	Pass@1	BoN	KT	SC	WSC	JC
AIME 2025	24	69.9±4.5	91.2±4.9	94.1±2.0	93.5±4.4	95.3±3.8	**95.5**±3.8
	48	69.9±4.4	91.1±5.0	94.4±1.7	94.1±4.2	96.0±3.6	**96.1**±3.5
	120	69.8±4.3	89.9±5.2	94.3±1.5	94.3±4.2	96.6±3.3	**96.6**±3.3
BRUMO 2025	24	86.1±3.7	91.5±4.8	81.6±2.3	97.3±2.8	99.0±1.8	**99.5**±1.3
	48	86.2±3.5	91.0±4.9	82.6±1.5	97.8±2.5	99.7±1.0	**99.8**±0.7
	120	86.1±3.5	90.6±4.5	83.3±0.5	98.7±2.0	**100.0**±0.0	**100.0**±0.0
HMMT 2025 (Feb)	24	57.9±4.6	91.1±4.8	95.5±3.0	92.2±4.7	94.9±4.0	**96.5**±3.3
	48	57.9±4.4	90.8±4.9	97.1±2.4	93.9±4.3	97.6±2.8	**98.2**±2.4
	120	58.0±4.3	90.6±5.0	98.5±1.7	94.3±4.2	99.7±1.0	**99.8**±0.9
HMMT 2025 (Nov)	6	87.9±4.5	90.9±5.0	62.5±3.0	93.3±4.5	93.3±4.5	**93.7**±4.4
	12	87.9±4.3	91.0±5.3	63.1±2.9	93.8±4.4	93.9±4.4	**94.1**±4.3
	30	87.9±4.2	90.4±5.2	61.9±2.2	93.7±4.4	93.9±4.4	**94.0**±4.3

4.2. Benchmark against Intrinsic-Evaluation TTA Baselines in Homogeneous Settings

We now consider homogeneous settings where traces are generated together with token-level log-probabilities by the same model, making intrinsic-evaluation-based TTA methods applicable. We thus add two representative baselines: Self-Certainty (Kang et al., 2025) implemented with Borda-p voting using $p = 2$, and DeepConf (Fu et al., 2025) implemented with tail confidence. We evaluate this setting on math reasoning tasks and code reasoning tasks.

For math reasoning, we conduct experiments on AIME 2025. We use Qwen3-8B, Qwen3-32B (Yang et al., 2025), and DeepSeek-R1-8B (Guo et al., 2025a) as trace generators, and DeepSeek-R1-8B and GPT-OSS-20B (OpenAI, 2025) as judge models. For each problem, we generate an initial pool of 256 reasoning traces and uniformly subsample 64 traces for aggregation. All experiments are repeated over 64 independent trials. Detailed hyperparameters are provided in Appendix B. As shown in Table 3, JC outperforms all other baselines across all trace generators and judge models.

For code reasoning, we conduct experiments on CruxEval-O (Gu et al., 2024a), a code reasoning task where the model is given a Python function and an input and must predict the corresponding program output. The dataset contains 800 questions. We use Qwen3-Coder-30B to generate 32 traces for each question and resample $N \in \{4, 8, 16, 24\}$ traces for test-time scaling. For a fair comparison, we constrain the computational budget of JC to scale linearly with N. The results are shown in Table 4, from which we observe that JC outperforms all baselines for every N. Moreover, the gain Δ over the strongest baseline increases with N, indicating that

Table 2. **Performance of TTA methods on traces from lower-tier models.** R denotes the subset of the R lowest-ranked models whose traces are aggregated, and models are ranked by their Pass@1 accuracy on AIME 2025.

R	Pass@1	BoN	KT	SC	WSC	JC
3	11.2±2.2	23.7±4.3	9.3±1.4	19.6±3.8	23.5±4.2	**26.3**±4.4
5	16.0±2.6	31.8±4.7	13.6±0.9	29.5±4.5	31.8±4.6	**36.3**±4.8
10	26.9±3.2	50.0±5.0	40.9±3.7	43.0±4.9	51.6±4.9	**61.3**±4.9
15	34.3±3.2	63.1±4.8	57.9±4.4	54.7±4.9	68.7±4.6	**80.0**±4.0
20	40.9±3.2	67.3±4.7	69.4±3.7	71.5±4.4	77.7±4.2	**90.0**±3.0
30	52.1±3.0	76.5±4.2	83.6±2.6	83.3±3.7	88.2±3.2	**96.0**±2.0

Table 3. **TTA accuracy on AIME 2025.** JC outperforms intrinsic-evaluation baselines across trace generators and judge models.

		Trace Generation Models		
Judge Models	TTA Methods	**Qwen3 8B**	**Qwen3 32B**	**DS-R1 8B**
-	SC	77.0	80.2	83.1
	Self-Cert.	76.6	80.0	83.1
	DeepConf	76.8	80.3	83.1
DS-R1 8B	WSC	76.8	80.7	85.5
	JC	**84.1**	**84.9**	**87.1**
GPT-OSS 20B	WSC	81.4	83.4	88.4
	JC	**86.2**	**90.9**	**93.5**

Table 4. **TTA accuracy on CruxEval-O code reasoning.** Δ denotes JC's gain over the strongest non-JC baseline.

Method	N=4	N=8	N=16	N=24
Pass@1	71.3	71.3	71.3	71.3
BoN	76.2	77.0	78.0	78.0
SC	74.7	76.0	76.3	76.4
WSC	76.9	77.8	78.1	78.1
Self-Cert.	73.5	74.7	75.6	76.0
DeepConf	71.9	74.7	75.6	75.8
JC	**77.3**	**78.7**	**79.9**	**80.5**
Δ	+0.4	+0.9	+1.8	+2.4

JC benefits more effectively from larger test-time budgets.

This result shows that **JC remains effective even when competing baselines have access to intrinsic evaluation signals, and continues to scale favorably as the number of traces increases.**

5. Conclusion

We introduced Joint Consistency, a unified test-time aggregation framework that integrates independent evaluations with pairwise comparative signals under a constrained Ising formulation. JC exposes interaction modeling as a key missing component in existing TTA methods. Our experiments showed that JC outperforms strong baselines in both crowd-sourced and homogeneous settings, while incurring only marginal additional cost via our efficient approximation strategy.

Limitations. This work has several limitations. First, our paper focuses on final-answer aggregation in reasoning tasks, and it remains unclear to what extent the observed improvements generalize to open-ended reasoning or research-level problem solving. Second, although we provide a theoretically motivated construction of the interaction matrix $\mathbf{J}$ in Section 3.1, this design is not necessarily optimal. Exploring alternative constructions of $\mathbf{J}$ remains an important direction for future work. Finally, our method may inherit the limitations and potential risks of the underlying generation and judge models.

References

Art of Problem Solving. 2025 AIME I. Art of Problem Solving Wiki, 2025a. URL https://artofproblemsolving.com/wiki/index.php/2025_AIME_I. Accessed: 2025.

Art of Problem Solving. 2025 AIME II. Art of Problem Solving Wiki, 2025b. URL https://artofproblemsolving.com/wiki/index.php/2025_AIME_II. Accessed: 2025.

Balachandran, V., Chen, J., Chen, L., Garg, S., Joshi, N., Lara, Y., Langford, J., Nushi, B., Vineet, V., Wu, Y., et al. Inference-time scaling for complex tasks: Where we stand and what lies ahead. *arXiv preprint arXiv:2504.00294*, 2025.

Balunović, M., Dekoninck, J., Petrov, I., Jovanović, N., and Vechev, M. Matharena: Evaluating llms on uncontaminated math competitions. *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmark*, 2025.

Brown, B., Juravsky, J., Ehrlich, R., Clark, R., Le, Q. V., Ré, C., and Mirhoseini, A. Large language monkeys:

Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

Chen, J., Ren, J., Chen, X., Yang, C., Sun, R., Yoon, J., and Arık, S. Ö. Sets: Leveraging self-verification and self-correction for improved test-time scaling. *arXiv preprint arXiv:2501.19306*, 2025.

Chen, X., Aksitov, R., Alon, U., Ren, J., Xiao, K., Yin, P., Prakash, S., Sutton, C., Wang, X., and Zhou, D. Universal self-consistency for large language models. In *ICML Workshop on In-Context Learning*, 2024.

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., and Amodei, D. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 2017.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026.

Ferraz, T. P., Mehta, K., Lin, Y.-H., Chang, H.-S., Oraby, S., Liu, S., Subramanian, V., Chung, T., Bansal, M., and Peng, N. Llm self-correction with decrim: Decompose, critique, and refine for enhanced following of instructions with multiple constraints. In *Findings of the Association for Computational Linguistics: EMNLP*, 2024.

Fu, Y., Wang, X., Tian, Y., and Zhao, J. Deep think with confidence. In *NeurIPS 2025 Workshop on Efficient Reasoning*, 2025.

Gou, Z., Shao, Z., Gong, Y., Yang, Y., Duan, N., Chen, W., et al. Critic: Large language models can self-correct with tool-interactive critiquing. In *International Conference on Learning Representations*, 2024.

Gu, A., Rozière, B., Leather, H., Solar-Lezama, A., Synnaeve, G., and Wang, S. I. Cruxeval: A benchmark for code reasoning, understanding and execution. *arXiv preprint arXiv:2401.03065*, 2024a.

Gu, J., Jiang, X., Shi, Z., Tan, H., Zhai, X., Xu, C., Li, W., Shen, Y., Ma, S., Liu, H., et al. A survey on llm-as-a-judge. *The Innovation*, 2024b.

Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025a.

Guo, J., Chi, Z., Dong, L., Dong, Q., Wu, X., Huang, S., and Wei, F. Reward reasoning models. In *Neural Information Processing Systems*, 2025b.

Hao, S., Gu, Y., Ma, H., Hong, J., Wang, Z., Wang, D., and Hu, Z. Reasoning with language model is planning with world model. In *Conference on Empirical Methods in Natural Language Processing*, 2023.

Huang, A., Block, A., Liu, Q., Jiang, N., Krishnamurthy, A., and Foster, D. J. Is best-of-n the best of them? coverage, scaling, and optimality in inference-time alignment. In *Forty-second International Conference on Machine Learning*, 2025.

Jinnai, Y., Morimura, T., Ariu, K., and Abe, K. Regularized best-of-n sampling with minimum bayes risk objective for language model alignment. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 2025.

Kadavath, S., Conerly, T., Askell, A., Henighan, T., Drain, D., Perez, E., Schiefer, N., Hatfield-Dodds, Z., DasSarma, N., Tran-Johnson, E., et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

Kang, Z., Zhao, X., and Song, D. Scalable best-of-n selection for large language models via self-certainty. *arXiv preprint arXiv:2502.18581*, 2025.

Knappe, T., Li, R. L., Chauhan, A., Chhua, K., Zhu, K., and O'Brien, S. Semantic self-consistency: Enhancing language model reasoning via semantic weighting. In *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS*, 2024.

Lee, K.-H., Fischer, I., Wu, Y.-H., Marwood, D., Baluja, S., Schuurmans, D., and Chen, X. Evolving deeper llm thinking. *arXiv preprint arXiv:2501.09891*, 2025.

Liu, Y., Yao, Z., Min, R., Cao, Y., Hou, L., and Li, J. Pair-judge rm: Perform best-of-n sampling with knockout tournament. *arXiv preprint arXiv:2501.13007*, 2025.

Long, J. Large language model guided tree-of-thought. *arXiv preprint arXiv:2305.08291*, 2023.

Lucas, A. Ising formulations of many np problems. *Frontiers in physics*, 2:5, 2014.

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 2023.

OpenAI. Introducing gpt-5.2. https://openai.com/index/introducing-gpt-5-2/, December 2025. Accessed: 2026-05-05.

OpenAI. gpt-oss-120b & gpt-oss-20b model card, 2025. URL https://arxiv.org/abs/2508.10925.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 2022.

Qu, Y., Zhang, T., Garg, N., and Kumar, A. Recursive introspection: Teaching language model agents how to self-improve. *Advances in Neural Information Processing Systems*, 2024.

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

Schaeffer, R., Kazdan, J., Hughes, J., Juravsky, J., Price, S., Lynch, A., Jones, E., Kirk, R., Mirhoseini, A., and Koyejo, S. How do large language monkeys get their power (laws)? In *Forty-second International Conference on Machine Learning*, 2025.

Shi, Z., Zhu, Y., Xie, Y., Shi, J., Xie, G., Zhang, H., Jiang, Y., Miao, C., and Li, Q. Reasoning under uncertainty: Efficient llm inference via unsupervised confidence dilution and convergent adaptive sampling. In *Conference on Empirical Methods in Natural Language Processing*, 2025.

Singh, H., Li, X., Sareen, K., Maheswaran, M., Tan, S., Wu, X., Wang, J., Ariyak, A., Wu, Q., Khaki, S., et al. V1: Unifying generation and self-verification for parallel reasoners. In *The 1st Workshop on Scaling Post-training for LLMs*, 2026.

Snell, C., Lee, J., Xu, K., and Kumar, A. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Taubenfeld, A., Sheffer, T., Ofek, E., Feder, A., Goldstein, A., Gekhman, Z., and Yona, G. Confidence improves self-consistency in llms. *arXiv preprint arXiv:2502.06233*, 2025.

Thatikonda, R. K., Buntine, W., and Shareghi, E. Logical reasoning with outcome reward models for test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 26113–26123, 2025.

Tian, K., Mitchell, E., Zhou, A., Sharma, A., Rafailov, R., Yao, H., Finn, C., and Manning, C. D. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. In *Conference on Empirical Methods in Natural Language Processing*, 2023.

Toshniwal, S., Sorokin, I., Ficek, A., Moshkov, I., and Gitman, I. Genselect: A generative approach to best-of-n. In *2nd AI for Math Workshop@ ICML 2025*, 2025.

Wan, G., Wu, Y., Chen, J., and Li, S. Reasoning aware self-consistency: Leveraging reasoning paths for efficient llm sampling. In *Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3613–3635, 2025.

Wan, Z., Feng, X., Wen, M., Mcaleer, S. M., Wen, Y., Zhang, W., and Wang, J. Alphazero-like tree-search can guide large language model decoding and training. In *International Conference on Machine Learning*, 2024.

Wang, X., Wei, J., Schuurmans, D., Le, Q. V., Chi, E. H., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Welleck, S., Bertsch, A., Finlayson, M., Schoelkopf, H., Xie, A., Neubig, G., Kulikov, I., and Harchaoui, Z. From decoding to meta-generation: Inference-time algorithms for large language models. *Transactions on Machine Learning Research*, 2024.

Weng, Y., Zhu, M., Xia, F., Li, B., He, S., Liu, S., Sun, B., Liu, K., and Zhao, J. Large language models are better reasoners with self-verification. In *Findings of the Association for Computational Linguistics: EMNLP*, 2023.

Xiong, M., Hu, Z., Lu, X., Li, Y., Fu, J., He, J., and Hooi, B. Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms. In *International Conference on Learning Representations*, 2024.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., and Narasimhan, K. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 2023.

Zhang, J. Confidence-aware reasoning: Optimizing self-guided thinking trajectories in large reasoning models. In *Conference on Empirical Methods in Natural Language Processing: Industry Track*, 2025.

Zhao, E., Awasthi, P., and Gollapudi, S. Sample, scrutinize and scale: Effective inference-time search by scaling verification. In *Forty-second International Conference on Machine Learning*, 2025.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 2023.

Zhou, Y., Xu, A., Wang, P., Xiong, C., and Joty, S. Evaluating judges as evaluators: The jetts benchmark of llm-as-judges as test-time scaling evaluators. In *International Conference on Machine Learning*, 2025a.

Zhou, Z., Tan, Y., Li, Z., Yao, Y., Guo, L.-Z., Li, Y.-F., and Ma, X. A theoretical study on bridging internal probability and self-consistency for llm reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b.

A. Related Work

Test-time scaling (TTS) improves accuracy on complex reasoning tasks for LLMs by allocating additional compute for sampling and aggregating candidate solutions into a final decision (Snell et al., 2024; Welleck et al., 2024; Balachandran et al., 2025). Existing TTS methods can be categorized along three aspects. First, depending on whether evaluation is employed, a TTS method can be *evaluation-free* or *evaluation-based*. Second, evaluation can be applied to the complete reasoning trace, known as *outcome reward models* (ORMs), or to intermediate reasoning steps, corresponding to *process reward models* (PRMs). Finally, TTA can be implemented *in parallel*, where candidate solutions are generated independently and aggregated post hoc, or *sequentially*, where generation, evaluation, and refinement are interleaved.

Self-Consistency (SC) (Wang et al., 2023; Brown et al., 2024; Schaeffer et al., 2025; Zhou et al., 2025b) is a classic *evaluation-free* method that aggregates multiple sampled reasoning paths via majority voting over final answers, treating all candidates with equal weight. Subsequent work extends SC by incorporating more structured aggregation mechanisms. Semantic self-consistency (Knappe et al., 2024) replaces string-level voting with semantic-aware clustering and aggregation while universal self-consistency (Chen et al., 2024) generalizes SC to free-form generation by prompting the LLM to select the final answer most consistent with the overall set.

PRM-based methods incorporate state- or step-level evaluation directly into the generation process, actively guiding reasoning via inference-time search algorithms. In Tree of Thoughts (ToT) (Yao et al., 2023; Long, 2023), intermediate reasoning states are evaluated either using task-specific heuristics or by prompting the LLM itself to assess the path promise. Monte Carlo Tree Search (MCTS) based methods (Hao et al., 2023; Wan et al., 2024) use step-level value estimation to prune unpromising branches and navigate the reasoning space. More recent work explores confidence-aware generation strategies, where intrinsic uncertainty estimates are used to dynamically control reasoning depth and trajectories during inference (Zhang, 2025; Fu et al., 2025). PRM-based methods rely on generation-time signals and thus fall outside the scope of the post-hoc aggregation setting, especially in crowdsourced scenarios where such information is typically unavailable.

In contrast to PRM-based methods that assess intermediate reasoning steps, ORM-based methods weight or rank candidate solutions using *solution-level evaluation* signals. These signals may be *intrinsic*, derived directly from the generator model, such as sequence log-probability, token-level certainty (Kadavath et al., 2022; Kang et al., 2025; Taubenfeld et al., 2025), or verbalized confidence (Tian et al., 2023; Xiong et al., 2024). However, such intrinsic evaluation signals are often not directly comparable across heterogeneous generators, and are usually unavailable in crowdsourced settings, limiting their applicability. Alternatively, *extrinsic* evaluation signals can be obtained from separate verifiers (Cobbe et al., 2021), trained reward models (Huang et al., 2025; Thatikonda et al., 2025; Wan et al., 2025), prompting-based evaluators such as LLM-as-a-judge (Zheng et al., 2023; Gu et al., 2024b; Zhou et al., 2025a; Liu et al., 2025). Beyond fixed-budget parallel aggregation, some ORM-based methods adopt adaptive sampling strategies, where evaluation statistics computed over previously sampled candidates are used to dynamically determine when to stop further sampling, thus improving inference efficiency (Shi et al., 2025).

Sequential ORM-based approaches iteratively verify, critique, and refine candidate solutions using evaluation feedback, progressively steering the reasoning process toward correctness (Chen et al., 2025). Representative methods include self-verification (Madaan et al., 2023; Weng et al., 2023; Ferraz et al., 2024; Zhao et al., 2025) and self-correction (Qu et al., 2024; Gou et al., 2024; Lee et al., 2025).

B. Hyperparameters and Dataset Statistics

The experiments are conducted in the following environment, i.e., Table 5.

Table 5. Environment to reproduce the experiment.

Hardware	Specs
CPU	AMD EPYC 7402 24-Core Processor
Memory	256G
GPUs	4* NVIDIA RTX A6000
OS	Ubuntu 22.04.5 LTS
Python	3.12.11

B.1. Generation

Table 6 summarizes the decoding hyperparameters used during generation for all models. For each model, the temperature, and maximum generation length are fixed across experiments, and we use the model's native tokenizer. A dash (—) denotes that the corresponding decoding option is disabled. The Reasoning budget (API param) column specifies how each model's reasoning behavior is controlled during generation. For models that expose explicit API-level controls, we report the corresponding parameter setting (e.g., `reasoning_effort` for GPT-OSS). For models without a dedicated reasoning parameter, reasoning is enabled either by default or via prompt-level instructions.

Table 6. Generation hyperparameters and statistics.

Model	Task	Temperature	Max seq len	Reasoning budget (API param)
Qwen3-8B	Tab. 3	0.6	40k	True (`enable_thinking`)
Qwen3-32B	Tab. 3	0.6	40k	True (`enable_thinking`)
DeepSeek-8B	Tab. 3	0.6	130k	—
Qwen3-Coder-30B	Tab. 4	0.7	256k	—

B.2. Scoring

Table 7 summarizes the decoding hyperparameters used during scoring for all models. For each model, the temperature, and maximum generation length are fixed across experiments, and we use the model's native tokenizer. The Reasoning budget (API param) column specifies how each model's reasoning behavior is controlled during generation. For models that expose explicit API-level controls, we report the corresponding parameter setting (e.g., `reasoning_effort` for GPT-OSS). For models without a dedicated reasoning parameter, reasoning is enabled either by default or via prompt-level instructions.

Table 7. Scoring hyperparameters and statistics.

Model	Task	Temperature	Max seq len	Reasoning budget (API param)
DeepSeek-8B	Tab. 3	0.7	40k	—
GPT-OSS-20B	Tab. 1, 2 3, 4, 9, Fig. 2, 3, 4	1.0	128k	low (`reasoning_effort`)
GPT-OSS-120B	Fig. 2, 3, 4	0.7	128k	low (`reasoning_effort`)
GPT-5.2	Fig. 3, 4	0.7	400k	w/, w/o low (`reasoning_effort`)
Qwen3.5-Flash	Fig. 3, 4	0.7	1000k	w/o
DeepSeek-V4-Flash	Fig. 3, 4	0.7	1000k	w/o

B.3. MathArena-C

Questions used in MathArena-C are detailed in Table 8. The Pass@1 accuracy is calculated across all available models in MathArena.

Table 8. Statistics for the challenging subset of questions. Higher values (↑) for #Traces/#Ans and Pass@1 typically indicate *lower* problem difficulty. Note that the number of traces is the number of models (as reported in footnote [2]) times the number of traces per model (four).

Dataset	Q Idx	#Traces/#Ans (↑)	Pass@1 (↑)
AIME	14	232/75≈3.1	13.6%
	15	232/78≈3.0	3.5%
	All	Avg≈7.0	69.8%
BRUMO	13	152/20≈7.6	46.1%
	30	152/48≈3.2	21.1%
	All	Avg≈14.7	86.1%
HMMT Feb	19	228/54≈4.2	10.5%
	20	228/103≈2.2	12.7%
	All	Avg≈5.8	60.0%
HMMT Nov	10	60/11≈5.5	6.7%
	28	60/20=3.0	11.7%
	All	Avg≈14.4	87.8%

C. Additional Experimental Results

C.1. Pass@1 Accuracy of LLM-as-a-Judge

Figure 5 shows Pass@1 accuracies of reasoning traces from 57 models on the HMMT-2025 (Feb), measuring the effectiveness of different judge models by scoring traces (prompts defined in Appendix E.1.1).

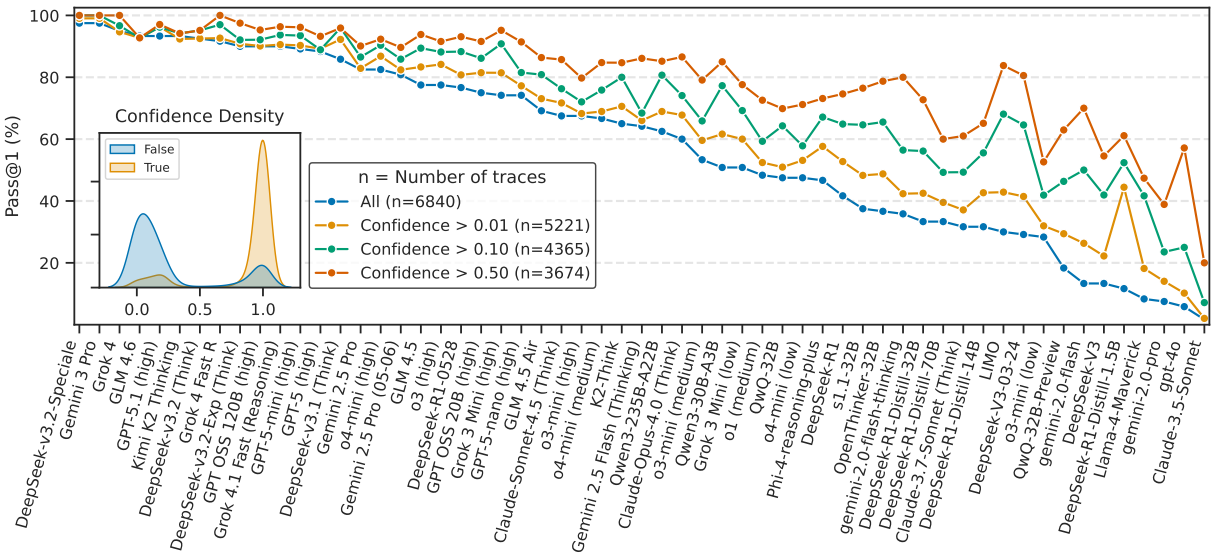

Figure 5. Pass@1 accuracies of reasoning traces from 57 models on the HMMT-2025 (Feb). Confidence scores for these traces are evaluated using a GPT-OSS-20B model with low reasoning effort. The inset plot illustrates the score density functions for correct versus incorrect reasoning traces across the full set of 6,840 traces.

C.2. Approximation to Interaction Matrix

Full data used to generate Figure 2 is listed in Table 9.

Table 9. **Accuracy (%) and cost (USD) across varying consistency budgets (κ) and number of traces (N) on challenging crowdsourced problems.** Each cell displays the format "Accuracy(%)/$Cost". Red text denotes the highest accuracy achieved for a specific N. Cells highlighted with a green background indicate configurations that outperform the best overall WSC accuracy (65%). Note that while WSC is theoretically independent of κ, minor fluctuations in its cost and performance are due to the sampling stochasticity of the traces.

N	0.20		0.40		0.60		0.80	
κ	JC ($\mathbf{h} \equiv \mathbf{0}$)	WSC ($\mathbf{J} \equiv \mathbf{0}$)	JC ($\mathbf{h} \equiv \mathbf{0}$)	WSC ($\mathbf{J} \equiv \mathbf{0}$)	JC ($\mathbf{h} \equiv \mathbf{0}$)	WSC ($\mathbf{J} \equiv \mathbf{0}$)	JC ($\mathbf{h} \equiv \mathbf{0}$)	WSC ($\mathbf{J} \equiv \mathbf{0}$)
2	44.0%/$0.09	54.4%/$0.20	51.9%/$0.17	62.8%/$0.41	56.9%/$0.25	65.0%/$0.63	60.6%/$0.34	65.0%/$0.83
3	42.9%/$0.25	57.2%/$0.22	52.5%/$0.48	63.7%/$0.41	58.8%/$0.74	63.7%/$0.63	60.0%/$1.00	62.5%/$0.83
4	66.2%/$0.47	63.1%/$0.22	75.6%/$0.94	62.8%/$0.42	78.8%/$1.39	64.4%/$0.63	85.6%/$1.85	62.5%/$0.83
5	68.1%/$0.72	59.1%/$0.21	78.1%/$1.49	62.5%/$0.42	80.0%/$2.21	64.4%/$0.64	85.6%/$2.95	63.1%/$0.83
6	60.6%/$1.11	57.5%/$0.21	75.0%/$2.23	64.4%/$0.42	80.0%/$3.35	63.1%/$0.62	79.4%/$4.47	65.0%/$0.83
8	62.5%/$2.15	57.5%/$0.22	80.0%/$4.23	62.8%/$0.41	81.2%/$6.37	65.0%/$0.64	86.2%/$8.43	63.7%/$0.83
10	70.0%/$3.36	58.3%/$0.21	75.6%/$6.71	63.4%/$0.42	83.1%/$10.12	64.4%/$0.61	85.6%/$13.44	63.1%/$0.84
12	61.9%/$4.66	59.4%/$0.20	76.9%/$9.36	63.1%/$0.41	80.0%/$14.19	63.7%/$0.61	80.6%/$18.87	63.1%/$0.83
14	58.8%/$5.76	59.4%/$0.21	74.4%/$11.52	64.7%/$0.42	71.2%/$17.58	63.7%/$0.63	67.5%/$23.35	63.7%/$0.83
16	61.3%/$6.85	60.9%/$0.20	71.2%/$13.73	63.7%/$0.41	65.0%/$20.96	64.4%/$0.62	64.4%/$27.79	62.8%/$0.84
18	63.1%/$8.23	61.6%/$0.21	68.1%/$16.43	63.7%/$0.43	70.6%/$25.35	63.7%/$0.63	65.6%/$33.53	64.4%/$0.83
20	53.8%/$9.50	59.1%/$0.21	70.0%/$18.93	60.6%/$0.42	66.9%/$29.29	63.7%/$0.63	63.1%/$38.75	62.5%/$0.83

C.3. MathArena-Full Computation Cost

Table 10 list the cost of experiments in Table 1.

Table 10. Costs for MathArena (Full) Tasks. This includes generation (G) costs and trace-scoring (Ising **h**/**J**) expenses. The scoring costs are are reported in $USD and calculated from API usage. The judge model is GPT-OSS-20B with low reasoning effort.

	Cost averaged over #traces ($)			JC Overhead	
Dataset	Generation (G)	Ising (**h**)	Ising (**J**)	**h**/G	**J**/G
AIME'25	0.072	4.67e-04	1.95e-04	0.65%	0.27%
BRUMO'25	0.076	3.51e-04	3.46e-04	0.46%	0.46%
HMMT'25-Feb	0.084	4.06e-04	2.57e-04	0.48%	0.31%
HMMT'25-Nov	0.078	4.11e-04	5.61e-04	0.53%	0.72%

C.4. Alternative designs of interaction matrix J

Here we provide a naive extension to the **J** in §3.1 by considering the τ-power of the probability $p_\theta(y_i \succeq y_j)$, i.e.,

$$\mathbf{C}_{ij} := \sqrt{\frac{p_\theta(y_i \succeq y_j)^\tau}{n_i^2 n_j}}. \qquad \forall i, j \in [N].$$

We conduct the code reasoning experiment in the same setting as Table 4 and present the results in Table 11 for $\tau = 1, 2, 3$. It shows that despite the simplicity of this extension, we already can find designs that improve existing result ($\tau = 1$) for small voting size N. We defer further discussion for future work.

Table 11. Naive alternative designs of **J**

N	$\tau = 1$	$\tau = 2$	$\tau = 3$
4	**77.3**	77.0	**77.3**
8	78.7	77.9	**78.8**
16	**79.9**	79.2	79.5
24	**80.5**	80.2	80.0

D. Proof of Proposition 3.1

Fix any feasible $\mathbf{x}$. By the constraints on $\mathbf{x}$, there exists a unique $k \in [K]$ such that $\mathbf{x} = \mathbb{1}_{\mathcal{I}_k}$, and hence

$$\mathbf{x}^\top \mathbf{J}\mathbf{x} = \sum_{i=1}^{N}\sum_{j=1}^{N} J_{ij} x_i x_j = \sum_{i,j\in\mathcal{I}_k} J_{ij}.$$

For any $i, j \in \mathcal{I}_k$, we have $n_i = n_j = |\mathcal{I}_k|$. Then,

$$\begin{aligned} J_{ij} &= \sum_{\ell=1}^{N} C_{i\ell} C_{j\ell} \\ &= \sum_{\ell=1}^{N} \frac{1}{|\mathcal{I}_k|^2 n_\ell} \sqrt{p_\theta(y_i \succeq y_\ell)\, p_\theta(y_j \succeq y_\ell)} \end{aligned}$$

By the homogeneity assumption of the proposition, for any $\ell \in \mathcal{I}_{k'}$, $p_\theta(y_i \succeq y_\ell) = p_\theta(y_j \succeq y_\ell) = \beta_\theta(a^{(k)} \succeq a^{(k')})$, and

$$\sqrt{p_\theta(y_i \succeq y_\ell)\, p_\theta(y_j \succeq y_\ell)} = \beta_\theta(a^{(k)} \succeq a^{(k')}).$$

It follows that

$$\begin{aligned} J_{ij} &= \sum_{\ell=1}^{N} \frac{1}{|\mathcal{I}_k|^2 n_\ell} \beta_\theta(a^{(k)} \succeq a^{(k')}) \\ &= \frac{1}{|\mathcal{I}_k|^2} \sum_{k'=1}^{K} \sum_{\ell\in\mathcal{I}_{k'}} \frac{1}{|\mathcal{I}_{k'}|} \beta_\theta(a^{(k)} \succeq a^{(k')}) \\ &= \frac{1}{|\mathcal{I}_k|^2} \sum_{k'=1}^{K} \beta_\theta(a^{(k)} \succeq a^{(k')}) \sum_{\ell\in\mathcal{I}_{k'}} \frac{1}{|\mathcal{I}_{k'}|} \\ &= \frac{1}{|\mathcal{I}_k|^2} \sum_{k'=1}^{K} \beta_\theta(a^{(k)} \succeq a^{(k')}), \end{aligned}$$

where we used $\sum_{\ell\in\mathcal{I}_{k'}} 1/|\mathcal{I}_{k'}| = 1$.

Finally, summing over $i, j \in \mathcal{I}_k$ yields

$$\begin{aligned} \mathbf{x}^\top \mathbf{J}\mathbf{x} &= \sum_{i,j\in\mathcal{I}_k} \frac{1}{|\mathcal{I}_k|^2} \sum_{k'=1}^{K} \beta_\theta(a^{(k)} \succeq a^{(k')}) \\ &= \sum_{k'=1}^{K} \beta_\theta(a^{(k)} \succeq a^{(k')}), \end{aligned}$$

which completes the proof.

E. Detailed Experimental Settings

E.1. Prompts Used in Math Reasoning Tasks

E.1.1. Ising-h Prompt

The prompt to query $\mathbf{h}$ in math reasoning tasks consists of three parts as follows.

user

Please reason step by step, and put your final answer within `\\boxed{}`.
{question}

assistant

{response (cot+answer)}

user

Please evaluate the above answer based on the following criteria:

1. Is the answer correct?
2. Is the reasoning process correct?

Please choose an evaluation score among 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0.
Please only output only the evaluation score.

E.1.2. ISING-J PROMPT

user

Suppose there are two responses to the same question. Please output the probability that Response 1 is a better answer than Response 2.

Question
{question}

Response 1
{response1 (cot+answer)}

Response 2
{response2 (cot+answer)}

Instruction
Now, please output the probability (a real number between 0 and 1) that Response 1 is a better answer than Response 2. Please only output the number.

E.1.3. GENERATION PROMPT

user

{question}
Please reason step by step, and put your final answer within `\\boxed{}`.

E.2. Prompt Used in Code Reasoning Tasks

E.2.1. ISING-H PROMPT

The prompt to query $\mathbf{h}$ in code reasoning tasks consists of three parts as follows.

user

Given the following Python function and input, predict the output.
Function: {code_str}
Input: {input_str}
Please think step by step after "Reasoning:`\n\n`" and then leave the output after "Output:`\n\n`". Note the output should be a python object and please ignore markdown format.{question}

assistant

{response}

user

Please evaluate the above answer based on the following criteria:

1. Is the answer correct?
2. Is the reasoning process correct?

Please choose an evaluation score among 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0.
Please only output only the evaluation score.

E.2.2. ISING-J PROMPT

user

Suppose there are two responses to the same Python function and input. Please output the probability that Response 1 is a better answer than Response 2.

Python function and input

Function:
{code_str}

Input:
{input_str}

Response 1
{content1}

Response 2
{content2}

Instruction

Now, please output the probability (a real number between 0 and 1) that Response 1 is a better answer than Response 2. Please only output the number.

E.2.3. GENERATION PROMPTS

user

Given the following Python function and input, predict the output.
Function: {code_str}
Input: {input_str}
Please think step by step after "Reasoning:`\n\n`" and then leave the output after "Output:`\n\n`". Note the output should be a python object and please ignore markdown format.{question}