

CTTS: COLLECTIVE TEST-TIME SCALING

Anonymous authors

Paper under double-blind review

ABSTRACT

Test-time scaling (TTS) has emerged as a promising, training-free approach for enhancing large language model (LLM) performance. However, the efficacy of existing methods, such as Best-of-N and Self-Consistency, is fundamentally constrained by the dominant single test-time scaling (STTS) paradigm, which relies on a single LLM agent interacting with a single reward model (SA-SR). Inspired by recent work showing that collective methods can surpass the performance ceiling of individual models, we introduce **Collective Test-Time Scaling (CTTS)**. First, we systematically investigate three primary interaction paradigms of existing multiple models: single-agent-multi-reward (SA-MR), multi-agent-single-reward (MA-SR), and multi-agent-multi-reward (MA-MR). Extensive experiments reveal that the MA-MR paradigm is consistently superior. Based on this finding, we further propose **CTTS-MM**, a novel framework that operationalizes multi-agent and multi-reward collaboration. CTTS-MM integrates two key technical contributions: (1) for agent collaboration, an **Agent Collaboration Search (ACS)** that identifies the most effective combination of LLMs from a candidate pool; and (2) for reward model collaboration, a **Mixture of Reward Models (MoR)** strategy that leverages a Prior Reward model Ensemble Selection (PRES) algorithm to select the optimal ensemble. Evaluations across seven mainstream benchmarks demonstrate that CTTS-MM significantly outperforms leading STTS methods (**+4.82% over Best-of-N**) and surpasses even flagship proprietary LLMs (**+7.06% over GPT-4.1**) and open-source LLMs. These results highlight the substantial potential of collective scaling to push the frontier of LLM inference.

1 INTRODUCTION

Recent advancements in large language models (LLMs) OpenAI (2025); Yang et al. (2024b); Brown et al. (2020); DeepSeek-AI & et al. (2025); Touvron et al. (2023) have marked a significant milestone in natural language understanding and generation. LLMs are typically optimized through training-time scaling, where huge amounts of data and parameters are applied, facing growing limitations due to their resource-intensive nature and the endless hunger for human data. To avoid introducing an extra expensive training process, test-time scaling (TTS) has emerged as an orthogonal direction for fully stimulating the ability of pre-trained LLMs during inference. The process of typical TTS methods Snell et al. (2025); Wang et al. (2023); Brown et al. (2024); Madaan et al. (2023), i.e., self-repetition-based methods Snell et al. (2025); Brown et al. (2024) can be divided into two sequential stages: 1) an LLM agent generates multiple candidate answers; 2) an external selector (reward model or manually designed selection metric) chooses the best answer. The performance of TTS highly relies on the inference quality of the LLM agent and the selection accuracy of the selector. Currently, TTS methods primarily adopt single TTS paradigm that consists of a single agent with a single selector (referred to as a “single to single” framework), which introduces two major limitations: 1) In the first stage, the limited capacity of a single agent causes a biased output distribution, leading to a constrained performance upper bound; 2) In the second stage, it imposes a prior selection preference, which hinders comprehensive and high-quality scoring of candidate answers. These intrinsic limitations of the “single to single” framework impede the further performance improvement of TTS and even lead to collapse. Thus, an essential question naturally arises: **How can TTS overcome the “single to single” framework to release the potential of LLM inference?**

Human behavior may offer some insights into the question. When tackling problems, people often engage in collaboration within teams to reach better solutions. Further, particularly challenging tasks

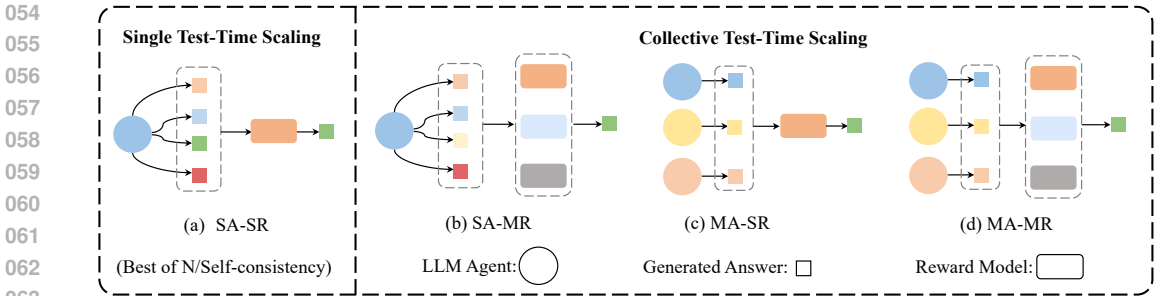


Figure 1: Comparison of previous Single Test-time Scaling (STTS) paradigm and our proposed novel Collective Test-time Scaling (CTTS) paradigms.

may require cooperation across multiple groups, combining diverse perspectives to aggregate a more comprehensive and effective outcome. This pattern is also reflected in recent developments of collective methods Wang et al. (2025); Chen et al. (2025); Shnitzer et al. (2024); Lu et al. (2024). For instance, Mixture-of-Agents (MoA) Wang et al. (2025) exploits the references from diverse LLM agents to aggregate a higher-quality final answer, breaking through the upper bound of single-agent systems. Inspired by collective methods, we advance the previous **Single TTS (STTS)** to the novel **Collective Test-Time Scaling (CTTS)**, taking the first step to explore the potential of CTTS. Specifically, we focus on investigating two key questions: (1) What is the optimal paradigm of collective test-time scaling? (2) How can we effectively scale the systems under such paradigm? To address the two questions above, we systematically design and explore three CTTS paradigms: (1) **single agent to multiple reward models (SA-MR)**; (2) **multiple agents to single reward model (MA-SR)**; (3) **multiple agents to multiple reward models (MA-MR)**. Figure 1 illustrates the differences between our proposed CTTS paradigms and the existing single TTS paradigm. To obtain the optimal CTTS paradigm, we conduct experiments to compare the above four paradigms under three benchmarks. The results are shown in Figure 2. It can be observed that, as the collective level increases, the performance improves, and the MA-MR paradigm consistently achieves the most substantial performance improvements, highlighting that both multi-agent and multi-reward-model collaboration play a critical role in the LLM inference performance.

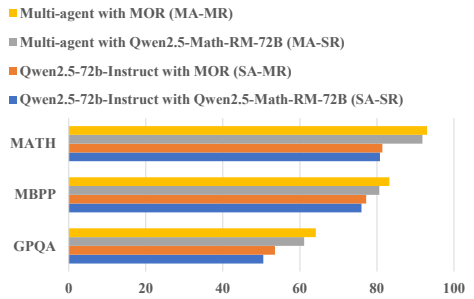


Figure 2: Comparison of three CTTS paradigm and one STTS paradigm on MATH, MBPP and GPQA.

Based on this observation, we adopt CTTS with MA-MR paradigm as the basic framework and propose a novel CTTS method called Collective Test-Time Scaling with Multiple agents to Multiple reward models (**CTTS-MM**) as an effective and simple specific instance of MA-MR paradigm. Specifically, for multi-agent collaboration specific to TTS, we first employ an **Agent Collaboration Search (ACS)** to choose the most effective agent ensemble from a candidate model pool. To guide the search with high-quality feedback, we propose a **Mixture of Reward Models (MoR)** to achieve multi-reward-model collaboration that breaks through the upper bound of a single reward model. To construct MoR regarding the given question, Prior Reward model Ensemble Selection (PRES) is proposed to select the optimal reward model or a weighted combination of them based on Pairwise Reward Ranking (PRR) metric over a curated question pool. To verify the effectiveness of our proposed CTTS-MM, we conduct extensive experiments on seven mainstream benchmarks with ten open-source LLM agents and eight reward models. Compared with existing popular TTS, collaboration methods and leading LLMs, CTTS-MM achieves significant superiority. For instance, CTTS-MM remarkably outperforms Self-consistency by 7.68% and Best of N by 4.83%. Moreover, by only utilizing open-source models, CTTS surpasses flagship closed-source LLMs, including GPT-4.1 and Claude-3.7-sonnet, which demonstrates CTTS-MM can fully release the potential of models during inference time. Our contribution can be summarized as follows:

- We take the first step towards formalizing and analyzing different paradigms of Collective Test-Time Scaling (CTTS), including 1) single agent to multiple reward models (SA-MR); 2) Multiple agents to Single reward model (MA-SR); and 3) multiple agents to multiple reward models(MA-MR). Our study reveals that MA-MR is the optimal CTTS paradigm due to the intra- and inter-collaboration of model groups.
- We propose a novel CTTS framework named CTTS-MM, which combines multiple LLM agents and multiple reward models in a unified search–reward–search pipeline. Specifically, Agent Collaboration Search (ACS) is proposed to dynamically select an optimal combination of agents from a candidate pool, and Mixture of Reward models (MoR) is proposed to provide high-quality feedback. To achieve MoR, a Prior Reward model Ensemble Selection (PRES) with a Pair-wise Reward Ranking (PRR) metric is designed to construct an effective and adaptive reward signal.
- Extensive experiments across multiple benchmarks demonstrate that our CTTS-MM consistently outperforms existing STTS methods (+6.02% over Symbolic-MoE, +7.09% over MoA), leading proprietary LLMs (+7.06% over GPT-4.1) and various open-source LLMs, validating the effectiveness of the proposed CTTS-MM framework and highlighting the substantial potential of collective test time scaling.

2 RELATED WORK

Test-Time Scaling Test-time scaling methods Snell et al. (2025); Brown et al. (2024); Madaan et al. (2023); Wang et al. (2023); Du et al. (2024); Wei et al. (2022); Yao et al. (2023); Chen et al. (2024e) mainly focus on how to enhance LLM agents’ capabilities at test time. Best of N Snell et al. (2025) is a classic TTS approach that generate answers multiple times with LLM agents and obtains the best answer based on the reward score. Similar methods Chen et al. (2024e) called self-consistency essentially follow the same paradigm, except that they use a verifier to select the answer. This verifier can be an evaluation tool or an algorithm like majority voting. Self-refine Madaan et al. (2023) obtains the optimal solution through a self-evaluation and self-correction approach while Du et al. (2024) employs a multi-round debating between two agents to reach the final answer.

Multi-agent Collaboration A growing number of researches have explored collaborative strategies among multiple agents. Emerging research Chen et al. (2025); Lu et al. (2024); Shnitzer et al. (2024); Srivatsa et al. (2024); Wang et al. (2025) aims to make selection decisions before response generation, directing queries to appropriate agents in advance. MoA Wang et al. (2025) exemplifies this by assigning LLM agents into an ensemble system. Symbolic-MoE Chen et al. (2025) proposes a Mixture-of-Experts framework that dynamically selects and combines agents based on skill-specific expertise. Other methods Chen et al. (2024c); junyou li et al. (2024); Chen et al. (2024d); Gui et al. (2024); Wang et al. (2023) fuse the results of multiple model outputs to yield a refined answer.

3 METHODOLOGY

In this section, we first provide a brief preliminary to elaborate on the specific framework of three CTTS paradigms. Then we introduce our proposed CTTS-MM. In Section 3.2, we introduce our Agent Collaboration Search (ACS). Section 3.3 details our proposed Mixture of Reward model (MoR) for selecting the optimal combination of reward models. The construction of a question pool for later selection is first presented. We then introduce Pair-wise Reward Ranking (PRR) and Prior Reward Model Ensemble Selection (PRES). Overall framework is illustrated in Figure 3

3.1 PRELIMINARY

Figure 1 illustrates three CTTS paradigms we aim to explore: (1) single agent to multiple reward models (SA-MR); (2) multiple agents to single reward model (MA-SR); (3) multiple agents to multiple reward models (MA-MR). We design a search-reward framework to systematically investigate all three paradigms. For the specific framework setting of each paradigm, MA-SR performs multi-agent ACS with a single reward model while SA-MR adopts ACS using a single agent with MoR. Note that for SA-MR, ACS is performed under multiple answers generated by a single agent. Finally, MA-MR builds upon the previous two paradigms by jointly leveraging ACS and MoR.

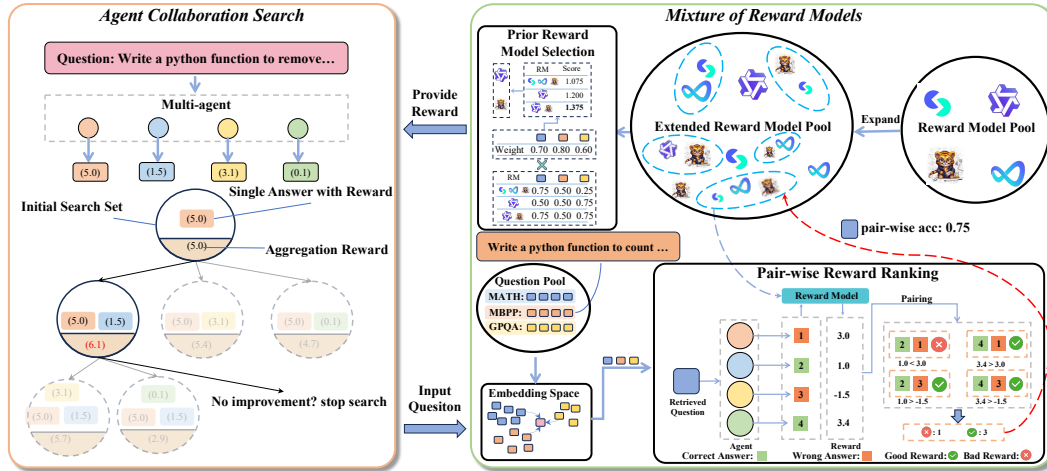


Figure 3: Overview of the proposed CTTS-MM framework. The left part illustrates the Agent Collaboration Search (ACS) while the right part depicts the Mixture of Reward Models (MoR).

3.2 AGENT COLLABORATION SEARCH

The process of ACS is illustrated in Figure 3. We design ACS based on a simple yet effective greedy search algorithm with early stop and residual aggregation. Specifically, given a question q , we first collect n candidate answers from n agents (under SA-MR setting, n candidates come from repeated generation of one agent), denoted as $\mathcal{A} = \{A_0, A_1, \dots, A_{n-1}\}$. Our goal is to obtain the optimal answer from these candidates through an iterative and reward-guided greedy search. We begin by computing the reward score for each candidate using Mixture of Reward Models, denoted as function MoR. Specifically, for each answer A_i , we obtain its reward score by:

$$r_i = \text{MoR}(q, A_i), \quad i = 0, 1, \dots, n-1. \quad (1)$$

We then sort the candidates based on their scores and select the top- k answers to initialize our search set $\mathcal{S}^{(0)}$:

$$\mathcal{S}^{(0)} = \{A_{(0)}, A_{(1)}, \dots, A_{(k-1)}\}, \quad (2)$$

where $A_{(i)}$ denotes the i -th ranked answer by score. An aggregator Agg is then used to summarize the current set of answers into a single composite response:

$$C^{opt} = \text{Agg}(\mathcal{S}^{(0)}), \quad (3)$$

and its corresponding reward score is computed as:

$$r^{opt} = \text{MoR}(q, C^{opt}). \quad (4)$$

where C^{opt} and r^{opt} are the current optimal answer and its corresponding reward score, respectively.

We then iteratively check whether augmenting the initial search set $\mathcal{S}^{(0)}$ with a remaining candidate $A_j \in \mathcal{A} \setminus \mathcal{S}^{(0)}$ can yield a better answer. For each such candidate A_j , we compute:

$$\hat{C}_j = \text{Agg}(\mathcal{S}^{(0)} \cup \{A_j\}), \quad (5)$$

$$\hat{r}_j = \text{MoR}(q, \hat{C}_j). \quad (6)$$

We identify the candidate A_j^* that yields the highest reward score:

$$A_j^* = \arg \max_{A_j \in \mathcal{A} \setminus \mathcal{S}^{(0)}} \hat{r}_j. \quad (7)$$

If $\hat{r}_{j^*} > r^{opt}$, we update:

$$\begin{cases} \mathcal{S}^{(1)} = \mathcal{S}^{(0)} \cup \{A_j^*\}, \\ C^{opt} = \hat{C}_{j^*}, \\ r^{opt} = \hat{r}_{j^*} \end{cases} \quad (8)$$

and repeat the process using $\mathcal{S}^{(1)}$ as the new base set. Otherwise, if no such improvement is found, the search terminates and C^{opt} is taken as the current optimal answer. Moreover, to mitigate potential information loss during greedy search, we incorporate a residual aggregation step. Specifically, we aggregate the final optimal answer C^{opt} with the initial candidate set \mathcal{A} to produce:

$$\begin{cases} C^{res} = \text{Agg}(\mathcal{A} \cup \{C^{opt}\}), \\ r^{res} = \text{MoR}(q, C^{res}) \end{cases} \quad (9)$$

If $r^{res} > r^{opt}$, we replace C^{opt} with C^{res} as the final output. Otherwise, we keep the original output.

3.3 MIXTURE OF REWARD MODEL

The multi-reward-model system aims to provide accurate reward scores for the preceding greedy search process. The key challenge lies in selecting suitable reward models for different questions since reward models are currently very domain-specific. Existing approaches Snell et al. (2025) typically rely on manually selecting specific reward models tailored to specific datasets or domains. While such methods may perform well on particular datasets, they lack generalization and flexibility. We argue that this challenge is fundamentally aligned with the motivation behind MoA, which aims to enhance performance and generalization through complementary collaboration among diverse agents. The essence of MoR is somewhat analogous, which is enhancing the precision of the provided rewards through the interaction and collaboration of reward models across different domains. Our core idea is to first expand the individual reward models by constructing a reward model pool and combining them through different subsets of reward models using various weighting methods. This approach allows us to significantly extend the capacity of the original pool. The next step is to select the most suitable individual reward models or their weighted combinations from the pool. Inspired by Retrieval-Augmented Generation (RAG) methods Lewis et al. (2020); Chen et al. (2024a), we introduce a diversified question pool as a prior for selecting the best reward model(s). We then propose a novel ranking metric called Pair-wise Reward Ranking (PRR) to evaluate the capability of reward models in assessing outputs from LLM agents. Based on the above techniques, a prior based reward model selection method named Prior Reward Model Ensemble Selection (PRES) is proposed. In this section, we first describe how the question pool is constructed, followed by the introduction of PRR. Finally, the details of PRES are presented.

3.3.1 DIVERSIFIED QUESTION POOL

As mentioned, manual selection of reward model(s) based on the domain of the dataset is neither generalizable nor flexible. On the other hand, it is difficult to directly select reward model(s) based on their architectures or parameters. To address this, we introduce a diversified question pool as a form of prior knowledge to guide the selection process. We construct the question pool using the validation sets of diverse tasks, such as math reasoning and coding. Then, for each question in the pool, we evaluate the correctness of each LLM agent’s response, which serves as prior knowledge for the subsequent selection process.

3.3.2 PAIR-WISE REWARD RANKING

Given the constructed question pool $\mathcal{Q} = \{q_1, q_2, \dots, q_N\}$, we aim to evaluate whether the reward score provided by the reward models is accurate. That is to say, for the same question, correct answers should receive higher scores than incorrect ones. Specifically, given a question $q \in \mathcal{Q}$, let $\mathcal{A}_q = \{(a_1, y_1), (a_2, y_2), \dots, (a_n, y_n)\}$ denote the set of answers provided by n agents, where a_i is the answer generated by the i -th agent and $y_i \in \{0, 1\}$ indicates whether the answer is correct (1) or incorrect (0). \mathcal{A}_q is then partitioned into two subsets:

$$\begin{cases} \mathcal{A}_q^{\text{pos}} = \{a_i \mid y_i = 1\}, \\ \mathcal{A}_q^{\text{neg}} = \{a_j \mid y_j = 0\}, \end{cases} \quad (10)$$

where $\mathcal{A}_q^{\text{pos}}$ and $\mathcal{A}_q^{\text{neg}}$ represent correct and incorrect responses, respectively. We then construct all possible pairs (a_i, a_j) where $a_i \in \mathcal{A}_q^{\text{pos}}$ and $a_j \in \mathcal{A}_q^{\text{neg}}$. For each pair, we query the reward function $\text{MoR}(\cdot)$ to obtain their reward scores, denoted as $r(a_i) = \text{MoR}(a_i, q)$ and $r(a_j) = \text{MoR}(a_j, q)$. If

270 $r(a_i) > r(a_j)$, we consider this pair to be accurately assessed by the reward model. The pair-wise
 271 accuracy of the reward model on question q can then be defined as:
 272

$$273 \text{Acc}_q = \frac{1}{|\mathcal{P}_q|} \sum_{(a_i, a_j) \in \mathcal{P}_q} I[r(a_i) > r(a_j)], \quad (11)$$

276 where \mathcal{P}_q is the set of all valid answer pairs for q , and $I[\cdot]$ is the indicator function. We can then
 277 rank the entire reward model pool on a given question q using Acc_q , which serves as the criterion for
 278 subsequent reward model selection. Besides, for questions where all agents provide either entirely
 279 correct or entirely incorrect answers, we consider them invalid, as the reward model’s accuracy
 280 cannot be evaluated on such questions. For multiple reward models $\{R_1, R_2, \dots, R_K\}$, the final
 281 reward score for a candidate answer is computed as a weighted combination of the individual reward
 282 scores from these K models. Specifically, for a given answer a , the reward score from multiple
 283 agents is defined as:

$$284 r_{\text{MoR}}(a, q) = \sum_{k=1}^K w_k \cdot R_k(a, q), \quad (12)$$

286 where w_k denotes the weight assigned to reward model R_k . The choice of weight computation plays
 287 a crucial role in the effectiveness of the MoR. In this work, we basically utilize three weighting
 288 strategies based on the individual reward model accuracies $\{\alpha_1, \alpha_2, \dots, \alpha_K\}$ obtained by PRR. For
 289 Linear weighting, the weight is proportional to the accuracy:
 290

$$291 w_k = \frac{\alpha_k}{\sum_{j=1}^K \alpha_j}. \quad (13)$$

293 For Softmax weighting, we compute the weights via a softmax by:
 294

$$295 w_k = \frac{\exp(\alpha_k/\tau)}{\sum_{j=1}^K \exp(\alpha_j/\tau)}, \quad (14)$$

298 where $\tau > 0$ is a temperature parameter. For naive sum, all reward models are treated equally and
 299 no weighting is applied. This corresponds to setting $w_k = 1$ for all k .
 300

301 3.3.3 PRIOR REWARD MODEL ENSEMBLE SELECTION

302 Given a question as q , a pre-trained embedding model is utilized to embed it into a d -dimensional
 303 semantic space, resulting in vector $\mathbf{e}_q \in \mathbb{R}^d$. Similarly, the question pool $\mathcal{Q} = \{q_1, \dots, q_N\}$ can be
 304 embedded into a matrix $\mathbf{E} \in \mathbb{R}^{N \times d}$, where each row \mathbf{e}_i is the embedding of q_i . We then compute
 305 the cosine similarity vector $\mathbf{s} \in \mathbb{R}^N$ by $\mathbf{s} = \mathbf{e}_q \cdot \mathbf{E}^T$. We select the top- k questions with the highest
 306 similarity scores, forming index set $\mathcal{I}_{\text{top}} \subset \{1, \dots, N\}$. For each reward model or combination R ,
 307 we retrieve its pair-wise accuracy vector $\boldsymbol{\alpha}^m \in \mathbb{R}^N$ over the top- k question set. Using the selected
 308 indices \mathcal{I}_{top} , we compute a final score by weighted dot product:
 309

$$310 \text{Score}_q^m = \sum_{i \in \mathcal{I}_{\text{top}}} s_i \cdot \alpha_i^m.$$

313 The final reward model(s) selected for q is:

$$314 R_q^* = \arg \max_m \text{Score}_q^m.$$

316 We then use the selected reward model(s) for greedy search.
 317

318 4 EXPERIMENT

319 In this section, we first analyze exploratory experiments among different CTTS paradigms. Then we
 320 present a comprehensive comparison between our CTTS-MM and existing methods across seven
 321 benchmark datasets. Finally, we perform a series of analytical and ablation studies to further inves-
 322 tigate the effectiveness of our approach.
 323

4.1 EXPERIMENTAL SETTING

Datasets. To ensure comprehensive evaluation across diverse capabilities, seven multi-domain datasets across four representative task types are utilized: (1) mathematical reasoning (MATH Hendrycks et al. (2021), AIME2024 MAA (2024)); (2) complex knowledge-based reasoning (GPQA Rein et al. (2024)); (3) instruction-following tasks (IFEval Zhou et al. (2023)); (4) code generation (MBPP Austin et al. (2021), LiveCodeBench Jain et al. (2024), HumanEval Mark Chen (2021)). All datasets except HumanEval are partitioned into non-overlapping validation and test subsets. Validation portions are utilized to construct the question pool. Details are provided in Section B of our Appendix.

Table 1: Comparison results of different TTS paradigms. MR* means utilizing our proposed MoR to select reward models. Multi-agent* means utilizing ten chosen LLMs.

Setting	Model	Reward Model	Weight Method	MBPP	MATH-500	GPQA	Avg.
Single Agent	Qwen2.5-32B-Instruct	-	-	76.00	75.60	40.91	64.17
	Qwen2.5-72b-Instruct	-	-	75.80	78.80	45.45	66.68
	Llama-3.3-Nemotron-Super-49B-v1	-	-	65.40	75.20	48.48	63.03
SA-SR	Qwen2.5-32B-Instruct	AceCodeRM-32B	-	77.40	78.2	47.47	67.69
	Qwen2.5-32B-Instruct	Qwen2.5-Math-RM-72B	-	77.00	78.80	46.46	67.42
	Qwen2.5-72b-Instruct	AceCodeRM-32B	-	76.60	80.20	51.01	69.27
	Qwen2.5-72b-Instruct	Qwen2.5-Math-RM-72B	-	76.00	80.80	50.51	69.10
	Llama-3.3-Nemotron-Super-49B-v1	AceCodeRM-32B	-	66.40	76.00	50.80	64.40
	Llama-3.3-Nemotron-Super-49B-v1	Qwen2.5-Math-RM-72B	-	65.80	76.80	50.00	64.20
SA-MR	Qwen2.5-32B-Instruct	AceCodeRM-32B+Qwen2.5-Math-RM-72B	softmax	76.6	78.2	48.48	67.76
	Qwen2.5-32b-Instruct	MR*	-	78.00	79.4	51.01	69.47
	Qwen2.5-72B-Instruct	AceCodeRM-32B+Qwen2.5-Math-RM-72B	softmax	76.8	80.20	51.51	69.50
	Qwen2.5-72b-Instruct	MR*	-	77.20	81.4	53.53	70.71
	Llama-3.3-Nemotron-Super-49B-v1	AceCodeRM-32B+Qwen2.5-Math-RM-72B	softmax	66.20	76.60	51.52	64.77
	Llama-3.3-Nemotron-Super-49B-v1	MR*	-	66.80	76.80	54.55	66.05
MA-SR	Multi-agent*	Skywork-Reward-V2-Llama-3.1-8B-40M	-	77.00	91.20	61.11	75.97
	Multi-agent*	Qwen2.5-Math-RM-72B	-	80.6	91.8	61.11	77.84
	Multi-agent*	AceCodeRM-32B	-	82.2	90.8	61.62	78.21
MA-MR (Proposed CTTS-MM)	Multi-agent*	MR*	-	83.20	93.00	64.14	80.11

Table 2: Main Results of CTTS-MM compared with leading LLMs and other related methods on seven mainstream benchmarks.

Model	AIME-2024	MATH-500	MBPP	LiveCodeBench	GPQA-Diamond	Human-eval	IFEval	Avg
<i>Open-source LLMs</i>								
Qwen-2.5-72B-Instruct	16.70	78.80	75.80	26.10	45.45	78.66	86.30	58.26
DeepSeek-R1-Distill-Llama-70B	60.00	82.80	76.40	40.70	60.10	92.07	80.30	70.34
Llama-3.3-Nemotron-Super-49B-v1	16.70	75.20	65.40	28.00	48.48	84.76	82.70	57.32
QwQ-32B	46.70	87.80	81.80	38.60	57.07	92.07	81.70	69.39
InternLM2.5-20B-Chat	3.30	55.20	55.00	14.90	34.85	69.51	64.70	42.49
Gemma-3-27b-it	30.00	84.00	70.40	27.70	50.51	86.59	81.00	61.46
Qwen2.5-32b-Instruct	20.00	75.60	76.00	24.00	40.91	77.44	78.70	56.09
TeleChat2-35B-32K	10.00	70.00	70.00	19.50	33.33	73.17	82.00	51.14
EXAONE-Deep-32B	33.30	84.38	72.80	31.60	58.59	93.90	76.30	64.41
GLM-Z1-32B-0414	66.70	90.00	74.40	44.40	59.60	96.34	83.00	73.49
Llama-3.3-70B-Instruct	30.00	73.00	70.40	30.10	46.97	84.15	90.00	60.66
Qwen3-32B	53.30	88.00	50.60	33.40	65.15	90.85	83.70	66.43
Qwen2.5-Coder-32B-Instruct	16.70	73.60	78.00	27.70	41.92	87.80	80.30	58.00
HuatuogPT-o1-72B	16.70	73.00	78.00	27.40	50.00	85.37	74.00	57.78
DeepSeek-R1-Distill-Qwen-32B	56.70	85.60	81.00	44.70	60.10	95.73	73.70	71.08
<i>Proprietary LLMs</i>								
GPT-4.1 (2025-04-14)	50.00	85.80	79.20	42.20	67.17	92.07	86.00	71.78
Claude-3.7-Sonnet (2025-02-19)	26.70	73.20	75.40	41.30	63.64	90.85	88.00	65.58
GPT-4o (2024-08-06)	10.00	74.60	74.20	29.80	52.53	85.36	82.30	58.40
Claude-3.5-Sonnet (2024-06-20)	16.70	74.20	75.80	34.30	61.62	89.63	80.30	61.79
<i>Related Methods</i>								
Majority Voting Chen et al. (2024c)	56.67	90.20	80.40	34.65	26.26	89.63	80.67	65.50
Multi-Agent Verification Lifshitz et al. (2025)	63.33	76.30	74.60	42.55	59.00	92.00	83.00	70.11
Symbolic-MoE Chen et al. (2025)	50.00	90.40	82.60	43.16	62.63	92.07	89.00	72.82
MoA Wang et al. (2025)	53.33	87.80	82.00	40.12	58.80	90.85	89.33	71.75
Self Consistency Chen et al. (2024e)	70.00	91.40	82.40	30.47	65.15	90.39	68.33	71.16
Best of N Snell et al. (2025)	66.70	90.8	75.00	44.98	60.61	96.34	83.66	74.01
<i>Ours v.s. Strong Baselines</i>								
CTTS-MM(ours)	70.00	93.00	83.20	52.28	64.14	97.56	91.67	78.84
- v.s. GLM-Z1-32B-0414	↑3.30	↑3.00	↑8.80	↑7.88	↑4.54	↑1.22	↑8.67	↑5.34
- v.s. GPT-4.1	↑20.00	↑17.20	↑4.00	↑10.08	↓3.03	↑5.49	↑5.67	↑7.06
- v.s. Best of N	↑3.30	↑2.20	↑8.20	↑7.29	↑3.53	↑1.22	↑8.01	↑4.82

LLM agents and Reward Models. For LLM agents used in our experiments, we assemble a set of ten mid-sized open-source LLMs (ranging from 20B to 72B parameters) from diverse architecture families. For reward models, we select eight off-the-shelf models, covering specialized domains like math and coding. Additional details are provided in the Appendix.

Table 3: Component ablation on four standard datasets.

MoR	ACS	Residual Aggregation	MATH-500	MBPP	AIME	LiveCodeBench
✗	✗	✗	90.80	80.00	56.67	40.12
✗	✓	✗	91.20	80.20	60.00	43.16
✓	✗	✗	91.40	80.20	63.33	43.77
✗	✓	✓	91.80	80.60	66.67	44.38
✓	✓	✗	92.40	83.00	70.00	51.67
✓	✓	✓	93.00	83.20	70.00	52.28

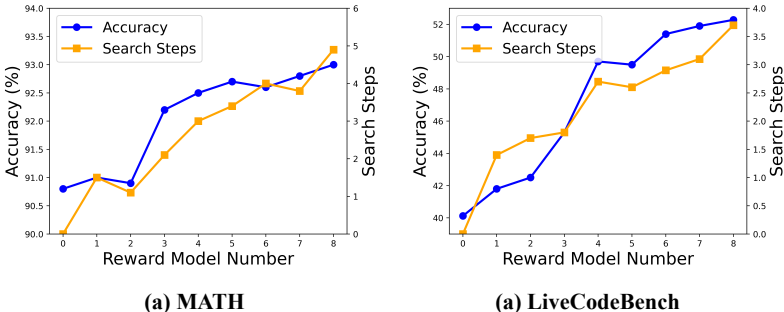


Figure 4: The curve of scaling capability of Reward Models.

4.2 ANALYSIS ON DIFFERENT TTS PARADIGMS

To thoroughly investigate CTTS and STTS paradigms, we conduct exploratory experiments measuring performance variation among different paradigms. As demonstrated in Table 1, CTTS paradigms outperform both STTS and single-model baselines under most settings. For example, under MA-MR setting, our method achieves an average improvement of 10.84% (80.11% v.s. 69.27%) over the best performance in the SA-SR setting, with gains of 5.8% (83.20% v.s. 77.40%) on MBPP, 12.20% (93.00% v.s. 80.80%) on MATH, and 13.13% (64.14 vs. 51.01) on GPQA, respectively. Under MA-SR setting, the best performance shows an average improvement of 8.94% over SA-SR while SA-MR yields an average gain of 1.44%. The results indicate the effectiveness of our proposed CTTS paradigm. In particular, our MA-MR based framework achieves superior improvements. Besides, we observe that under the SA-MR setting, a fixed combination of reward models may lead to performance degradation compared to SA-SR. This indicates that naively combining multiple reward models is unlikely to yield improvements and can even bring performance dropping. Such results are expected since most reward models are domain-specific. Fixed combination cannot guarantee consistent gains across all problems from different domains, which underscores the necessity of our proposed MoR for adaptive reward model selection. Comparative results between MA-MR and MA-SR also demonstrate that multi-RM achieves superior generalization compared to domain-specific reward models. Refer to our Appendix for more results.

4.3 MAIN RESULTS

As demonstrated in Table 2, our proposed CTTS-MM demonstrates significant improvements across seven diverse benchmarks. Through comprehensive comparisons with (1) fifteen representative open-source models (2) four leading Proprietary models and (3) five existing collaboration methods, our approach demonstrates consistent and substantial improvements across all evaluation dimensions. Our framework achieves 78.84% average accuracy on seven benchmarks. Compared to existing collaboration approaches, CTTS-MM outperforms Majority Voting Chen et al. (2024c) by +13.34%, MAV Lifshitz et al. (2025) by +8.73%, Symbolic-MoE Chen et al. (2025) by +6.02%, MoA Wang et al. (2025) by 7.09%, Self Consistency Chen et al. (2024e) by 7.68% and Best of N Snell et al. (2025) by 4.83%. Remarkably, our approach demonstrates superior performance compared to strong baselines from open-source LLM agents, proprietary LLM agents, and related multi-agent methods. Specifically, on average accuracy, CTTS-MM surpasses the best-performing open-source LLM GLM-Z1-32B by 5.34%, proprietary LLM GPT-4.1 by 7.06%, and STTS-based method Best-of-N by 4.82%. These results demonstrate that our CTTS-MM can effectively leverage the complementary advantages of multiple agents and multiple reward models, leading to a superior performance increase. This further validates the substantial potential of the CTTS paradigm, par-

Table 4: Comparative results of average inference time on MATH-500 and LiveCodeBench.

Method	MATH-500		LiveCodeBench	
	Inference Time (s)	Accuracy (%)	Inference Time (s)	Accuracy (%)
Multi-Agent Verification	22.45	76.30	24.52	42.55
Symbolic-MoE	15.05	90.40	17.76	43.16
MoA	19.54	87.80	21.97	40.12
Best of N	13.12	90.80	16.17	44.98
CTTS-MM (Ours)	20.14	93.00	22.04	52.28

ticularly highlighting the MA-MR framework’s robust capability in multiple domains. Refer to our Appendix for more results.

4.4 ABLATION STUDY

We perform a comprehensive component-wise ablation study on four standard benchmarks to quantify the contribution of each component in our CTTS-MM framework. Note that Residual Aggregation can only be applied when Agent Collaboration Search (ACS) is utilized. As illustrated in Table 3, the baseline obtains 90.08% accuracy on MATH. Utilizing ACS and MoR improves performance by 0.4% and 0.6%, respectively, reaching 92.40% when combined. Further gains come from Residual Aggregation, which contributes an additional 0.6%. Similar improvements are observed on the other three benchmarks, indicating the effectiveness of each component in CTTS-MM.

4.5 ANALYSIS ON SCALING CAPABILITY OF RMs

To investigate the scalability of MoR, we conduct experiments measuring performance improvements with increasing numbers of reward models. As shown in Figure 4, the performance of CTTS-MM consistently improves with increasing number of reward models on both MATH and LiveCodeBench. For instance, on LiveCodeBench, CTTS-MM achieves an accuracy of approximately 41.8% with a single reward model. When the number of reward models increases to four, the accuracy improves to nearly 50%, and ultimately reaches 52.28% with all eight reward models. Moreover, we observe that the search step also gradually increases with more reward models. It indicates that with the increase of reward models, our Mixture of Reward Model can enhance the entire model pool to extract cross-domain information, leading to more accurate reward scores for question-answer pairs during search. It guides the search process in a more optimal direction. In contrast, when the RM pool is limited in size, its robustness is weaker, resulting in inaccurate reward scores. This can mislead the search direction, potentially causing early stop and local optimal solutions.

4.6 ANALYSIS ON TIME AND COST EFFICIENCY

Although we primarily focus on exploring the optimal paradigm under collective test-time scaling, we further investigate the inference time and cost efficiency of our CTTS-MM in comparison to other related methods. For time efficiency, we report average inference time per response on MATH-500 and LiveCodeBench. As shown in Table 4, CTTS-MM achieves a significant improvement in accuracy with bearable inference time compared to other methods. For cost efficiency, we plot results of performance against inference cost. Refer to Section E of our Appendix for the detailed figure.

5 CONCLUSION

In this manuscript, we first explore Collective Test-Time Scaling (CTTS). We propose and investigate three CTTS paradigms: SA-MR, MA-SR and MA-MR. Experiments demonstrate that CTTS outperforms previous Single TTS paradigms, while the MA-MR variant consistently achieves superior performance. Based on it, we further propose a CTTS framework called CTTS-MM. To search for optimal agent ensembles, we propose Agent Collaboration Search approach. For adaptively selecting multiple reward models, Prior Reward Models Ensemble Selection is proposed. Experiments on seven benchmarks verify the superiority of CTTS-MM, revealing the strong potential of CTTS.

ETHICS STATEMENT

All authors have read and adhered to the ICLR Code of Ethics. This work involves no human subjects, sensitive personal data, or potentially harmful applications. All datasets used in our experiments are publicly available from official sources, and no proprietary or private data are involved. The authors declare that there are no potential conflicts of interest related to this work.

REPRODUCIBILITY STATEMENT

To support reproducibility, we provide a complete anonymous codebase in the supplementary material. The repository contains the implementation of our proposed CTTS-MM, the evaluation pipeline, and an associate example dataset, along with a README that provides installation and usage instructions. Besides, in our Appendix, Section B contains more details on our experiment datasets, Section C contains more details on LLMs and reward models we use for the experiment while Section D contains our implementation details. Together, these resources enable reliable reproduction of our experiments and results.

REFERENCES

- Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.
- Akhiad Bercovich, Itay Levy, Izik Golán, Mohammad Dabbah, Ran El-Yaniv, Omri Puny, Ido Galil, Zach Moshe, Tomer Ronen, Najeeb Nabwani, et al. Llama-nemotron: Efficient reasoning models. *arXiv preprint arXiv:2505.00949*, 2025.
- Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher R’e, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *ArXiv*, abs/2407.21787, 2024.
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Ma teusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *ArXiv*, abs/2005.14165, 2020.
- Zheng Cai, Maosong Cao, Haojiong Chen, Kai Chen, Keyu Chen, Xin Chen, Xun Chen, Zehui Chen, Zhi Chen, Pei Chu, Xiaoyi Dong, Haodong Duan, Qi Fan, Zhaoye Fei, Yang Gao, Jiaye Ge, Chenya Gu, Yuzhe Gu, Tao Gui, Aijia Guo, Qipeng Guo, Conghui He, Yingfan Hu, Ting Huang, Tao Jiang, Penglong Jiao, Zhenjiang Jin, Zhikai Lei, Jiaying Li, Jingwen Li, Linyang Li, Shuaibin Li, Wei Li, Yining Li, Hongwei Liu, Jiangning Liu, Jiawei Hong, Kaiwen Liu, Kuikun Liu, Xiaoran Liu, Chengqi Lv, Haijun Lv, Kai Lv, Li Ma, Runyuan Ma, Zerun Ma, Wenchang Ning, Linke Ouyang, Jiantao Qiu, Yuan Qu, Fukai Shang, Yunfan Shao, Demin Song, Zifan Song, Zhihao Sui, Peng Sun, Yu Sun, Huanze Tang, Bin Wang, Guoteng Wang, Jiaqi Wang, Jiayu Wang, Rui Wang, Yudong Wang, Ziyi Wang, Xingjian Wei, Qizhen Weng, Fan Wu, Yingtong Xiong, Chao Xu, Ruiliang Xu, Hang Yan, Yirong Yan, Xiaogui Yang, Haochen Ye, Huaiyuan Ying, Jia Yu, Jing Yu, Yuhang Zang, Chuyu Zhang, Li Zhang, Pan Zhang, Peng Zhang, Ruijie Zhang, Shuo Zhang, Songyang Zhang, Wenjian Zhang, Wenwei Zhang, Xingcheng Zhang, Xinyue Zhang, Hui Zhao, Qian Zhao, Xiaomeng Zhao, Fengzhe Zhou, Zaida Zhou, Jingming Zhuo, Yicheng Zou, Xipeng Qiu, Yu Qiao, and Dahua Lin. Internlm2 technical report, 2024.
- Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. Benchmarking large language models in retrieval-augmented generation. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI’24/IAAI’24/EAAI’24. AAAI Press, 2024a. ISBN 978-1-57735-887-9.

- 540 Junying Chen, Zhenyang Cai, Ke Ji, Xidong Wang, Wanlong Liu, Rongsheng Wang, Jianye Hou,
541 and Benyou Wang. Huatuogpt-o1, towards medical complex reasoning with llms, 2024b.
542
- 543 Justin Chih-Yao Chen, Sukwon Yun, Elias Stengel-Eskin, Tianlong Chen, and Mohit Bansal. Sym-
544 bolic mixture-of-experts: Adaptive skill-based routing for heterogeneous reasoning. *ArXiv*,
545 abs/2503.05641, 2025.
- 546 Lingjiao Chen, Jared Quincy Davis, Boris Hanin, Peter Bailis, Ion Stoica, Matei Zaharia, and James
547 Zou. Are more LLM calls all you need? towards the scaling properties of compound AI systems.
548 In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024c.
549
- 550 Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while
551 reducing cost and improving performance. *Transactions on Machine Learning Research*, 2024d.
552 ISSN 2835-8856.
- 553 Shikai Chen. Ldl-reward-gemma-2-27b-v0.1, 2025. Accessed: 2025-02-15.
554
- 555 Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash,
556 Charles Sutton, Xuezhi Wang, and Denny Zhou. Universal self-consistency for large language
557 models. In *ICML 2024 Workshop on In-Context Learning*, 2024e.
- 558 DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning,
559 2025.
- 560 DeepSeek-AI and Daya Guo et al. Deepseek-r1: Incentivizing reasoning capability in llms via
561 reinforcement learning. *ArXiv*, abs/2501.12948, 2025.
562
- 563 Nicolai Dorka. Quantile regression for distributional reward models in rlhf. *ArXiv*, abs/2409.10164,
564 2024.
- 565 Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving
566 factuality and reasoning in language models through multiagent debate. In *Proceedings of the*
567 *41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.
568
- 569 Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu
570 Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, Hongning Wang, Jiadai Sun, Jiajie Zhang, Jiale Cheng,
571 Jiayi Gui, Jie Tang, Jing Zhang, Juanzi Li, Lei Zhao, Lindong Wu, Lucen Zhong, Mingdao Liu,
572 Minlie Huang, Peng Zhang, Qinkai Zheng, Rui Lu, Shuaiqi Duan, Shudan Zhang, Shulin Cao,
573 Shuxun Yang, Weng Lam Tam, Wenyi Zhao, Xie Liu, Xiao Xia, Xiaohan Zhang, Xiaotao Gu,
574 Xin Lv, Xinghan Liu, Xinyi Liu, Xinyue Yang, Xixuan Song, Xunkai Zhang, Yifan An, Yifan
575 Xu, Yilin Niu, Yuantao Yang, Yueyan Li, Yushi Bai, Yuxiao Dong, Zehan Qi, Zhaoyu Wang,
576 Zhen Yang, Zhengxiao Du, Zhenyu Hou, and Zihan Wang. Chatglm: A family of large language
577 models from glm-130b to glm-4 all tools, 2024.
- 578 Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad
579 Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd
580 of models. *arXiv preprint arXiv:2407.21783*, 2024.
- 581 Lin Gui, Cristina Garbacea, and Victor Veitch. BoNBon alignment for large language models and the
582 sweetness of best-of-n sampling. In *The Thirty-eighth Annual Conference on Neural Information*
583 *Processing Systems*, 2024.
584
- 585 Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song,
586 and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv*
587 *preprint arXiv:2103.03874*, 2021.
- 588 Binyuan Hui, Jian Yang, Zeyu Cui, Jiayi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang,
589 Bowen Yu, Kai Dang, et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*,
590 2024.
591
- 592 Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando
593 Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free
evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

- 594 junyou li, Qin Zhang, Yangbin Yu, QIANG FU, and Deheng Ye. More agents is all you need.
595 *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856.
596
- 597 Jihoon Kwon Sangmo Gu Yejin Kim, Minkyung Cho Jy-yong Sohn Chanyeol, Choi Junseong Kim,
598 and Seolhwa Lee. Linq-embed-mistral: Elevating text retrieval with improved gpt data through
599 task-specific control and quality refinement. linq ai research blog, 2024.
- 600 Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E.
601 Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model
602 serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating
603 Systems Principles*, 2023.
604
- 605 Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal,
606 Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe
607 Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the
608 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook,
609 NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.
- 610 LG AI Research. Exaone deep: Reasoning enhanced language models. *arXiv preprint
611 arXiv:2503.12524*, 2025.
612
- 613 Shalev Lifshitz, Sheila A. McIlraith, and Yilun Du. Multi-agent verification: Scaling test-time com-
614 pute with goal verifiers. In *ICLR 2025 Workshop on Modularity for Collaborative, Decentralized,
615 and Continual Deep Learning*, 2025.
- 616 Chris Liu, Liang Zeng, Jiakai Liu, Rui Yan, Jujie He, Chaojie Wang, Shuicheng Yan, Yang Liu, and
617 Yahui Zhou. Skywork-reward: Bag of tricks for reward modeling in llms. *ArXiv*, abs/2410.18451,
618 2024.
- 619 Chris Liu, Liang Zeng, Yuzhen Xiao, Jujie He, Jiakai Liu, Chaojie Wang, Rui Yan, Wei Shen, Fux-
620 iang Zhang, Jiacheng Xu, Yang Liu, and Yahui Zhou. Skywork-reward-v2: Scaling preference
621 data curation via human-ai synergy. 2025.
622
- 623 Keming Lu, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou.
624 Routing to the expert: Efficient reward-guided ensemble of large language models. In Kevin
625 Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North
626 American Chapter of the Association for Computational Linguistics: Human Language Technolo-
627 gies (Volume 1: Long Papers)*, pp. 1964–1974, Mexico City, Mexico, June 2024. Association for
628 Computational Linguistics.
- 629 MAA. American invitational mathematics examination. [https://maa.org/math-competitions/
630 american-invitational-mathematics-examination-aime.](https://maa.org/math-competitions/american-invitational-mathematics-examination-aime.), 2024.
631
- 632 Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri
633 Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad
634 Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine:
635 Iterative refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Pro-
636 cessing Systems*, 2023.
- 637 et al. Mark Chen. Evaluating large language models trained on code. *ArXiv*, abs/2107.03374, 2021.
638
- 639 OpenAI. Introducing gpt-4.1 in the api. *Accessed: 2025-05-07*, 2025.
640
- 641 Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. Yarn: Efficient context window
642 extension of large language models. *arXiv preprint arXiv:2309.00071*, 2023.
- 643 David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Di-
644 rani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a bench-
645 mark. In *First Conference on Language Modeling*, 2024.
646
- 647 Tal Shnitzer, Anthony Ou, Mírian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson,
and Mikhail Yurochkin. Large language model routing with benchmark datasets, 2024.

- 648 Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time com-
649 pute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth*
650 *International Conference on Learning Representations*, 2025.
- 651
- 652 Kv Aditya Srivatsa, Kaushal Maurya, and Ekaterina Kochmar. Harnessing the power of multiple
653 minds: Lessons learned from LLM routing. In Shabnam Tafreshi, Arjun Akula, João Sedoc,
654 Aleksandr Drozd, Anna Rogers, and Anna Rumshisky (eds.), *Proceedings of the Fifth Workshop*
655 *on Insights from Negative Results in NLP*, pp. 124–134, Mexico City, Mexico, June 2024. Asso-
656 ciation for Computational Linguistics.
- 657 Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya
658 Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open
659 models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.
- 660
- 661 Qwen Team. Qwen2.5: A party of foundation models, September 2024a.
- 662
- 663 Qwen Team. Qwq: Reflect deeply on the boundaries of the unknown, November 2024b.
- 664
- 665 Qwen Team. Qwq-32b: Embracing the power of reinforcement learning, March 2025.
- 666
- 667 Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée
668 Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Ar-
669 mand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation
670 language models. *ArXiv*, abs/2302.13971, 2023.
- 671
- 672 Junlin Wang, Jue WANG, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-agents en-
673 hances large language model capabilities. In *The Thirteenth International Conference on Learn-
674 ing Representations*, 2025.
- 675
- 676 Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha
677 Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language
678 models. In *The Eleventh International Conference on Learning Representations*, 2023.
- 679
- 680 Zhilin Wang, Alexander Bukharin, Olivier Delalleau, Daniel Egert, Gerald Shen, Jiaqi Zeng, Oleksii
681 Kuchaiev, and Yi Dong. Helpsteer2-preference: Complementing ratings with preferences. *ArXiv*,
682 abs/2410.01257, 2024a.
- 683
- 684 Zihan Wang, Xinzhang Liu, Shixuan Liu, Yitong Yao, Yuyao Huang, Zhongjiang He, Xuelong Li,
685 Yongxiang Li, Zhonghao Che, Zhaoxi Zhang, Yan Wang, Xin Wang, Luwen Pu, Huihan Xu,
686 Ruiyu Fang, Yu Zhao, Jie Zhang, Xiaomeng Huang, Zhilong Lu, Jiabin Peng, Wenjun Zheng,
687 Shiquan Wang, Bingkai Yang, Xuwei he, Zhuoru Jiang, Qiyi Xie, Yanhan Zhang, Zhongqiu Li,
688 Lingling Shi, Weiwei Fu, Yin Zhang, Zilu Huang, Sishi Xiong, Yuxiang Zhang, Chao Wang, and
689 Shuangyong Song. Telechat technical report, 2024b.
- 690
- 691 Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi,
692 Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language
693 models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances*
694 *in Neural Information Processing Systems*, 2022.
- 695
- 696 An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu,
697 Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu,
698 Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical ex-
699 pert model via self-improvement. *ArXiv*, abs/2409.12122, 2024a.
- 700
- 701 Qwen An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan
702 Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu,
703 Jianwei Zhang, Jianxin Yang, Jiabin Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu,
704 Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin,
705 Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yi-Chao Zhang,
706 Yunyang Wan, Yuqi Liu, Zeyu Cui, Zhenru Zhang, Zihan Qiu, Shanghaoran Quan, and Zekun
707 Wang. Qwen2.5 technical report. *ArXiv*, abs/2412.15115, 2024b.

702 Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R
703 Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Thirty-*
704 *seventh Conference on Neural Information Processing Systems, 2023*.
705

706 Huaye Zeng, Dongfu Jiang, Haozhe Wang, Ping Nie, Xiaotong Chen, and Wenhui Chen. Acecoder:
707 Acing coder rl via automated test-case synthesis. In *Annual Meeting of the Association for Com-*
708 *putational Linguistics, 2025*.

709 Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu,
710 Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical
711 reasoning. In *Annual Meeting of the Association for Computational Linguistics, 2025*.
712

713 Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny
714 Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint*
715 *arXiv:2311.07911, 2023*.
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755

APPENDIX

This supplementary document is organized as follows:

- Section A contains details on our use of Large Language Models.
- Section B contains more details on our experiment datasets.
- Section C contains more details on LLMs and reward models we use for the experiment.
- Section D contains our implementation details.
- Section E contains experiment results on cost efficiency.
- Section F contains more comparison experiment results.
- Section G contains results for the analysis on specific cases from our approaches.
- Section H contains experiment results for the analysis on question pool.
- Section I contains more details on our Agent Collaboration Search.
- Section J contains details on our prompts for each of the seven benchmarks.

A DETAILS ON THE USE OF LARGE LANGUAGE MODELS

In this paper, we employ a large language model (LLM) to assist our writing, primarily for aiding or polishing the paper and no other applications are included. Specifically, we use DeepSeek of chat-version via this website: <https://yuanbao.tencent.com/chat>. Our usage of the LLM is limited to the following purposes: (1) translating terms and sentences, and (2) refining the phrasing of the manuscript. Finally, we acknowledge the convenient user-interactive LLM service provided by Tencent Yuan Bao based on DeepSeek.

B DETAILS ON DATASET

In our experiments, we assess the effectiveness of our proposed CTTS-MM across seven diverse benchmarks covering mathematical reasoning, complex QA, instruction following, and code generation. Note that, except for HumanEval Mark Chen (2021), all datasets are split into test and validation sets, with the validation sets utilized to construct the question pool. For MBPP Austin et al. (2021), we retain the original test set and merge the training and validation sets to serve as the validation split. Specifically, the validation set consists of 464 samples while the test set contains 500 samples. For LiveCodeBench Jain et al. (2024), we utilize their v5 version as the test set, reserving v6 for validation. For MATH Hendrycks et al. (2021), we evaluate on the MATH-500 subset and randomly sample 1,000 samples from the original dataset for validation. For AIME MAA (2024), we use the 2024 competition problems as the test set, leveraging historical questions (1983–2023) for validation. For GPQA Rein et al. (2024), we adopt the diamond subset consisting of graduate-level science questions as the test set, with the rest used for validation. In the IFEval Zhou et al. (2023), 300 instruction-following samples are selected at random for testing, with 241 used for validation. Finally, for Human-eval, we simply use their original version for test split (164 samples) and no validation split is constructed as mentioned.

C DETAILS ON LLMs AND RMS

C.1 LLM USAGE

As we mentioned in our manuscript, we assemble a set of ten mid-sized open-source LLMs (ranging from 20B to 72B parameters) from diverse architecture families. Specifically, the selected LLMs include: Qwen2.5-32B-Instruct Team (2024a), Qwen-2.5-72B-Instruct Team (2024a), Qwen2.5-Coder-32B-Instruct Hui et al. (2024), GLM-Z1-32B-0414 GLM et al. (2024), DeepSeek-R1-Distill-Qwen-32B DeepSeek-AI (2025), DeepSeek-R1-Distill-Llama-70B DeepSeek-AI (2025), QwQ-32B Team (2024b), InternLM2.5-20B-Chat Cai et al. (2024), Llama-3.3-70B-Instruct Grattafiori et al. (2024), Llama-3.3-Nemotron-Super-49B-v1 Bercovich et al. (2025). Note that this pool of

10 LLMs primarily acts as multi-agent in our CTTS-MM framework and is utilized for comparative experiments on TTS paradigms. For comparison experiments against other methods, we additionally include five open-source models: Gemma-3-27b-it Team et al. (2024), TeleChat2-35B-32K Wang et al. (2024b), EXAONE-Deep-32B LG AI Research (2025), Qwen3-32B Team (2025), HuatuoGPT-o1-72B Chen et al. (2024b). Details are listed in Table E.

Table E: Details on utilized LLMs.

Name	Size	Type
TeleChat2-35B-32K	35B	Instruction-tuned
GLM-Z1-32B-0414	32B	Deep Thinking
Qwen-2.5-72B-Instruct	72B	Instruction-tuned
Llama-3.3-70B-Instruct	70B	Instruction-tuned
DeepSeek-R1-Distill-Llama-70B	70B	Deep Thinking
DeepSeek-R1-Distill-Qwen-32B	32B	Deep Thinking
Gemma-3-27b-it	27B	Instruction-tuned
Qwen2.5-Coder-32B-Instruct	32B	Instruction-tuned
Qwen3-32B	32B	Deep Thinking
Llama-3.3-Nemotron-Super-49B-v1	49B	Deep Thinking
Qwen2.5-32B-Instruct	32B	Instruction-tuned
QwQ-32B	32B	Deep Thinking
EXAONE-Deep-32B	32B	Deep Thinking
HuatuoGPT-o1-72B	72B	Deep Thinking
InternLM2.5-20B-Chat	20B	Instruction-tuned

C.2 RM USAGE

We collect eight off-the-shelf reward models for all our experiments. Specifically, the collected reward models include: Qwen2.5-Math-RM-72B Yang et al. (2024a), Qwen2.5-Math-PRM-7B Zhang et al. (2025), Skywork-Reward-Gemma-2-27B Liu et al. (2024), INF-ORM-Llama3.1-70B Wang et al. (2024a), LDL-Reward-Gemma-2-27B-v0.1 Chen (2025), AceCodeRM-32B Zeng et al. (2025), QRM-Gemma-2-27B Dorka (2024), Skywork-Reward-V2-Llama-3.1-8B-40M Liu et al. (2025). Details are listed in Table F

Table F: Details on utilized Reward Models.

Name	Size	Base Model	type
Qwen2.5-Math-RM-72B	72B	Qwen2.5-Math-72B	ORM
Qwen2.5-Math-PRM-7B	7B	Qwen2.5-Math-7B-Instruct	PRM
Skywork-Reward-Gemma-2-27B	27B	Gemma-2-27B-it	ORM
INF-ORM-Llama3.1-70B	70B	Llama-3.1-70B-Instruct	ORM
LDL-Reward-Gemma-2-27B-v0.1	27B	Gemma-2-27B-it	ORM
AceCodeRM-32B	32B	Qwen2.5-Coder-32B-Instruct	ORM
QRM-Gemma-2-27B	32B	Gemma-2-27B-it	ORM
Skywork-Reward-V2-Llama-3.1-8B-40M	8B	Llama-3.1-8B-Instruct	ORM

D IMPLEMENTATION DETAILS

D.1 INFERENCE DETAILS

All experiments are conducted under the same inference settings. We employ VLLM Kwon et al. (2023) as the backend for executing LLM inference. The sampling temperature is fixed at 0.7, and the output sequence is set to 8,192 tokens to prevent excessively long generations. A presence penalty of 1.05 is applied to discourage repetitive outputs. In cases where the input context exceeds the model’s token limit, we apply the YaRN method Peng et al. (2023) to extend the context

Table G: Comparison results of different TTS paradigms on MBPP, MATH-500 and GPQA. MR* means utilizing our proposed MoR to select reward models. Multi-agent* means utilizing ten chosen LLMs.

Setting	Model	Reward Model	Weight Method	MBPP	MATH-500	GPQA	Avg.
Single Agent	Qwen2.5-32B-Instruct	-	-	76.00	75.60	40.91	64.17
	Qwen2.5-72B-Instruct	-	-	75.80	78.80	45.45	66.68
	Llama-3.3-Nemotron-Super-49B-v1	-	-	65.40	75.20	48.48	63.03
	Llama-3.3-70B-Instruct	-	-	70.40	73.00	46.97	63.46
	DeepSeek-R1-Distill-Llama-70B	-	-	76.40	82.8	60.10	73.10
SA-SR	Qwen2.5-32B-Instruct	AceCodeRM-32B	-	77.40	78.2	47.47	67.69
	Qwen2.5-32B-Instruct	Qwen2.5-Math-RM-72B	-	77.00	78.80	46.46	67.42
	Qwen2.5-72B-Instruct	AceCodeRM-32B	-	76.60	80.20	51.01	69.27
	Qwen2.5-72B-Instruct	Qwen2.5-Math-RM-72B	-	76.00	80.80	50.51	69.10
	Llama-3.3-Nemotron-Super-49B-v1	AceCodeRM-32B	-	66.40	76.00	50.80	64.40
	Llama-3.3-Nemotron-Super-49B-v1	Qwen2.5-Math-RM-72B	-	65.80	76.80	50.00	64.20
	Llama-3.3-70B-Instruct	AceCodeRM-32B	-	71.20	73.40	48.00	64.20
	Llama-3.3-70B-Instruct	Qwen2.5-Math-RM-72B	-	70.80	73.80	47.47	64.02
	DeepSeek-R1-Distill-Llama-70B	AceCodeRM-32B	-	77.00	82.60	59.09	72.90
DeepSeek-R1-Distill-Llama-70B	Qwen2.5-Math-RM-72B	-	76.60	83.20	59.09	72.96	
SA-MR	Qwen2.5-32B-Instruct	AceCodeRM-32B+Qwen2.5-Math-RM-72B	softmax	76.60	78.20	48.48	67.76
	Qwen2.5-32B-Instruct	AceCodeRM-32B+Qwen2.5-Math-RM-72B	linear	76.60	78.00	47.80	67.47
	Qwen2.5-32B-Instruct	MR*	-	78.00	79.4	51.01	69.47
	Qwen2.5-72B-Instruct	AceCodeRM-32B+Qwen2.5-Math-RM-72B	softmax	76.80	80.20	51.51	69.50
	Qwen2.5-72B-Instruct	AceCodeRM-32B+Qwen2.5-Math-RM-72B	linear	77.00	80.20	52.02	69.74
	Qwen2.5-72B-Instruct	MR*	-	77.20	81.4	53.53	70.71
	Llama-3.3-Nemotron-Super-49B-v1	AceCodeRM-32B+Qwen2.5-Math-RM-72B	softmax	66.20	76.60	51.52	64.77
	Llama-3.3-Nemotron-Super-49B-v1	AceCodeRM-32B+Qwen2.5-Math-RM-72B	linear	66.20	76.40	51.52	64.70
	Llama-3.3-Nemotron-Super-49B-v1	MR*	-	66.80	76.80	54.55	66.05
	Llama-3.3-70B-Instruct	AceCodeRM-32B+Qwen2.5-Math-RM-72B	softmax	71.40	74.00	48.48	64.63
	Llama-3.3-70B-Instruct	AceCodeRM-32B+Qwen2.5-Math-RM-72B	linear	71.40	74.00	48.99	64.80
	Llama-3.3-70B-Instruct	MR*	-	72.00	74.40	49.49	65.30
	DeepSeek-R1-Distill-Llama-70B	AceCodeRM-32B+Qwen2.5-Math-RM-72B	softmax	76.60	83.00	60.10	73.23
DeepSeek-R1-Distill-Llama-70B	AceCodeRM-32B+Qwen2.5-Math-RM-72B	linear	76.80	83.20	60.10	73.36	
DeepSeek-R1-Distill-Llama-70B	MR*	-	77.20	83.60	60.60	73.80	
MA-SR	Multi-agent*	Skywork-Reward-V2-Llama-3.1-8B-40M	-	77.00	91.20	61.11	75.97
	Multi-agent*	Qwen2.5-Math-RM-72B	-	80.6	91.8	61.11	77.84
	Multi-agent*	LDL-Reward-Gemma-2-27B-v0.1	-	78.80	91.00	62.63	77.48
	Multi-agent*	AceCodeRM-32B	-	82.2	90.8	61.62	78.21
MA-MR (Proposed CTTS-MM)	Multi-agent*	MR*	-	83.20	93.00	64.14	80.11

Table H: Main results of CTTS-MM compared with the optimal setting of Best of N on five benchmarks.

Model	AIME	MATH-500	MBPP	LiveCodeBench	Human-eval	Avg
<i>Open-source LLMs</i>						
Qwen-2.5-72B-Instruct	16.70	78.80	75.80	26.10	78.66	55.21
DeepSeek-R1-Distill-Llama-70B	60.00	82.80	76.40	40.70	92.07	70.39
Llama-3.3-Nemotron-Super-49B-v1	16.70	75.20	65.40	28.00	84.76	54.01
QwQ-32B	46.70	87.80	81.80	38.60	92.07	69.39
InternLM2.5-20B-Chat	3.30	55.20	55.00	14.90	69.51	39.58
Gemma-3-27b-it	30.00	84.00	70.40	27.70	86.59	59.74
Qwen2.5-32b-Instruct	20.00	75.60	76.00	24.00	77.44	54.61
TeleChat2-35B-32K	10.00	70.00	70.00	19.50	73.17	48.53
EXAONE-Deep-32B	33.30	84.38	72.80	31.60	93.90	63.20
GLM-Z1-32B-0414	66.70	90.00	74.40	44.40	96.34	74.37
Llama-3.3-70B-Instruct	30.00	73.00	70.40	30.10	84.15	57.53
Qwen3-32B	53.30	88.00	50.60	33.40	90.85	63.23
Qwen2.5-Coder-32B-Instruct	16.70	73.60	78.00	27.70	87.80	56.76
HuatuoGPT-o1-72B	16.70	73.00	78.00	27.40	85.37	56.09
DeepSeek-R1-Distill-Qwen-32B	56.70	85.60	81.00	44.70	95.73	72.75
<i>Setting for Best of N</i>						
Baseline	66.70	90.8	75.00	44.99	96.34	74.77
Optimal Setting	66.70	90.8	82.20	46.20	96.34	76.45
<i>Ours v.s. Optimal Setting for Best of N</i>						
CTTS-MM(ours)	70.00	93.00	83.20	52.28	97.56	79.21
- v.s. Best of N Baseline	↑3.30	↑2.20	↑8.20	↑7.29	↑1.22	↑4.82
- v.s. Best of N Optimal Setting	↑3.30	↑2.20	↑1.00	↑6.08	↑1.22	↑2.76

window. For aggregator, we use Llama-3.3-70B-Instruct. For embedding computation, we adopt Linq-Embed-Mistral Kim et al. (2024) across all experiments, with a fixed embedding dimension of 4,096. For reward models, VLLM is also utilized as inference backend except for Qwen2.5-Math-PRM-7B and Skywork-Reward-V2-Llama-3.1-8B-40M (These two reward models are already fast enough using their huggingface version with Flash Attention). As for other RMs, official VLLM only support Qwen2.5-Math-RM-72B. For other RMs used, we implement their VLLM version by

Table I: PRR accuracy of different reward models on four validation datasets.

Reward Model	MBPP-Val	MATH-Val	AIME-Val	LiveCodeBench-Val	Avg
Skywork-Reward-Gemma-2-27B	61.83	51.37	50.53	49.19	53.23
LDL-Reward-Gemma-2-27B-v0.1	61.23	47.69	43.75	47.13	49.95
Skywork-Reward-V2-Llama-3.1-8B-40M	58.44	76.75	82.08	80.72	74.50
INF-ORM-Llama3.1-70B	66.98	51.51	47.43	49.29	53.80
Qwen2.5-Math-RM-72B	68.54	87.73	89.13	86.39	82.95
Qwen2.5-Math-PRM-7B	67.05	67.15	49.32	35.74	54.82
AceCodeRM-32B	75.00	78.73	75.83	88.48	79.51
QRM-Gemma-2-27B	61.98	49.64	45.46	53.08	52.54

ourself (This will be released along with our code). All reward models are set to bfloat16 while other configurations stick to their original settings.

D.2 HYPERPARAMETERS

For all experiments, we use the same hyperparameters to ensure fair comparison. Specifically, for greedy search process of our ACS, we set top $k = 2$ to initialize our search subset while the number of aggregating is set to 8. For expanding the reward model pool, we consider combinations involving 2 and 3 reward models under three weight method: softmax, linear and sum. The selection number k is set to 100 while the tolerance threshold coefficient $\gamma = 0.95$.

D.3 DETAILS ON RELATED METHODS

Besides comparing the performance of single LLMs, we also compare our CTTS-MM with five popular multi-LLMs collaboration methods, and the experimental settings are as follows: Symbolic-MoE Chen et al. (2025) retains its original model profiling and LLM selection framework while employing Llama-3.3-70B-Instruct for final response aggregation. MoA Wang et al. (2025) employs 15 LLMs as references, also utilizing Llama-3.3-70B-Instruct as the aggregator. For Self Consistency Chen et al. (2024e), we select the best LLM on the validation datasets of each benchmark to generate eight responses per query, respectively. Majority Voting Chen et al. (2024c) determines the final output through voting among 15 reference LLMs. For Best of N* Snell et al. (2025), N is set to 8. And we use GLM-Z1-32B-0414 as our base model which obtains the highest average accuracy in open-source LLMs while Qwen2.5-Math-RM-72B is utilized as reward model for choosing the best answer as it achieves the best average PRR accuracy as shown in Table I. Like Self Consistency, we also conduct experiments of stronger settings for Best of N, which is shown in Section F.

D.4 DETAILS ON THE EXPERIMENTS OF INFERENCE TIME

In the manuscript, we investigate the inference time of our CTTS-MM in comparison to other related methods. Here, we provide more details on the experiment setting. Specifically, we report the average inference time per response on MATH-500 and LiveCodeBench with 10 reference LLMs, each deployed on a server with eight A800 GPUs. Although compared with a single LLM, multi-agent systems basically require more LLM forward passes and need more computational resources, most of these forward passes, e.g., the inferences of different referencers and aggregating multiple times, are independent and can be parallelized, making the overall inference time primarily determined by the slowest LLM. For a fair comparison, we apply parallel inference acceleration for all related methods.

E RESULTS ON COST EFFICIENCY

In the manuscript, we discuss about our analysis on the cost efficiency of our method. Here, we provide qualitative results. As shown in Figure E, we observe that different variants of our CTTS-MM lie along this frontier, in contrast to proprietary LLMs such as GPT-4.1 and Claude Sonnet-3.5, as well as other multi-agent approaches like MOA Wang et al. (2025) and Symbolic-MoE Chen et al. (2025), which are not cost-optimal and incur higher expenses for comparable performance. When quality is the primary objective, CTTS-MM-L represents the best configuration. For scenarios

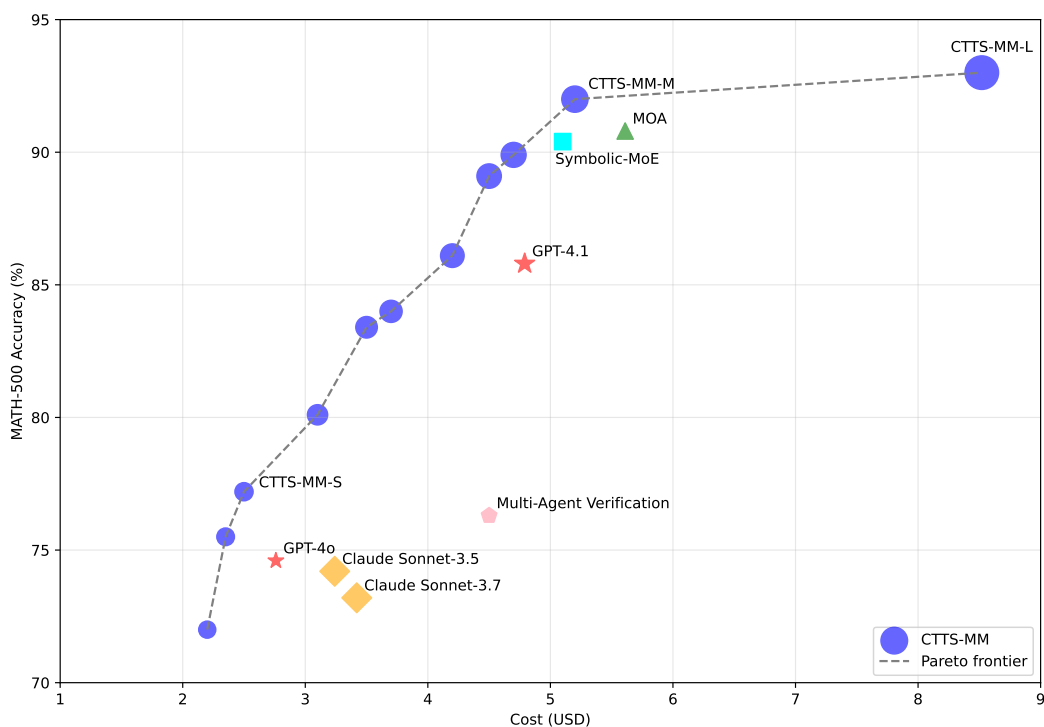


Figure E: Results of performance versus cost on MATH-500.

requiring a favorable trade-off between quality and cost, CTTS-MM-M achieves competitive cost with multi-agent based methods like MOA and Symbolic-MoE or GPT-4.1 while obtains higher level of quality. Notably, it outperforms GPT-4.1 by approximately 6% and Symbolic-MoE by around 2%.

F MORE EXPERIMENT RESULTS

F.1 COMPARISON RESULTS ON TTS PARADIGMS

In our manuscript, we conduct exploratory experiments measuring performance variation among different paradigms. Here, we present additional results on more base models and weight method in Table G. Results basically reveal the same conclusion on Llama-3.3-70B-Instruct and DeepSeek-R1-Distill-Llama-70B as CTTS paradigms outperform both STTS and single-model baselines under most settings.

F.2 COMPARISON RESULTS ON FIVE BENCHMARKS

We conduct additional experiments to compare our CTTS-MM with Best of N under its optimal setting on AIME, MATH-500, MBPP, LiveCodeBench and Human-eval. The results are shown in Table H. For the baseline of Best of N, we keep it the same with our manuscript, where we use GLM-Z1-32B-0414 as our base model which obtains the highest average accuracy in open-source LLMs while Qwen2.5-Math-RM-72B is utilized as reward model for choosing the best answer. As for Optimal Setting, we select the best open-source LLM on the validation datasets of each benchmark while the best reward model is utilized based on results from Table I. Note that we have no validation split on Human-eval, thus AceCodeRM-32B is utilized since it has best performance on coding benchmark. Results consistently show that our CTTS-MM superior performance increase, still outperforming the optimal setting of Best of N across all five benchmarks and by +2.76% on average accuracy.

Table J: Comparison results of cross-domain question pools.

Question Pool	AIME	MBPP	MATH-500	LiveCodeBench
MATH-Val	66.67	80.1	92.2	48.94
MBPP-Val	66.67	82.8	91.8	50.15
All (Seven Datasets)	70.00	83.2	93.00	52.28

G CASE STUDY

We further study the detailed case from our experiments. The detailed contents are provided in Figure F, G and H. We observe that our CTTS-MM is capable of deriving the correct answer through greedy search over the LLM pool using reward scores provided by MoR, even when only one or two models initially produce the correct answer. This result highlights the efficacy of our proposed search process guided by reward scores and underscores the crucial role of our MoR method in providing precise rewards.

H ANALYSIS ON QUESTION POOL

We further investigate the impact of utilizing question pools out of domains on overall performance. As shown in Table J, the performance degradation caused by employing out-of-domain question pools in MoR remains marginal. For instance, using an out-of-domain dataset (MBPP) as the question pool for evaluation on a math-related dataset (MATH) results in a marginal performance decrease by 0.4% compared to using an in-domain dataset as the question pool. Similar trends can be observed among other datasets, demonstrating the robustness and stability of our MoR approach. Moreover, when comparing against using a combined question pool from all datasets, we observe consistent performance improvements, highlighting the strong scalability of the question pool.

I MORE DETAILS ON ACS

Algorithm A shows the detailed algorithm of our Agent Collaboration Search. Our ACS employs a reward-guided greedy search with early stopping and residual aggregation. Given a set of n candidate answers, ACS first ranks them using a Mixture of Reward Models (MoR) and selects the top- k answers to initialize the search set. An aggregator then combines these into a composite answer, whose reward score is evaluated. Iteratively, ACS examines whether adding remaining candidates improves the aggregated answer. The search terminates when no further improvement is found. Finally, a residual aggregation step merges the best answer with all initial candidates to mitigate information loss, and the output with the higher reward is selected.

J DETAILS ON PROMPT

To obtain the optimal task-specific performance across heterogeneous benchmarks, we design prompt individually for each of the seven benchmarks, taking into account their unique characteristics, as shown in Figure I. Moreover, the design of the aggregator prompt within our CTTS-MM framework is refined based on MOA Wang et al. (2025), as illustrated in Figure J.

1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133

Algorithm A Greedy Search Paradigm of ACS

Require: Question q , LLM set D_A , An initial Answer Set A_0 , Reward Model set D_R , Question Pool Q_p , MOR Selective Fuction F , Search Steps T , Aggregator Agg , Initial Search Set Num k .

Ensure: The optimal answer A to question q .

```

1: for  $M$  in  $D_A$  do
2:    $A_0.add(M(q))$                                 {Initialize the answer set.}
3: end for
4:  $MOR \leftarrow F(q, Q_p, D_R)$                     {Select suitable reward models or their combinations.}
5:  $Score_0 \leftarrow MOR(A_0)$ 
6: for  $i = 1$  to  $T$  do
7:   if  $i == 0$  then
8:      $chosen\_index \leftarrow Score_0.topk(k).index$ 
9:      $best\_ans\_set \leftarrow A_0[chosen\_index]$     {Top k answers ranked by their reward scores}
10:    constitute the initial search subset.
11:     $current\_best \leftarrow Agg(best\_ans\_set)$     {Aggregate the above top k answers.}
12:     $current\_best\_score \leftarrow MOR(current\_best)$ 
13:   else
14:     if  $(A_0 - best\_ans\_set).empty()$  then
15:        $break$ 
16:     end if
17:      $improvement \leftarrow False$ 
18:      $ans\_to\_be\_searched \leftarrow A_0 - best\_ans\_set$ 
19:     for  $ans$  in  $ans\_to\_be\_searched$  do
20:        $temp\_candidate \leftarrow best\_ans\_set$     {Search for answers with improvement.}
21:        $temp\_candidate.add(ans)$ 
22:        $temp\_ans \leftarrow Agg(temp\_candidate)$ 
23:        $temp\_ans\_score \leftarrow MOR(temp\_ans)$ 
24:       if  $temp\_ans\_score \succ current\_best\_score$  then
25:          $best\_ans\_set \leftarrow temp\_candidate$ 
26:          $current\_best \leftarrow temp\_ans$ 
27:          $current\_best\_score \leftarrow temp\_ans\_score$ 
28:          $improvement \leftarrow True$ 
29:       end if
30:     end for
31:     if  $!improvement$  then
32:        $break$                                 {If no improvement, stop.}
33:     end if
34:   end if
35:   if  $residual$  then
36:      $residual\_set \leftarrow A_i \cup \{current\_best\}$     {Residual Aggregation}
37:      $residual\_answer \leftarrow Agg(residual\_set)$ 
38:      $residual\_score \leftarrow MoR(residual\_answe)$ 
39:     if  $residual\_score \succ current\_best\_score$  then
40:        $current\_best \leftarrow residual\_answer$ 
41:     end if
42:   end if
43:  $A \leftarrow current\_best$ 

```

1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187

Case 1 (AIME)

Question: Let $\triangle ABC$ be a triangle inscribed in circle ω . Let the tangents to ω at B and C intersect at point S , and let \overline{AD} intersect ω at P . If $SAB = 5^\circ$, $SBC = 9^\circ$, and $SAC = 10^\circ$, SAP can be written as the form $\frac{m}{n}$, where m and n are relatively prime integers. Find $m + n$.

Ground Truth: 113

Some Initial Answers from LLM Pool:

"Qwen2.5-32b-Instruct": "To solve for $\angle PAD$ in the given triangle $\triangle ABC$ inscribed in circle ω , we start by noting that D is the intersection of the tangents to ω at points B and C . This implies that $DB = DC$ and that $\angle DBA = \angle DCA = 90^\circ - \angle BAC$. Since AD is the symmedian of $\triangle ABC$, it follows that $\angle PAD = \angle PAB + \angle PBC$. Given the side lengths $AB = 5$, $BC = 9$, and $AC = 10$, we substitute these values into the formula for the length of a symmedian segment, which is given by $\frac{AB^2 + AC^2}{2BC}$. Thus, the length AD is $\frac{5^2 + 10^2}{2 \cdot 9} = \frac{125}{9}$. Since 125 and 9 are relatively prime, the fraction is in its simplest form. Therefore, $m = 125$ and $n = 9$, and the sum $m + n$ is $\boxed{134}$."

"Meta-Llama-3.3-70B-Instruct": ".....We omit some of the thinking steps to save space Taking the square root of both sides yields $SAP = \sqrt{50} = \frac{\sqrt{50}}{1} = \frac{\sqrt{50}}{1}$. However, since the answer needs to be in the form $\frac{m}{n}$ where m and n are relatively prime integers, and considering the specific request for the format of the final answer, we need to ensure our solution aligns with the problem's requirements. Step 11: Simplify AP to Fit the Required Format Given $SAP = \frac{\sqrt{50}}{1}$, to express it as a fraction with integers and to fit the format required for the solution, we recognize that the value of SAP itself doesn't need simplification in terms of removing the radical, but in terms of remembering the question asks for $m + n$ after expressing SAP in a certain form. Thus, $SAP = \frac{\sqrt{50}}{1}$ implies $m = 5$ and $n = 1$, but since m and n should be integers and relatively prime, the actual values we consider for m and n should directly relate to the fraction form requested, taking into account the simplification and the nature of the numbers involved. The final answer is: $\boxed{13}$."

"TeleChat2-35B-32K": ".....We omit some of the thinking steps to save space Using Symmedian Ratio: *n Using the ratio $\frac{PD}{AD} = \frac{r}{R}$, we get $\frac{PD}{AD} = \frac{r}{R}$. Then $\frac{PD}{AD} = \frac{r}{R}$. From the power of a point $\frac{PD}{AD} = \frac{r}{R}$. Calculate $\frac{PD}{AD}$ and $\frac{r}{R}$. Substitute back into $x = \frac{PD}{AD}$ and $y = \frac{r}{R}$ to find x and y and subsequently $\frac{PD}{AD}$. Solving this quadratic equation gives $y = \frac{3}{11}$ and $x = \frac{1}{11}$. Hence $\frac{PD}{AD} = \frac{3}{11}$ and $\frac{r}{R} = \frac{1}{11}$. Finally, $m + n = 3 + 1 = \boxed{4}$."

"GLM-Z1-32B-0414": "Okay, so I need to find the length of AP in triangle ABC inscribed in a circle. (.....We omit some of the thinking steps to save space) The sides are $a = BC = 9$, $b = AC = 10$, $c = AB = 5$. The semi-perimeter $s = (5+9+10)/2 = 24/2 = 12$. Then $\cos A = \frac{b^2 + c^2 - a^2}{2bc} = \frac{10^2 + 9^2 - 5^2}{2 \cdot 10 \cdot 9} = \frac{100 + 81 - 25}{180} = \frac{156}{180} = \frac{13}{15}$. Therefore, the circumradius $R = \frac{abc}{4\Delta} = \frac{9 \cdot 10 \cdot 5}{4 \cdot 156} = \frac{450}{624} = \frac{25}{42}$. Rationalizing the denominator: $\frac{25 \cdot 42}{42 \cdot 42} = \frac{1050}{1764}$. So $R = \frac{1050}{1764}$. But maybe I don't need the actual value of R. Let's see. Alternatively, since D is the intersection of the tangents at B and C, the distance $\frac{PD}{AD} = \frac{r}{R}$. Coordinates of A and P are $(0,0)$ and $(\frac{10}{13}, \frac{9}{13})$ respectively. $P = (\frac{10}{13}, \frac{9}{13})$. Distance $AP = \sqrt{(\frac{10}{13})^2 + (\frac{9}{13})^2} = \frac{\sqrt{100 + 81}}{13} = \frac{\sqrt{181}}{13}$. Hence $\frac{PD}{AD} = \frac{\sqrt{181}}{13}$. The final answer is $\boxed{113}$."

"gemma_3_27b_it": ".....We omit some of the intermediate answers to save space The final answer is $\boxed{486}$."

"Qwen3-32B": ".....We omit some of the intermediate answers to save space let me check calculations again to make sure. $\frac{PD}{AD}$ squared was calculated as $\frac{15.210.000}{507.00062}$. Then $\sqrt{\frac{15.210.000}{507.00062}} = \frac{3900}{507}$. Now $\frac{3900}{507} = \frac{1300}{169} = \frac{100}{13}$. Hence $\frac{PD}{AD} = \frac{100}{13}$. Correct. Therefore $AP = \frac{100}{13}$. So $m + n = 100 + 13 = 113$. Therefore the answer is $\boxed{113}$."

.....(answers from other LLM)

Search process of CTS-MM:

Win response from Search Step-1 (final reward score: 4.12): "## Step 1: Understand the given problem and recall the concept of power of a point. The problem involves a triangle $\triangle ABC$ inscribed in circle ω , with tangents at B and C intersecting at S , and \overline{AD} intersecting ω again at P . We are given the side lengths $SAB = 5^\circ$, $SBC = 9^\circ$, and $SAC = 10^\circ$. To find the length of SAP , we need to apply the power of a point theorem. Step 2: Apply the power of a point theorem to find SAP . By the power of a point theorem, $SDB^2 = DP \cdot DAS$. Let's denote SAP as x and SDB as y . Then, $SDP = x + 5$ and $SDA = 5 + x$. Substituting these expressions into the equation $SDB^2 = DP \cdot DAS$, we get $SDB^2 = (x + 5)(5 + x)$. Step 3: Use the given information about the triangle $\triangle ABC$ to determine SDB . Using the Law of Cosines on $\triangle ABC$, we can find $\cos A$. With $\cos A$ known, we can further calculate $\sin A$. Then, applying the Law of Sines to $\triangle ABD$, we can relate SAD and SDB . With SAD known, we can then solve for SDB . Step 4: Solve for SAP by equating the two expressions for SDB^2 . After obtaining SDB from the previous step, we equate it to $(x + 5)(5 + x)$ and solve for x , which represents SAP . Step 5: Express SAP in the form $\frac{m}{n}$ where m and n are relatively prime integers and find $m + n$. Once SAP is found, express it in the required form $\frac{m}{n}$, ensuring that m and n are relatively prime. Finally, compute $m + n$. The final answer is: $\boxed{113}$."

Win response from Search Step-2 (final reward score: 10.36): "## Step 1: Apply the Power of a Point Theorem To solve this problem, we will use the Power of a Point theorem. This theorem states that for any point S outside a circle with points of intersection SX and SYS with the circle, the product of the lengths $SPXS$ and $SPYS$ is constant for any line through S to the circle. In our case, we consider point S as S and apply the theorem to find a relationship between the lengths of segments intersecting the circle. Step 2: Find the relationship using Power of a Point. By Power of a Point from S , we have $SDB \cdot DC = DP \cdot DAS$. Since SDB and $SDCS$ are tangents from the same external point to the circle, and given $SAB = 5^\circ$, $SBC = 9^\circ$, $SAC = 10^\circ$, we need to leverage these to find expressions for SDB and $SDCS$. Step 3: Calculate DB and DC . Using the fact that in a triangle, the lengths of the tangents from an external point to the circle are equal, let's denote $Sx = DBS$ and $Sy = DCS$. We also know that $SDA = DP + PAS$. However, to directly apply Power of a Point, we recognize the need to express SDB and $SDCS$ in terms of SAB , SBC , and SAC , but directly calculating them requires additional relationships. Step 4: Use the Law of Sines to find SDB . Step 5: Simplify AP to Fit the Required Format Given $SAP = \frac{\sqrt{50}}{1}$, to express it as a fraction with integers and to fit the format required for the solution, we recognize that the value of SAP itself doesn't need simplification in terms of removing the radical, but we must remember the question asks for $m + n$ after expressing SAP in a certain form. Thus, $SAP = \frac{\sqrt{50}}{1}$ implies $m = 5$ and $n = 1$, but since m and n should be integers and relatively prime, the actual values we consider for m and n should directly relate to the fraction form requested, taking into account the simplification and the nature of the numbers involved. The final answer is: $\boxed{113}$."

.....(omit intermediate search steps (step-3 final score: 12.37, step-4 final score: 14.09))

Win response from Search Step-5 (final reward score: 16.25): "## Step 1: Apply the Power of a Point theorem The Power of a Point theorem states that for any point S outside a circle with tangents drawn and intersecting chords as well, the product of the lengths of one tangent segment and its external secant segment is equal to the square of the length of the tangent segment from S to the point of tangency. In this problem, we're given $SAB = 5^\circ$, $SBC = 9^\circ$, and $SAC = 10^\circ$. The theorem tells us $SDB^2 = DP \cdot DAS$. Step 2: Calculate SDB . First, let's calculate SDB . Because $\triangle ABC$ is a right triangle (given $SAB = 5^\circ$, $SBC = 9^\circ$, $SAC = 10^\circ$, which does not satisfy Pythagoras' theorem for an integer-sided right triangle but let's proceed with calculations), we can find the circumradius using the formula $SR = \frac{abc}{4\Delta}$ where Δ is the area of $\triangle ABC$. However, recognizing the Pythagorean triple $(5, 12, 13)$ and comparing with our sides, our triangle isn't a Pythagorean triple. Instead, we should directly apply the Power of a Point theorem after finding the necessary lengths. Step 3: Identify relevant relationships for $SDAS$ and $SDPS$. Since $SAP = DP - DAS$, we need to express $SDAS$ and $SDPS$ in terms of known quantities or find a direct way to calculate SAP . To do so, let's utilize the properties of inscribed angles, central angles, and the fact that S is the intersection of tangents. Step 4: Calculate $SDAS$ or $SDPS$. To proceed further, let's recall the relation $SDB^2 = DA \cdot DP$. Without loss of generality, assume we have SDB and need $SDAS$ or $SDPS$. Knowing SAB , SBC , and SAC , and that S is external to ω , we might relate these lengths through similar triangles or trigonometric ratios, considering the angles formed at S and within $\triangle ABC$. Step 5: Consider alternative approaches to find SAP . Given the complexities in calculating SDB and $SDAS$ or $SDPS$ directly from previous steps without explicit angle measures or side lengths involving S , consider leveraging geometric properties such as similar triangles, the intersecting chord theorem, or utilizing the Law of Cosines and Law of Sines to derive relationships between the segments. Step 6: Reflect on provided solutions for insight. Considering the complexity of direct calculation without specific trigonometric values or additional geometric constructions, reflect on the approach provided in the rationale which calculates SAP using the identified relationship $SAP = \frac{\sqrt{100}}{13}$ by leveraging geometric principles and the Power of a Point theorem correctly applied to the given configuration. The final answer is: $\boxed{113}$."

Win response from Search Step-6 (final reward score: 11.43):

The reward score decrease, Stop Here. The final answer will be the one from Step-5

Figure F: Detailed case results from AIME.

1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295

```

Case 3 (MBPP)

Question: Write a function to get the angle of a complex number. Your code should satisfy the following assertion with the same function name:
python assert angle_complex(0,1j)==1.5707963267948966

Some Initial Answers from LLM Pool:

Qwen2.5-32b-Instruct (exec result: false): "To calculate the angle of a complex number, you can use the 'cmath' library in Python, which provides a function to find the phase (angle) of a complex number. Here is how you can write the function:
python import cmath
def angle_complex(real, imag):
    # Create a complex number from real and imaginary parts
    complex_num = complex(real, imag)
    # Calculate the angle (phase) of the complex number
    angle = cmath.phase(complex_num)
    return angle
Test the function with the provided assertion:
assert angle_complex(0, 1j) == 1.5707963267948966
In this code, 'complex(real, imag)' creates a complex number from the real and imaginary parts. 'cmath.phase(complex_num)' calculates the phase (angle) of the complex number in radians."

Meta-Llama-3.3-70B-Instruct (exec result: false): "Of course! I'll guide you through creating a function to calculate the angle of a complex number, ensuring it meets the provided assertion. Here is the Python implementation:
python import math
def angle_complex(real, imag):
    return math.atan2(imag, real)
Explanation: The function 'angle_complex' takes two parameters, 'real' and 'imag', representing the real and imaginary parts of a complex number, respectively. It utilizes the 'math.atan2' function from Python's math library, which returns the arc tangent of 'imag' divided by 'real' in radians. The 'atan2' function is used instead of 'atan' to handle the case where 'real' is zero and to get the correct quadrant. The 'math.atan2' function automatically handles the signs of 'real' and 'imag' to determine the correct angle in the range of  $[-\pi, \pi]$ . The calculated angle is then returned by the 'angle_complex' function, satisfying the given assertion."

TeleChat2-35B-32K (exec result: false): "To get the angle of a complex number in Python, you can use the 'cmath' module, which provides functions to work with complex numbers. The 'cmath.phase' function returns the phase (angle) of a complex number in radians. Here is the function that satisfies the given assertion:
python import cmath
def angle_complex(complex_num):
    return cmath.phase(complex_num)
Test the function with the given assertion:
assert angle_complex(0+1j) == 1.5707963267948966
This function takes a complex number as input and returns its angle in radians. The provided assertion checks that the function correctly computes the angle of the complex number  $\sqrt{0 + 1j}$ , which should be  $\frac{\pi}{2}$  radians or approximately 1.5707963267948966."

GLM-Z1-32B-0414 (exec result: true): "Okay, I need to write a Python function called angle_complex (.....We omit some of the intermediate answers to save space) The 'cmath.phase()' function returns the angle in radians between the positive real axis and the complex number in the complex plane. Solution Code:
python import cmath
def angle_complex(real, imag):
    z = real + imag * 1j
    return cmath.phase(z)
Explanation: Combining Real and Imaginary Parts: The function takes two arguments, 'real' and 'imag'. By adding these two arguments directly, we form the complex number. For example, if 'real' is 0 and 'imag' is 1j, the result of 'real + imag' is the complex number '0 + 1j'. Calculating the Phase: The 'cmath.phase()' function is used to compute the angle of the complex number. This function handles all edge cases, including when the real part is zero, ensuring the correct angle is returned. This approach efficiently constructs the complex number and computes its angle using Python's built-in capabilities, ensuring accuracy and reliability."

gemma_3_27b_it (exec result: false): "python import cmath
def angle_complex(x, y):
    """Calculates the angle of a complex number.
    Args:
        x: The real part of the complex number.
        y: The imaginary part of the complex number.
    Returns:
        The angle of the complex number in radians.
    """
    return cmath.phase(complex(x, y))

Owen3-32B (exec result: false): "think>Okay, I need to write a Python function called angle_complex (.....We omit some of the intermediate answers to save space) Implementation: Here is the implementation of the 'angle_complex' function:
python import cmath
def angle_complex(a, b):
    """Returns the angle (in radians) of a complex number constructed from the real part of 'a' and the imaginary part of 'b'.
    Parameters:
        a (complex or real): Real part of the complex number.
        b (complex or real): Imaginary part of the complex number.
    Returns:
        float: The phase (angle) of the resulting complex number.
    """
    real = a.real
    imag = b.imag
    return cmath.phase(complex(real, imag))
Explanation: Extract the real component from the first argument. Extract the imaginary component (the coefficient of 'j') from the second argument. Construct the complex number 'real + imag*j'. 'cmath.phase()' computes the angle (argument) of the complex number in radians. Test Case:
python angle_complex(0, 1j) == 1.5707963267948966
This satisfies the provided assertion and works as expected. Edge Case Handling: Note: If both 'real' and 'imag' are zero, 'cmath.phase(complex(0,0))' will raise a 'ValueError'. It's important to ensure the function is used with valid inputs representing a non-zero complex number."

.....(answers from other LLM)

Search process of CTS-MM:

Win response (extracted solution) from Search Step-1 (final reward score: 1.3, exec result: false): "python import math
def angle_complex(real, imag):
    # Check if the input is a complex number
    if isinstance(imag, complex):
        imag = imag.imag
    return math.atan2(imag, real)"

Win response from Search Step-2 (final reward score: 5.27, exec result: false): "python import math
def angle_complex(real, imag):
    z = real + imag * 1j
    return math.atan2(z.imag, z.real)"

Win response from Search Step-3 (final reward score: 16.57, exec result: true): "python import cmath
def angle_complex(real, imag):
    # Combine real and imag into a complex number
    z = real + imag * 1j
    # Return the phase (angle) of the complex number
    return cmath.phase(z)
Test the function with the given assertion:
assert angle_complex(0, 1j) == 1.5707963267948966"

Win response from Search Step-4 (final reward score: 16.57, exec result: true): "python import cmath
def angle_complex(real, imag):
    # Combine real and imag into a complex number
    z = real + imag * 1j
    # Return the phase (angle) of the complex number
    return cmath.phase(z)
Test the function with the given assertion:
assert angle_complex(0, 1j) == 1.5707963267948966"

The reward score ceases increasing. Stop Here. The final answer will be the one from Step-3

```

Figure H: Detailed case results from MBPP.

1296	Prompt for MBPP benchmark System Prompt: "You are an exceptionally intelligent coding assistant that consistently delivers accurate and reliable responses to user instructions." User Prompt: "Question: {question}."
1297	
1298	
1299	
1300	
1301	
1302	Prompt for LiveCodeBench benchmark System Prompt: "You are an expert Python programmer. You will be given a question (problem specification) and will generate a correct Python program that matches the specification and passes all tests." User Prompt: "Question: {question}."
1303	
1304	
1305	
1306	
1307	
1308	Prompt for Human-eval benchmark System Prompt: "You are an expert Python programmer. You will be given a coding question (problem specification) and will generate a correct Python program that matches the specification and passes all tests. Directly give the executable function body, without any comments or test cases." User Prompt: "Question: {question}."
1309	
1310	
1311	
1312	
1313	
1314	Prompt for AIME benchmark System Prompt: "Please reason step by step, and put your final answer within <code>\\boxed{}</code> ." User Prompt: "Question: {question}."
1315	
1316	
1317	
1318	
1319	
1320	Prompt for MATH benchmark System Prompt: "You are a math problem solver. Please solve the following math problem. Be sure to explain your solution in detail. The numerical values in the answer should be surrounded by <code>\\boxed</code> . The final answer should start with 'The answer is' and give the conclusion directly. Do not add any extra content." User Prompt: "Question: {question}."
1321	
1322	
1323	
1324	
1325	
1326	Prompt for GPQA benchmark System Prompt: "You are a very intelligent assistant, who follows instructions directly." User Prompt: "Question: {question}."
1327	
1328	
1329	
1330	
1331	
1332	Prompt for IFEval benchmark User Prompt: "Instruction: {question}."
1333	
1334	
1335	
1336	
1337	
1338	
1339	
1340	
1341	
1342	
1343	
1344	
1345	
1346	
1347	
1348	
1349	

Figure I: Prompts for seven benchmarks.

1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403

Prompt for Aggregator

System Prompt: "You have been provided with a set of responses from various open-source models to the latest user query. Your task is to synthesize these responses into a single, high-quality response. It is crucial to critically evaluate the information provided in these responses, recognizing that some of it may be biased or incorrect. Your response should not simply replicate the given answers but should offer a refined, accurate, and comprehensive reply to the instruction. Ensure your response is well-structured, coherent, and adheres to the highest standards of accuracy and reliability.

Responses from models:

- 1. {Response1}
- 2. {Response2}
- ...

User Prompt: "Question: {question}."

Figure J: Prompt for Aggregator within our CCTS-MM