

Your Reward Function for RL is Your Best PRM for Search: Unifying RL and Search-Based TTS

Can Jin¹ Yang Zhou¹ Qixin Zhang² Hongwu Peng³ Di Zhang⁴
 Zihan Dong¹ Marco Pavone⁵ Ligong Han^{6,7} Zhang-Wei Hong⁸
 Tong Che^{5,†} Dimitris N. Metaxas^{1,†}

¹Rutgers University ²Nanyang Technological University ³University of Connecticut
⁴Fudan University ⁵NVIDIA Research ⁶Red Hat AI Innovation
⁷MIT-IBM Watson AI Lab ⁸Massachusetts Institute of Technology

Abstract

Test-time scaling (TTS) for large language models (LLMs) has thus far fallen into two largely separate paradigms: (1) reinforcement learning (RL) methods that optimize *sparse* outcome-based rewards, yet suffer from instability and low sample efficiency; and (2) search-based techniques guided by independently trained, *static* process reward models (PRMs), which require expensive human- or LLM-generated labels and often degrade under distribution shifts. In this paper, we introduce AIRL-S, the *first natural unification* of RL-based and search-based TTS. Central to AIRL-S is the insight that the reward function learned during RL training inherently represents the ideal PRM for guiding downstream search. Specifically, we leverage adversarial inverse reinforcement learning (AIRL) combined with group relative policy optimization (GRPO) to learn a *dense, dynamic* PRM directly from correct reasoning traces, entirely eliminating the need for labeled intermediate process data. At inference, the resulting PRM simultaneously serves as the critic for RL rollouts *and* as a heuristic to effectively guide search procedures, facilitating robust reasoning chain extension, mitigating reward hacking, and enhancing cross-task generalization. Experimental results across eight benchmarks, including mathematics, scientific reasoning, and code generation, demonstrate that our unified approach improves performance by **9%** on average over the base model, matching GPT-4o. Furthermore, when integrated into multiple search algorithms, our PRM consistently outperforms all baseline PRMs trained with labeled data. These results underscore that, indeed, *your reward function for RL is your best PRM for search*, providing a robust and cost-effective solution to complex reasoning tasks in LLMs.

1 Introduction

Recently, test-time scaling (TTS) has been explored as an effective method to enhance the reasoning performance of large language models (LLMs) [20, 27, 28, 30, 48, 52, 62, 74, 85]. Specifically, reinforcement learning (RL) methods [20, 52, 60] and search strategies such as Monte Carlo Tree Search (MCTS), beam search, and Best-of-N sampling have been adopted to support TTS on complex reasoning benchmarks [62, 75, 82, 83, 85]. Notably, OpenAI’s *o*-series models [52] and DeepSeek-R1 [20] demonstrate that large-scale RL training can lengthen and refine the chains of thought (CoT) produced at inference time. The RL training of LLMs is generally guided by outcome reward models (ORMs) [9, 20, 78] and process reward models (PRMs) [35, 37, 60, 68, 69], which provide supervisory signals to improve model performance.

[†]Equal advising, Correspondence to: Can Jin <can.jin@rutgers.edu>, Tong Che <tongc@nvidia.com>.

Although DeepSeek-R1 achieves strong TTS performance using only ORMs or rule-based rewards during RL training [20], sparse outcome rewards often degrade training stability and sample efficiency [6, 12, 37, 79]. Search-based TTS methods that use static PRMs trained on labeled step-wise datasets can guide test-time search and improve reasoning performance, but building fine-grained PRMs that score each intermediate reasoning step requires extensive human annotation or LLM-generated pseudo-labels [12, 37, 69]. Furthermore, distributional shifts between a fixed PRM and the continually updated policy can lead to reward hacking [3, 16, 47], thereby undermining the stability and effectiveness of the policy model. Additionally, the reward hacking can also limit the effectiveness of PRMs in search-based TTS, where the PRM serves as a verifier during test-time search.

In this paper, we investigate *how to effectively combine RL-based and Search-based TTS in complex reasoning*. Specifically, to reduce the cost of training high-quality PRMs and alleviate reward hacking risks from static PRMs trained on separate datasets during test time search, we propose AIRL-S, a framework that integrates adversarial inverse reinforcement learning (AIRL) [13, 14] with group relative policy optimization (GRPO) [12, 20] to support long CoT reasoning in LLMs through both RL-based and search-based TTS. During training, AIRL learns a step-wise PRM from the reference rollouts. The policy model is then updated using the combined objectives of the dense rewards in AIRL and the binary outcome rewards in GRPO. During inference, the PRM naturally serves as a verifier that guides the search procedure, extending the reasoning of the policy model. AIRL-S enables the training of generalizable PRMs without requiring any labeled process reward data, thereby significantly reducing the cost of constructing dense reward models and mitigating reward hacking under distributional shift. Furthermore, the PRM is theoretically invariant to environmental dynamics [14], allowing it to be reused during inference across different datasets and policy models in search-based TTS.

We evaluate the effectiveness of the LLM policy and the PRM trained with AIRL-S on eight standard reasoning benchmarks spanning mathematics, science, and code generation. The policy model achieves an average performance improvement of **9%** over the base model and matches GPT-4o across these tasks. When combined with search-based TTS methods, the PRM surpasses PRMs trained on labeled process data across multiple policy models and datasets. We further pair the PRM with widely used test-time search algorithms to demonstrate its compatibility and effectiveness under varied TTS configurations. Overall, AIRL-S provides an effective and cost-efficient approach for scaling test-time computation in LLMs for complex reasoning tasks.

2 Related Works

Inverse Reinforcement Learning. Inverse reinforcement learning (IRL) [1, 13, 14, 40, 41, 58] aims to recover reward functions from expert demonstrations, enabling subsequent policy training through standard RL methods. Classic IRL methods include maximum-margin IRL [1, 58] and probabilistic maximum-entropy IRL [86, 87]. Adversarial IRL (AIRL) [14] reformulated IRL into a GAN-style adversarial game [18], improving generalization and theoretical grounding. Subsequent advances have further streamlined IRL: IQ-Learn optimizes inverse soft- Q functions without explicitly modeling rewards [17]; Inverse Preference Learning uses offline pairwise preferences to avoid explicit reward modeling [21]; "IRL without RL" reframes imitation as supervised regression over trajectory data [65]. Our method leverages AIRL’s disentanglement of reward and policy, extending it to RL for large language models (LLMs) and producing a generalizable reward function suitable for guiding test-time search.

RL for LLMs. Reinforcement learning from human feedback (RLHF) is standard for aligning LLM outputs with user intent [8, 54]. Nonetheless, many open-source reasoning improvement efforts still rely on imitation learning from curated Chain-of-Thought (CoT) datasets [29, 42, 43, 48, 66, 80, 81, 84]. Recent large-scale RL methods using sparse outcome rewards, such as OpenAI o1 [51] and DeepSeek-R1 [20], have achieved significant gains. Specialized RL frameworks for mathematical reasoning, such as Math-Shepherd [69] and DeepSeek-Math-7B-RL [12], utilize dense supervision or extensive RL training to match larger models. Concurrently, PRIME [11] employs "free process rewards" [61] to implicitly derive a token-level process reward from log-likelihood ratios between two LLMs. PRIME is closely related to our work, sharing the fundamental insight of utilizing an implicitly learned reward function from RL training. However, PRIME differs crucially by:

- (a) Employing per-token rewards derived from log-likelihood ratios, which reward-guided generation literatures (discrete GANs, human preference modeling, generation quality evaluation etc) [39, 50, 57] suggests is much less effective than our holistic step-wise discriminators.
- (b) Producing a policy-dependent reward function unsuitable for training new policies or guiding external search procedures.

In contrast, our AIRL-based framework yields an actor-independent reward function capable of both optimal policy recovery (as shown theoretically in AIRL [14]) and direct use as a PRM for guiding search algorithms across different LLMs.

Test-Time Scaling. Test-time scaling (TTS) methods enhance reasoning capabilities of fixed LLMs by allocating additional inference-time computation. Parallel TTS approaches aggregate independent samples to improve outcomes [4, 25, 70, 83]. Methods such as Self-Consistency [70], Best-of-N sampling [5, 62], and Beam Search [62, 75] utilize diversity-driven strategies for output selection. Monte-Carlo Tree Search (MCTS) integrates lookahead search guided by learned PRMs, achieving strong reasoning performance [55, 73, 76, 82, 83, 85]. Sequential TTS refines outputs iteratively based on previous attempts [44, 48, 62]. Our work integrates the AIRL-trained PRM directly into popular TTS methods, including MCTS, Beam Search, and Best-of-N sampling, demonstrating superior performance compared to static PRMs trained from labeled data.

3 Method

Our objective is to learn a generalizable step-wise PRM that benefits both RL training and test-time search, thereby improving the reasoning accuracy of LLMs. We adopt AIRL and GRPO to train the PRM and the policy model jointly. The learned PRM then guides the search procedure of the policy LLM during inference, yielding additional gains in performance. The detailed pseudo-code for RL training in AIRL-S is presented in Algorithm 1.

3.1 Problem Setup

Let $\mathcal{Q}$ be a dataset of questions and $q \in \mathcal{Q}$ a specific question. From the reference rollouts, we obtain a CoT,

$$C(q) = \{C_1, C_2, \dots, C_T\},$$

where each C_i is a reasoning step that leads to the final answer. Our aim is to learn a PRM r_ϕ that assigns a reward to every step and provides a dense training signal for optimizing the policy model π_θ via RL. At inference time, an online-adapted r_ϕ steers the search process of π_θ , enhancing its reasoning ability on new questions.

3.2 Data Generation and Replay Buffer

At the start of training, we obtain the reference rollouts for each question $q \in \mathcal{Q}$ by either sampling multiple chains of thought (CoTs) from the current policy π_θ or reusing existing CoTs. We assign a binary outcome reward, indicates whether a CoT yields the correct final answer. CoTs with a positive reward are stored in a replay buffer $\mathcal{B}$ and serve as the reference rollouts for training the AIRL reward model. To keep the reliability and diversity of $\mathcal{B}$, we can periodically remove low-quality entries and add newly discovered high-quality CoTs during training.

3.3 Learning PRM via AIRL

To avoid costly step-wise labels utilized in training static PRMs, we leverage the generative adversarial network guided reward learning in AIRL to train a discriminator that distinguishes the reference rollouts from the policy outputs [13, 14, 18, 50]. In AIRL, the discriminator D_ϕ is defined over state-action pairs. For LLM reasoning, we represent

- *state*: the question q together with the preceding reasoning steps $\{C_1, \dots, C_{i-1}\}$, and
- *action*: the current step C_i .

Algorithm 1 AIRL-S

Input Initial LLM policy $\pi_{\theta_{\text{init}}}$; question set $\mathcal{Q}$; Total iterations E ; initial replay buffer $\mathcal{B}$

Initialize policy model π_{θ} , reference model π_{ref} , old policy π_{old} , and PRM r_{ϕ} with $\pi_{\theta_{\text{init}}}$

Update replay buffer $\mathcal{B}$ by collecting correct rollouts for $q \in \mathcal{Q}$ using π_{θ}

for iteration = 1, ..., E **do**

 Sample a batch of questions $\mathcal{B}_i$ from $\mathcal{B}$

 Generate a group of policy rollouts: $\{C^1, \dots, C^G\} \sim \pi_{\theta}(\cdot | q)$ for $q \in \mathcal{B}_i$

 Update the PRM according to AIRL loss in Equation (2)

 Update the policy π_{θ} according to the composite objectives in Equation (6)

 Update the old policy π_{old} using π_{θ}

 Update the replay buffer by adding the correct rollouts to $\mathcal{B}$

end for

Output Optimized policy model π_{θ} and PRM r_{ϕ}

The discriminator is:

$$D_{\phi}(C_i | q, C_{<i}) = \frac{\exp\{f_{\phi}(q, C_{\leq i})\}}{\exp\{f_{\phi}(q, C_{\leq i})\} + \pi_{\theta}(C_i | q, C_{<i})},$$

where f_{ϕ} is a learned scoring function.

The step-wise reward of C_i is then

$$r_{\phi}(C_i | q, C_{<i}) = \log \frac{D_{\phi}(C_i | q, C_{<i})}{1 - D_{\phi}(C_i | q, C_{<i})}. \quad (1)$$

And r_{ϕ} serves as the PRM.

We train the discriminator by minimizing

$$\mathcal{L}_{\text{AIRL}} = \sum_{i=1}^T \left[-\mathbb{E}_{q \sim \mathcal{Q}, C \sim \pi_e(\cdot | q)} [\log D_{\phi}(C_i | q, C_{<i})] - \mathbb{E}_{q \sim \mathcal{Q}, C \sim \pi_{\theta}(\cdot | q)} [\log(1 - D_{\phi}(C_i | q, C_{<i}))] \right], \quad (2)$$

where π_e denotes the reference rollout distribution drawn from the replay buffer $\mathcal{B}$ and π_{θ} is the current policy. Updating the discriminator can be seen as updating the reward function r_{ϕ} .

3.4 Policy Training with Combined RL Objectives

The policy model is optimized via RL. In AIRL [14], the objective of the policy model is to maximize the discriminator-derived rewards, thereby "fool" the discriminator into classifying its rollouts as reference rollouts. The AIRL objective is

$$\mathcal{J}_{\text{AIRL}}(\theta) = \mathbb{E}_{q \sim \mathcal{Q}, C \sim \pi_{\theta}(\cdot | q)} \left[\sum_{i=1}^{|C|} \min \left(\frac{\pi_{\theta}(C_i | q, C_{<i})}{\pi_{\text{old}}(C_i | q, C_{<i})} A_i, \text{clip} \left(\frac{\pi_{\theta}(C_i | q, C_{<i})}{\pi_{\text{old}}(C_i | q, C_{<i})}, 1 - \epsilon, 1 + \epsilon \right) A_i \right) \right], \quad (3)$$

where C is a chain of thought (CoT) produced by π_{θ} , $|C|$ is its length, π_{old} is the sampling policy, and A_i is the advantage at step i , estimated with a REINFORCE-style method [72].

DeepSeek-R1 achieves strong reasoning performance by pairing binary outcome rewards with group relative policy optimization (GRPO) [20]. For a group of G CoTs $\{C^k\}_{k=1}^G$, we utilize the same outcome rewards based GRPO, and the objective is

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{q \sim \mathcal{Q}, \{C^k\}_{k=1}^G \sim \pi_{\text{old}}(\cdot | q)} \left[\frac{1}{G} \sum_{k=1}^G \left(\min \left(\frac{\pi_{\theta}(C^k | q)}{\pi_{\text{old}}(C^k | q)} A^k, \text{clip} \left(\frac{\pi_{\theta}(C^k | q)}{\pi_{\text{old}}(C^k | q)}, 1 - \epsilon, 1 + \epsilon \right) A^k \right) - \beta \text{D}_{\text{KL}}(\pi_{\theta} \| \pi_{\text{ref}}) \right) \right], \quad (4)$$

$$\text{D}_{\text{KL}}(\pi_{\theta} \| \pi_{\text{ref}}) = \frac{\pi_{\text{ref}}(C^k | q)}{\pi_{\theta}(C^k | q)} - \log \frac{\pi_{\theta}(C^k | q)}{\pi_{\text{ref}}(C^k | q)} - 1, \quad (5)$$

where π_{ref} is a frozen reference model, and ϵ and β are hyper-parameters. The advantage A^k for each CoT in the group is computed from the binary outcome rewards.

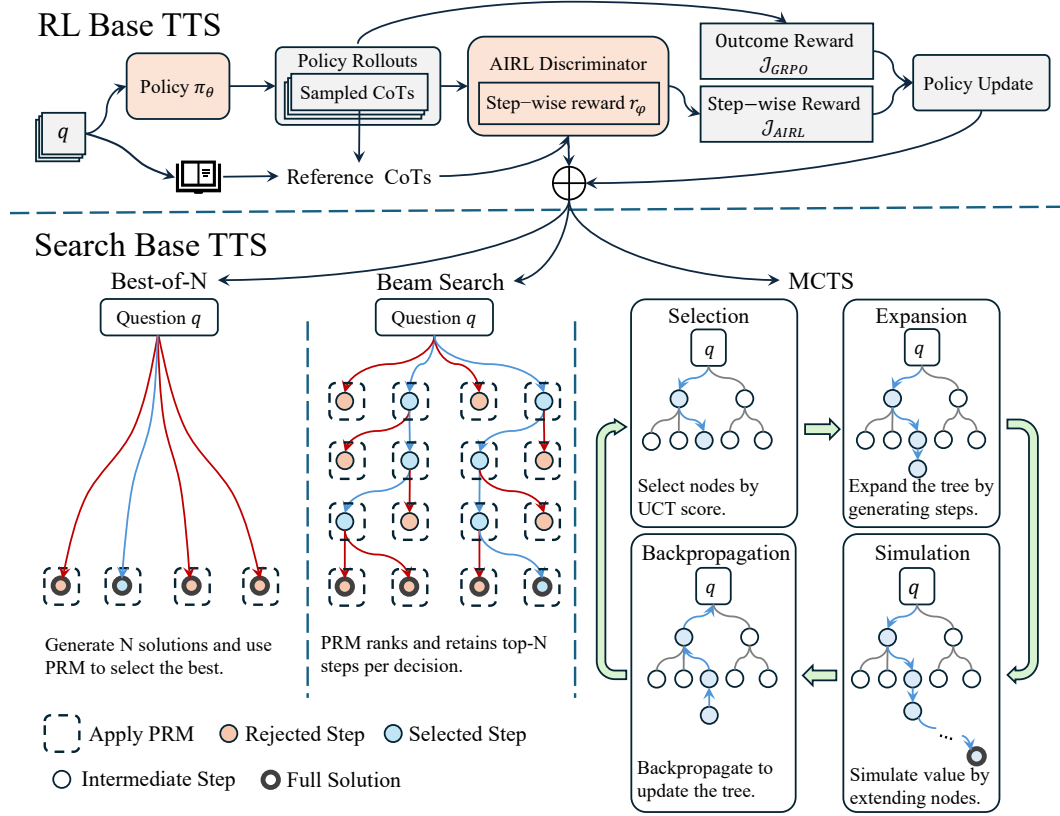


Figure 1: Overview of AIRL-S. During training, AIRL-S uses the AIRL discriminator to learn a PRM and optimizes the policy with both dense rewards from AIRL and outcome rewards from GRPO. At test time, the trained policy and PRM jointly guide downstream search algorithms.

We define a composite objective that combines the AIRL objective and GRPO objective to incorporate both the intermediate step rewards and the outcome rewards, and update the policy model to maximize the combined RL objectives:

$$\mathcal{J}(\theta) = \lambda \mathcal{J}_{\text{AIRL}}(\theta) + (1 - \lambda) \mathcal{J}_{\text{GRPO}}(\theta), \quad (6)$$

where λ is a hyperparameter to balance the outcome rewards and process rewards.

The training alternates between (i) updating the discriminator D_ϕ to distinguish reference rollouts from policy rollouts, and (ii) optimizing the policy by maximizing the composite objectives of AIRL and GRPO in Equation (6).

3.5 PRM-guided Test-Time Search

Test-Time Search. At inference time, search-based TTS explores multiple CoTs rather than generating a single pass. Approaches such as Best-of- N [62], beam search [75], and Monte Carlo Tree Search (MCTS) [85] improve reasoning by leverage outcome or process reward models to guide exploration. Because our reward function r_ϕ is learned and adapted on the policy LLM during RL, we can seamlessly reuse it as a PRM to steer the search procedure. When the policy proposes several actions, either intermediate steps or complete solutions, we score each candidate with r_ϕ and retain those with the highest scores according to the chosen search strategy. Figure 1 illustrates how r_ϕ integrates with Best-of- N , beam search, and MCTS; implementation details are provided in Appendix A.

- **Best-of- N .** For a question q , we sample N full solutions from $\pi_\theta(\cdot | q)$ and select the one with the highest PRM aggregation score (defined below).

- **Beam Search.** With beam size N and beam width M , we extend each beam node by M candidate steps, rank those steps by r_ϕ , and keep the top N nodes. The process continues until we obtain N complete solutions.
- **MCTS.** Using expansion width M , we select a node by the upper confidence bound for trees (UCT) criterion, which combines PRM rewards with visit counts [31, 63]. We then generate M child steps, simulate rollouts using r_ϕ to estimate node values, and back-propagate the rewards. The search returns N full solutions.

PRM Aggregation. After search, we obtain N candidate solutions $\{C^k\}_{k=1}^N$ for each question q . Following Snell et al. [62], for every solution C^k we compute a *step-wise aggregation* score

$$s(C^k) = \min_i r_\phi(C_i^k),$$

namely, the smallest PRM reward among its steps. Let A^k be the final answer produced by C^k . For *inter-answer aggregation*, we aggregate the step-wise scores of all solutions that yield the same distinct answer a :

$$S(a) = \sum_{k: A^k=a} s(C^k).$$

The answer with the largest $S(a)$ is selected as the model’s prediction; we term this method **PRM-Min-Sum**. Further implementation details and the definition of other PRM aggregation methods appear in Appendix A.

4 Experiments

To evaluate the effectiveness of the unified RL-based and search-based TTS technique in AIRL-S, we conduct extensive experiments to (i) compare the performance of our policy model to current state-of-the-art (SoTA) comparable size models on eight reasoning tasks, (ii) compare our PRM to PRMs trained on labeled datasets and implicit PRMs, (iii) combine our policy model and PRM with various search-based TTS techniques, (iv) conduct additional experiments to investigate the effect of the combined AIRL and GRPO objectives in RL training and the impact of various answer aggregation techniques in search-based TTS.

4.1 Experimental Details

LLMs. Policy Baselines: We select SoTA open-source and API models as baselines for our policy model. The open-source LLMs encompass models specifically designed for or excelling in reasoning through RL training, such as Math-Shepherd-Mistral-7B-RL [71], DeepSeek-Math-7B-RL [60], Qwen2.5-Math-7B-Instruct [24], and Euris-2-7B-PRIME [11]. We also include Qwen2.5-7B-Instruct [56] and Phi-4-14B [2] as general-purpose open-source baselines. We further include s1.1-7B [48] as a baseline trained through direct imitation learning on DeepSeek-R1 demonstrations instead of AIRL. For API models, we select DeepSeek-R1 [20], DeepseekV3 (2025-03-24) [38] and GPT-4o (2024-11-20) [53] as baselines. **PRM Baselines:** We select Math-Shepherd-Mistral-7B-PRM [69] and Llama3.1-8B-PRM-Deepseek-Data [77] as PRM baselines trained on labeled step-wise data. We choose the recent EurisPRM-Stage2 [79] as an implicit PRM baseline trained through ORM. More details of the baseline models are expanded in Appendix B.

Tasks. Mathematical/Scientific Reasoning: We evaluate on AIME2024 [45], AMC [46], MATH500 [23], and GPQA-Diamond [59] to validate the effectiveness of AIRL-S in mathematical or scientific reasoning tasks. **Coding:** We further evaluate the reasoning capabilities of our model in coding tasks on four benchmarks, including HumanEval [7], MBPP [19], LeetCode [10], and LiveCodeBench (v4) [26]. Details of all the tasks are introduced in Appendix B.

Training. Our full training dataset contains 160K questions, mostly sampled from NuminaMATH [33]. We use the LLM policy self-generated responses as the reference rollouts, and the rollouts are generated using rollout prompts, and more details are shared in Appendix B. We initialize both π_θ and f_ϕ using Qwen2.5-7B-Instruct, while we modify the head of f_ϕ into a scalar-value prediction head. The training is conducted on 8 NVIDIA A100 GPUs with Pytorch FSDP. We use a learning rate of $2e-6$ for the reward model and $5e-7$ for the policy model, maximum output length is set to

Table 1: Comparison of our policy model with ten SoTA open- and closed-source LLMs across eight reasoning benchmarks. Boldface indicates the top result within each model category; green highlighting denotes the best result among local models. Our policy achieves the highest average performance among local models and matches GPT-4o overall. (GPQA: GPQA-Diamond; LCB-v4: LiveCodeBench-v4)

Model	Mathematical/Scientific Reasoning				Coding				Avg.
	AIME2024	AMC	MATH500	GPQA	HumanEval	Leetcode	LCB-v4	MBPP	
API Models									
DeepSeek-R1	79.8	85.5	96.5	71.2	97.6	90.4	71.1	95.8	86.0
DeepSeek-V3	36.7	81.9	90.2	61.1	93.3	88.3	67.0	88.9	75.9
GPT-4o	13.3	50.6	65.8	43.9	90.2	60.6	44.2	87.3	57.0
Local Models (RL with PRMs)									
Math-Shepherd-Mistral-7B-RL	0.0	7.2	28.6	9.6	32.3	5.6	3.9	51.1	17.3
DeepSeekMath-7B-RL	3.3	18.1	48.6	23.2	58.5	11.2	6.2	73.1	30.3
Qwen2.5-Math-7B-Instruct	13.3	50.6	79.6	29.3	57.9	11.7	9.3	46.0	37.2
Eurus-2-7B-PRIME	20.0	56.6	79.2	33.8	70.7	31.1	24.3	70.1	48.2
Local Models (Other Baselines)									
s1.1-7B	16.7	38.6	72.6	35.4	76.8	11.1	9.3	76.7	42.2
Qwen2.5-7B-Instruct	16.7	33.7	72.0	32.5	81.7	47.4	28.0	79.4	48.9
Phi-4-14B	13.3	44.6	78.6	55.6	84.1	45.6	31.2	74.3	53.4
Qwen2.5-7B-AIRL-S (Ours)	26.7	59.0	80.2	40.2	85.1	54.4	31.3	83.3	57.5

8,192, and the training batch size is 1024. For each question, we sample 8 outputs using the rollout prompts. The KL coefficient is set to 0.001, λ is set to 0.5, and the total training epochs are 2.

Evaluation. For results where search-based TTS is not applied, we report zero-shot Accuracy@1 for math and science tasks, and Pass@1 for coding tasks using a temperature of 0 for reproducibility. For results where search-based TTS is applied, we use a temperature of 0.7 to enable step exploration, tree width M is set to 4, and the performance of the PRM aggregated solution is reported. The main results are averaged over three runs.

4.2 Main Results

Effectiveness of the Policy Model. To comprehensively validate the performance of our policy model Qwen2.5-7B-AIRL-S, we compare it to ten SoTA LLMs (open- and closed-source) on eight reasoning tasks. The results in Table 1 lead to the following observations: (i) AIRL-S *can effectively enhance the reasoning performance of the base model*. When trained on Qwen2.5-7B-Instruct, our policy achieves an average improvement of **9%** across eight benchmarks and a **13%** improvement on mathematical and scientific reasoning tasks. (ii) *Our policy exhibits strong performance compared to existing LLMs trained through RL*. AIRL-S outperforms Math-Shepherd-Mistral-7B-RL, DeepSeekMath-7B-RL, and Qwen2.5-Math-7B-Instruct (trained with PRMs from labeled step-wise datasets) and Eurus-2-7B-PRIME (trained with an implicit PRM derived from its policy). Compared to these models, AIRL-S consistently achieves higher reasoning accuracy on all tasks by learning a policy-independent step-wise reward function through AIRL. (iii) *Our approach surpasses direct imitation learning on reference rollouts*. Compared to s1.1-7B trained via supervised fine-tuning on DeepSeek-R1 demonstrations, AIRL-S yields an average improvement of **15%** by learning a reward function from reference rollouts and maximizing the rewards instead of direct imitation learning. (iv) Finally, AIRL-S outperforms the recently released Phi-4-14B on average despite having half its parameter count and matches the performance of the general-purpose model GPT-4o.

Effectiveness of the PRM. We assess the generalizability of our PRM, Qwen2.5-AIRL-S-PRM, by comparing it to PRMs trained on labeled step-wise datasets. We evaluate four PRMs using Best-of-N search with 64 rollouts on AIME2024, AMC, and MATH500 across four generative LLMs. The average performance of each LLM-PRM pair is shown in Figure 2, and detailed results are provided in Appendix C. Our observations are: (i) *Qwen2.5-AIRL-S-PRM improves Best-of-N performance for all LLMs and datasets*, outperforming Math-Shepherd-Mistral-7B-PRM and EurusPRM-Stage2 by 2.4% and 1.4%, respectively. (ii) *Combining Qwen2.5-7B-AIRL-S with Qwen2.5-AIRL-S-PRM yields the best performance among all LLM-PRM combinations*, with an **11%** gain over Qwen2.5-

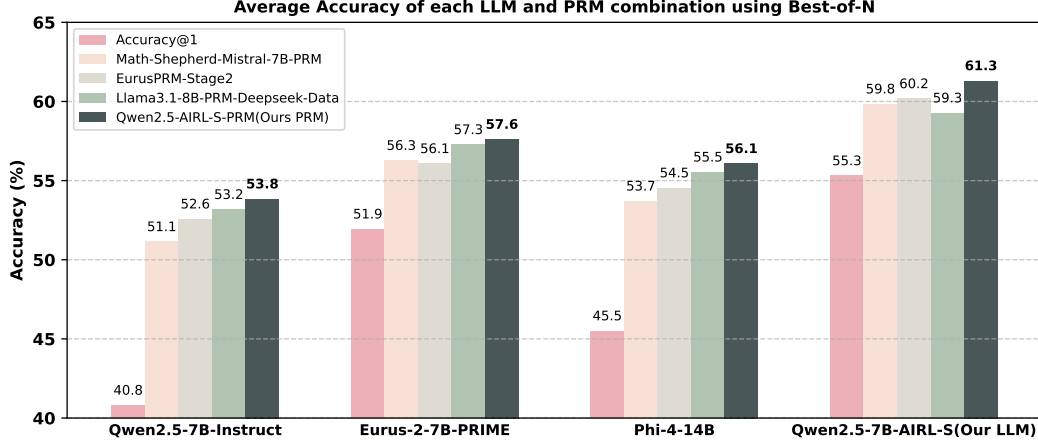


Figure 2: Average performance of four PRMs applied to four generative LLMs using Best-of-N with 64 rollouts on AIME2024, AMC, and MATH500. Our PRM (Qwen2.5-AIRL-S-PRM) consistently delivers the highest test-time search performance across all models and datasets.

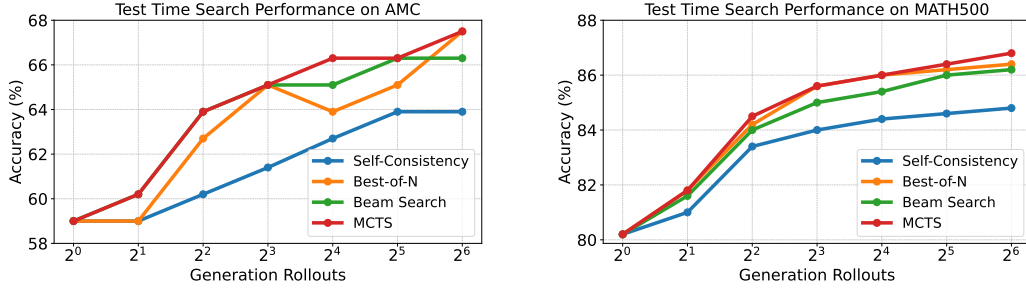


Figure 3: Comparison of test-time search performance with our PRM applied to MCTS, Beam Search, and Best-of-N across varying rollout counts. Our PRM consistently improves performance for all search techniques.

7B-Instruct with Math-Shepherd-Mistral-7B-PRM. These results demonstrate that AIRL-S effectively unifies RL-based TTS and Search-based TTS and that our PRM generalizes across different models and datasets.

Effectiveness on Different Search-Based TTS Methods. We evaluate our PRM within three search-based TTS methods, MCTS, beam search, and Best-of-N, using Qwen2.5-7B-AIRL-S as the generative LLM and Qwen2.5-AIRL-S-PRM as the verifier. We include Self-Consistency [70] as a baseline for sanity check: it generates multiple complete solutions and uses majority vote to select the final answer without any PRM or ORM. Results on AMC and MATH500 in Figure 3 show that: (i) *Our PRM effectively guides the search, selection, and aggregation processes for all search algorithms.* It improves performance across all search-based methods as the number of rollouts increases and outperforms Self-Consistency under the same rollouts. This result indicates that reward-guided step generation and selection produce more accurate intermediate decisions than majority voting on final answers. (ii) The PRM is especially effective for value-based methods like MCTS. By estimating step values during tree expansion and selecting nodes by UCT score, MCTS benefits more from the PRM’s guidance, leading to more accurate intermediate steps.

4.3 Ablation Study

Impact of Our PRM on RL Training. To assess the contribution of our reward function on RL training, we compare AIRL-S to GRPO trained with outcome rewards only (no PRM), while keeping all other settings identical to AIRL-S. The results in Figure 4 show that adding our PRM improves

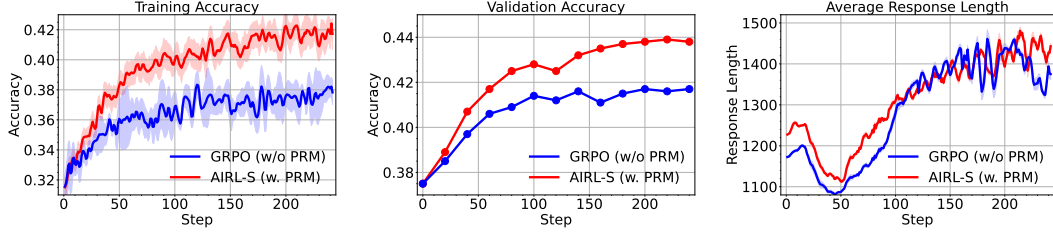


Figure 4: Comparison of AIREL-S and GRPO trained with outcome rewards only. AIREL-S improves training and validation performance and enables longer response generation at test time.

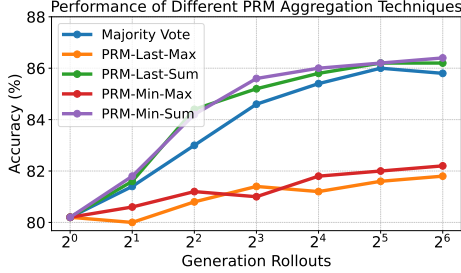


Figure 5: Performance of different PRM aggregation techniques evaluated on MATH500.

Table 2: Comparison of our policy model and PRM with PRIME [11] train on the same setting. AIREL-S consistently outperforms PRIME both with and without test-time search. AIME, MATH, and BoN denote AIME2024, MATH500, and Best-of-N, respectively.

Method	AIME	AMC	MATH	Avg.
PRIME	16.7	55.4	78.4	50.2
PRIME w. BoN	20.0	62.7	84.2	55.6
AIREL-S	26.7	59.0	80.2	55.3
AIREL-S w. BoN	30.0	67.5	86.4	61.3

both training and validation accuracy. Furthermore, as training progresses, AIREL-S enables longer response generation at test time, demonstrating its ability to scale inference computation.

Impact of PRM Aggregation Techniques. As described in Section 3.5 and Appendix A, different PRM aggregation techniques can select the final answer from multiple rollouts. We evaluate these techniques using our policy and PRM with Best-of-N search on MATH500; results of five aggregation techniques (defined in Appendix A) appear in Figure 5. PRM-Min-Vote and PRM-Last-Vote achieve the highest accuracy, showing that step-level rewards enable effective answer aggregation. In contrast, PRM-Min-Max and PRM-Last-Max achieve the worst performance, demonstrating the importance of inter-answer aggregation via PRM weighted voting to integrated the answers from various rollouts.

Comparison with Concurrent Implicit PRM Methods. We compare AIREL-S with PRIME [11], which trains a policy-dependent dense reward function based on [79]. Both methods are trained on Qwen2.5-7B-Instruct under identical settings. At inference, we apply Best-of-N search under 64 rollouts with each method’s PRM. Results on AIME2024, AMC, and MATH500 in Figure 2 show that our policy and PRM combination outperforms PRIME, further confirming that AIREL-S unifies RL-based and search-based TTS.

5 Conclusion

In this paper, we present AIREL-S, a framework that unifies RL-based and search-based TTS for LLMs. AIREL-S integrates AIREL with GRPO to learn a step-wise PRM from reference rollouts without any labeled data. This PRM guides both RL training by improving training stability and sample efficiency, and inference search by extending reasoning chains and mitigating reward hacking. We evaluate AIREL-S on eight benchmarks covering mathematics, scientific reasoning, and code generation. The results show that AIREL-S significantly improves the performance of base models and outperforms existing RL-trained and imitation-based methods. The PRM generalizes across different LLMs, datasets, and search algorithms, consistently enhancing test-time search performance. These findings validate that unifying RL-based and search-based TTS offers an efficient and robust approach for complex reasoning.

References

- [1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004.
- [2] Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, et al. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*, 2024.
- [3] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.
- [4] Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.
- [5] Diego Carraro and Derek Bridge. Enhancing recommendation diversity by re-ranking with large language models. *ACM Transactions on Recommender Systems*, 2024.
- [6] Alex J Chan, Hao Sun, Samuel Holt, and Mihaela Van Der Schaar. Dense reward for free in reinforcement learning from human feedback. In *Proceedings of the 41st International Conference on Machine Learning*, pages 6136–6154, 2024.
- [7] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pinto, Jared Kaplan, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL <https://arxiv.org/abs/2107.03374>.
- [8] Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, pages 4299–4307, 2017. URL <https://papers.nips.cc/paper/7017-deep-reinforcement-learning-from-human-preferences.pdf>.
- [9] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- [10] Tristan Coignion, Clément Quinton, and Romain Rouvoy. A performance study of llm-generated code on leetcode. *arXiv preprint arXiv:2407.21579*, 2024. URL <https://arxiv.org/abs/2407.21579>.
- [11] Ganqu Cui, Lifan Yuan, Zefan Wang, et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025. URL <https://arxiv.org/abs/2502.01456>.
- [12] DeepSeek-AI. Deepseek-math-7b-rl: Reinforcement learning enhanced math reasoning model. GitHub repository, 2024. URL <https://github.com/deepseek-ai/DeepSeek-Math>.
- [13] Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *arXiv preprint arXiv:1611.03852*, 2016.
- [14] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. In *International Conference on Learning Representations*, 2018.
- [15] Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Daoguang Zan, Shangaoran Quan, Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. Omni-math: A universal olympiad level mathematic benchmark for large language models, 2024. URL <https://arxiv.org/abs/2410.07985>.
- [16] Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pages 10835–10866. PMLR, 2023.

- [17] Divyansh Garg, Shuvam Chakraborty, Chris Cundy, Jiaming Song, and Stefano Ermon. Iq-learn: Inverse soft-\$q\$ learning for imitation. In *NeurIPS 2021 Spotlight*, 2021. URL <https://proceedings.neurips.cc/paper/2021/hash/210f760a89db30aa72ca258a3483cc7f-Abstract.html>.
- [18] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.
- [19] Google Research, Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Mbpp-sanitized. Hugging Face Datasets, 2021. URL <https://huggingface.co/datasets/Muennighoff/mbpp>. Sanitized split of the MBPP benchmark.
- [20] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-rl: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- [21] Joey Hejna and Dorsa Sadigh. Inverse preference learning: Preference-based rl without a reward function. volume 36, pages 18806–18827, 2023.
- [22] Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. Measuring coding challenge competence with APPS. *CoRR*, abs/2105.09938, 2021. URL <https://arxiv.org/abs/2105.09938>.
- [23] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. URL <https://arxiv.org/abs/2103.03874>.
- [24] Binyuan Hui, Jian Yang, Zeyu Cui, Jiayi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.
- [25] Robert Irvine, Douglas Boubert, Vyas Raina, Adian Liusie, Ziyi Zhu, Vineet Mudupalli, Aliaksei Korshuk, Zongyi Liu, Fritz Cremer, Valentin Assassi, et al. Rewarding chatbots for real-world engagement with millions of users. *arXiv preprint arXiv:2303.06135*, 2023.
- [26] Ishaan Jain, Fan Zhang, Wei Liu, et al. Livecodebench: Holistic and contamination-free evaluation of llms for code. In *International Conference on Learning Representations (ICLR) 2025*, 2025. URL <https://livecodebench.github.io/>.
- [27] Can Jin, Tong Che, Hongwu Peng, Yiyuan Li, Dimitris Metaxas, and Marco Pavone. Learning from teaching regularization: Generalizable correlations should be easy to imitate. *Advances in Neural Information Processing Systems*, 37:966–994, 2024.
- [28] Can Jin, Hongwu Peng, Qixin Zhang, Yujin Tang, Dimitris N Metaxas, and Tong Che. Two heads are better than one: Test-time scaling of multi-agent collaborative reasoning. *arXiv preprint arXiv:2504.09772*, 2025.
- [29] Mingyu Jin, Weidi Luo, Sitao Cheng, Xinyi Wang, Wenyue Hua, Ruixiang Tang, William Yang Wang, and Yongfeng Zhang. Disentangling memory and reasoning ability in large language models. *arXiv preprint arXiv:2411.13504*, 2024.
- [30] Mingyu Jin, Qinkai Yu, Dong Shu, Haiyan Zhao, Wenyue Hua, Yanda Meng, Yongfeng Zhang, and Mengnan Du. The impact of reasoning step length on large language models. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 1830–1842, 2024.
- [31] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

- [32] Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. Numinamath. <https://huggingface.co/datasets/AI-MO/NuminaMath-CoT>, 2024.
- [33] Jia LI, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. Numinamath. [<https://huggingface.co/AI-MO/NuminaMath-CoT>] (https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf), 2024.
- [34] Rongao Li, Jie Fu, Bo-Wen Zhang, Tao Huang, Zhihong Sun, Chen Lyu, Guang Liu, Zhi Jin, and Ge Li. TACO: Topics in algorithmic code generation dataset. *arXiv preprint arXiv:2312.14852*, 2023.
- [35] Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. Making language models better reasoners with step-aware verifier. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5315–5333, 2023.
- [36] Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d’Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel J. Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022. doi: 10.1126/science.abq1158. URL <https://www.science.org/doi/abs/10.1126/science.abq1158>.
- [37] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let’s verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.
- [38] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.
- [39] Fei Liu et al. Learning to summarize from human feedback. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 583–592, 2020.
- [40] Shicheng Liu and Minghui Zhu. Distributed inverse constrained reinforcement learning for multi-agent systems. *Advances in Neural Information Processing Systems*, 35:33444–33456, 2022.
- [41] Shicheng Liu and Minghui Zhu. In-trajectory inverse reinforcement learning: Learn incrementally before an ongoing trajectory terminates. *Advances in Neural Information Processing Systems*, 37:117164–117209, 2024.
- [42] Yue Liu, Hongcheng Gao, Shengfang Zhai, Xia Jun, Tianyi Wu, Zhiwei Xue, Yulin Chen, Kenji Kawaguchi, Jiaheng Zhang, and Bryan Hooi. Guardreasoner: Towards reasoning-based llm safeguards. *arXiv preprint arXiv:2501.18492*, 2025.
- [43] Yue Liu, Shengfang Zhai, Mingzhe Du, Yulin Chen, Tri Cao, Hongcheng Gao, Cheng Wang, Xinfeng Li, Kun Wang, Junfeng Fang, Jiaheng Zhang, and Bryan Hooi. Guardreasoner-vl: Safeguarding vlms via reinforced reasoning. *arXiv preprint arXiv:2505.11049*, 2025.
- [44] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594, 2023.
- [45] Mathematical Association of America. 2024 american invitational mathematics examination (aime i). Technical report, Mathematical Association of America, 2024. URL https://artofproblemsolving.com/wiki/index.php/2024_AIME_I.

- [46] Mathematical Association of America. American Mathematics Competitions (AMC). <https://www.maa.org/student-programs/amc>, 2025. Accessed: 2025-05-05.
- [47] Yuchun Miao, Sen Zhang, Liang Ding, Rong Bao, Lefei Zhang, and Dacheng Tao. Inform: Mitigating reward hacking in rlhf via information-theoretic reward modeling. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.
- [48] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.
- [49] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling, 2025. URL <https://arxiv.org/abs/2501.19393>.
- [50] Weili Nie, Nina Narodytska, and Ankit Patel. Relgan: Relational generative adversarial networks for text generation. In *International conference on learning representations*, 2018.
- [51] OpenAI. Openai o1 system card. Technical report, OpenAI, December 2024. URL <https://arxiv.org/abs/2412.16720>.
- [52] OpenAI. Learning to reason with llms, September 2024. URL <https://openai.com/index/learning-to-reason-with-llms/>.
- [53] OpenAI. Gpt-4o system card. Technical report, OpenAI, November 2024. URL <https://cdn.openai.com/gpt-4o-system-card.pdf>.
- [54] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelvin, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022. URL <https://arxiv.org/abs/2203.02155>.
- [55] Sungjin Park, Xiao Liu, Yeyun Gong, and Edward Choi. Ensembling large language models with process reward-guided tree search for better complex reasoning, 2024.
- [56] Qwen Team, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. URL <https://arxiv.org/abs/2412.15115>.
- [57] Ahmad Rashid, Ruotian Wu, Julia Grosse, Agustinus Kristiadi, and Pascal Poupart. A critical look at tokenwise reward-guided text generation. *arXiv preprint arXiv:2406.07780*, 2024.
- [58] Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. Maximum margin planning. In *Proceedings of the 23rd international conference on Machine learning*, pages 729–736, 2006.
- [59] David Rein, Betty Li, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023. URL <https://arxiv.org/abs/2311.12022>.
- [60] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.
- [61] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Free process rewards without process labels. *arXiv preprint arXiv:2412.01981*, 2024. URL <https://arxiv.org/abs/2412.01981>.
- [62] Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling test-time compute optimally can be more effective than scaling LLM parameters. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=4FWAwZtd2n>.

- [63] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias W Seeger. Information-theoretic regret bounds for gaussian process optimization in the bandit setting. *IEEE transactions on information theory*, 58(5):3250–3265, 2012.
- [64] Matrix Studio. Codeforces–python submissions. <https://huggingface.co/datasets/MatrixStudio/Codeforces-Python-Submissions>, 2024.
- [65] Gokul Swamy, David Wu, Sanjiban Choudhury, Drew Bagnell, and Steven Wu. Inverse reinforcement learning without reinforcement learning. pages 33299–33318, 2023.
- [66] Jie Tan, Kangfei Zhao, Rui Li, Jeffrey Xu Yu, Chengzhi Piao, Hong Cheng, Helen Meng, Deli Zhao, and Yu Rong. Can large language models be query optimizer for relational databases? *arXiv preprint arXiv:2502.05562*, 2025.
- [67] Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.
- [68] Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.
- [69] Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439, 2024.
- [70] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. 2022.
- [71] Zihuai Wang, Xueyan Liu, Hao Chen, Stefano Ermon, et al. Math-shepherd: Verify and reinforce llms step-by-step. *arXiv preprint arXiv:2312.08935*, 2023. URL <https://arxiv.org/abs/2312.08935>.
- [72] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.
- [73] Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models. *arXiv preprint arXiv:2408.00724*, 2024.
- [74] Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for LLM problem-solving. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=VNckp7JEHn>.
- [75] Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, James Xu Zhao, Min-Yen Kan, Junxian He, and Michael Xie. Self-evaluation guided beam search for reasoning. *Advances in Neural Information Processing Systems*, 36:41618–41650, 2023.
- [76] Yuxi Xie, Anirudh Goyal, Wenye Zheng, Min-Yen Kan, Timothy P Lillicrap, Kenji Kawaguchi, and Michael Shieh. Monte carlo tree search boosts reasoning via iterative preference learning. *arXiv preprint arXiv:2405.00451*, 2024.
- [77] Wei Xiong, Hanning Zhang, Nan Jiang, and Tong Zhang. An implementation of generative prm. <https://github.com/RLHFlow/RLHF-Reward-Modeling>, 2024.
- [78] Fei Yu, Anningzhe Gao, and Benyou Wang. Outcome-supervised verifiers for planning in mathematical reasoning. *arXiv preprint arXiv:2311.09724*, 2(6), 2023.
- [79] Lifan Yuan, Wendi Li, Huayu Chen, Ganqu Cui, Ning Ding, Kaiyan Zhang, Bowen Zhou, Zhiyuan Liu, and Hao Peng. Free process rewards without process labels. *arXiv preprint arXiv:2412.01981*, 2024.

- [80] Siyu Yuan, Zehui Chen, Zhiheng Xi, Junjie Ye, Zhengyin Du, and Jiecao Chen. Agent-r: Training language model agents to reflect via iterative self-training. *arXiv preprint arXiv:2501.11425*, 2025.
- [81] Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhua Chen. Mammoth: Building math generalist models through hybrid instruction tuning. 2023.
- [82] Di Zhang, Jianbo Wu, Jingdi Lei, Tong Che, Jiatong Li, Tong Xie, Xiaoshui Huang, Shufei Zhang, Marco Pavone, Yuqiang Li, et al. Llama-berry: Pairwise optimization for o1-like olympiad-level mathematical reasoning. *arXiv preprint arXiv:2410.02884*, 2024.
- [83] Shun Zhang, Zhenfang Chen, Yikang Shen, Mingyu Ding, Joshua B Tenenbaum, and Chuang Gan. Planning with large language models for code generation. *arXiv preprint arXiv:2303.05510*, 2023.
- [84] Xuan Zhang, Chao Du, Tianyu Pang, Qian Liu, Wei Gao, and Min Lin. Chain of preference optimization: Improving chain-of-thought reasoning in llms. *Advances in Neural Information Processing Systems*, 37:333–356, 2024.
- [85] Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning, acting, and planning in language models. In *Proceedings of the 41st International Conference on Machine Learning*, pages 62138–62160, 2024.
- [86] Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI '08)*, pages 1433–1438, 2008. URL <https://cdn.aaai.org/AAAI/2008/AAAI08-227.pdf>.
- [87] Brian D Ziebart, J Andrew Bagnell, and Anind K Dey. Modeling interaction via the principle of maximum causal entropy. Carnegie Mellon University, 2010.

A Details for Search-based TTS

A.1 PRM Aggregation Techniques

Answer Aggregation. After search, we obtain N candidate reasoning chains for each question q . Following Snell et al. [62], we compute a *step-wise aggregation score* for each chain by either taking the minimum PRM reward across its steps or using the reward of its final step. For *inter-answer aggregation*, we either select the answer whose chain has the highest step-wise score (Maximum Score) or sum the step-wise scores of all chains that produce the same answer and choose the answer with the highest total (Sum Score).

Step-wise Aggregation. For each reasoning chain, we compute its score by one of two methods:

- **Minimum Step Reward:** Use the minimum PRM reward among all steps in the chain.
- **Last Step Reward:** Use the PRM reward at the chain’s final step.

Inter-answer Aggregation. After computing step-wise scores, we aggregate across chains that yield the same final answer using one of two methods:

- **Maximum Score:** Select the answer of the chain with the highest step-wise score.
- **Sum Score:** Sum the step-wise scores of all chains with the same answer and select the answer with the highest total.

These choices yield four PRM aggregation methods:

- **PRM-Min-Max:** Minimum Step Reward + Maximum Score.
- **PRM-Min-Sum:** Minimum Step Reward + Sum Score.
- **PRM-Last-Max:** Last Step Reward + Maximum Score.
- **PRM-Last-Sum:** Last Step Reward + Sum Score.

In addition, we include **Majority Vote**, which selects the final answer that appears most frequently among all reasoning chains.

A.2 Test-Time Search Techniques

Best-of-N Sampling. We sample N complete chains of thought (CoTs) from the policy $\pi_\theta(\cdot \mid q)$. By default, we aggregate candidates using the PRM-Min-Sum method described in Section 3.5.

Beam Search. We adopt the BFS-V implementation with beam size N and beam width M :

1. Generate N initial proposals from π_θ .
2. Score each proposal using the PRM’s step-wise reward.
3. Retain the top $\frac{N}{M}$ proposals.
4. For each surviving proposal, sample M next-step candidates and return to Step 2.

Repeat until N complete solutions are generated. By default, we aggregate candidates with PRM-Min-Sum (Section 3.5).

Monte Carlo Tree Search (MCTS). MCTS combines exploration via UCT with PRM-based value estimation:

1. **Selection.** From the root, traverse by choosing the child n that maximizes

$$\text{UCT}(n) = \mu_n + \alpha \sqrt{\frac{\ln N_{\text{parent}}}{N_n}},$$

where μ_n is the cumulative PRM reward of all future steps, N_{parent} and N_n are the visit counts of the parent and current node, respectively, and $\alpha = 1.25$ by default.

2. **Expansion.** At a leaf node, generate M new steps from π_θ .
3. **Simulation.** Roll out each new node to completion (temperature 0) and use the terminal PRM reward as the node’s value.
4. **Backpropagation.** Update μ_n , N_{parent} , and N_n along the selected path.

Terminate when N complete solutions are obtained. By default, we use PRM-Min-Sum to aggregate candidates (Section 3.5).

Self-Consistency. We sample N CoTs from π_θ without PRM guidance and aggregate answers by majority vote (see Appendix A). Self-Consistency exploits output diversity but does not use any PRM feedback.

B Implementation Details

B.1 Prompts

By default, we use the same CoT prompts for both training and evaluation. The CoT prompt in our experiments are indicated in Table B.1.

Task	Prompt
MATHEMATICAL/SCIENTIFIC REASONING	Please reason step by step, and put your final answer within <code>\boxed{Your answer}</code> .
CODING	Write Python code to solve the problem. Please reason step by step and present the code in <code>'''python YOUR CODE'''</code> at the end.

B.2 Models

To clarify the context of our empirical study, Table 3 summarises the LLMs we utilized in our experiments, detailing their backbone architectures and key training procedures.

B.3 Training Datasets

Our training datasets consist of math questions sampled from NuminaMath [32], openaimath [49], Omni-MATH [15], and code questions sampled APPS [22], CodeContests [36], TACO [34], and Codeforces [64]. The detailed information of our training dataset is indicated in Table 4.

B.4 Validation Datasets

Our validation dataset is sampled from NuminaMath[32], APPS[22], CodeContests[36], and Codeforces[64] with a total of 2K mathematical and coding questions.

B.5 Evaluation Tasks

For completeness, Table 5 enumerates the evaluation tasks—covering mathematical/scientific reasoning, and code generation—on which we evaluate all systems throughout this paper.

C Additional Results

The detailed performance of each LLM-PRM combination using Best-of-N with 64 rollouts on AIME2024, AMC, and MATH500 are shown in Figure 6, Figure 7 and Figure 8, respectively. Our

Table 3: Overview of LLMs used in our experiments.

Model	Backbone	Introduction
Eurus-2-7B-PRIME	Qwen2.5-Math-7B	Training by two stages: (i) supervised fine-tuning on the Eurus instruction corpus; (ii) online RL with PRIME implicit process reward using GSM8K, MATH, and synthetic Chain-of-Thought data [11].
DeepSeek-Math-7B-RL	DeepSeek-R1-7B	SFT on <i>DeepSeekMath-Instruct</i> (800 K math CoT pairs) followed by GRPO reinforcement learning [60].
Math-Shepherd-Mistral-7B-RL	Mistral-7B	PRM-guided step-wise PPO training on Meta-MATH and MATH Level-5 problems [71].
Qwen-2.5-Math-7B-Instruct	Qwen-2.5	Supervised fine-tuning on 200 K math instructions plus RLHF preference optimization [24].
DeepSeek-R1	-	Four-stage pipeline: pre-training on 1.2T tokens, SFT, RLHF, and DPO. Public endpoint <code>deepseek-ai/DeepSeek-R1</code> . We use this model for inference based on the together API [20].
DeepSeek-V3	-	Continued bilingual web-scale pre-training on DeepSeek-R1, followed by alignment stages; version tag 0324. We use this model for inference based on the together API [38].
s1.1-7B	Qwen2.5-7B-Instruct	Few-shot SFT and low-resource RL on ≈ 1 K GPT-4-generated reasoning tasks (s1K-1.1) [48].
Qwen-2.5-7B-Instruct	-	2M general instruction pairs plus RLHF; supports 131K-token context [56].
DeepSeek-R1-Distill-Qwen-7B	Qwen2.5-Math-7B	Distillation of DeepSeek-R1 pseudo-labels into Qwen-7B for reduced inference cost.
GPT-4o	-	OpenAI RLHF pipeline; accessed via <code>chat/completions</code> with model <code>gpt-4o-2024-11-20</code> . We use this model for inference based on the OpenAI API [53].

policy model combined with our PRM consistently achieves the best performance in all datasets and LLM-PRM combinations, demonstrating the effectiveness of AIRL-S in unifying RL-based and Search-based TTS.

D Limitations

While our experiments demonstrate that AIRL-S effectively unifies RL-based and search-based TTS across multiple test-time search algorithms using a single policy model and PRM, we have not tested its scalability on other base model architectures such as Phi-4 [2] and Gemma-3 [67]. We also have not studied how training dataset size affects AIRL-S’s performance. Future work will evaluate AIRL-S on larger language models and more extensive datasets, including tasks with more difficult reasoning questions.

E Broader Impacts

Large-scale reward-model-guided reinforcement learning (RL) and test-time search (TTS) can *amplify* the reasoning power of language models, lowering the barrier to advanced problem solving. **Positive impacts** include: (i) *Scientific acceleration*—open-sourced PRMs provide a drop-in verifier for symbolic mathematics and formal-methods research, potentially shortening proof discovery cycles;

Table 4: Training dataset source in our experiments.

Dataset	# Questions	Introduction
<i>Mathematical Reasoning</i>		
NuminaMath	125K	Olympiad-level competition problems with chain-of-thought solutions; the metric is exact-match accuracy [32].
Openaimath	12K	Competition-style questions released by OpenAI, spanning algebra, combinatorics, geometry, etc.; metric is exact-match accuracy [49].
Omni-MATH	4K	Universal Olympiad-level problems covering 8 subjects and 6 difficulty tiers; evaluated via exact-match correctness [15].
<i>Coding</i>		
APPS	3K	Real-world programming challenges of varying difficulty; graded by hidden unit tests— metric is Pass@1 [22].
CodeContests	3K	Codeforces-style contest tasks curated for AlphaCode; metric is Pass@1 [36].
TACO	10K	Algorithmic coding problems grouped by topic and difficulty; metric is Pass@1 [34].
Codeforces	3K	A cleaned subset of Python submissions from Codeforces transformed into standalone tasks; metric is Pass@1 [64].

Table 5: Introduction of evaluation tasks used in our experiments.

Dataset	Introduction
<i>Mathematical/Scientific Reasoning</i>	
AIME2024	American Invitational Mathematics Examination 2024. Two papers (AIME I & II) with 15 questions each, for a total of 30 short-answer problems. We evaluate <i>exact-match answer accuracy</i> and average across three random seeds for stability [45].
AMC	All come from AMC12 2022, AMC12 2023 with 83 multiple-choice questions. Models must output the correct option; metric is accuracy [46].
MATH500	A subset of the MATH benchmark containing 500 competition problems spanning seven topics and five difficulty levels. We report accuracy[23] .
GPQA-Diamond	High-quality slice of GPQA with 198 graduate-level multiple-choice questions in biology, chemistry and physics [59]. We report accuracy.
<i>Coding Tasks</i>	
HumanEval	Classic Python code-generation set with 164 functions, each graded by unit tests. We report Pass@1 [7].
MBPP	Sanitized version of Mostly Basic Python Problems comprising 427 hand-verified tasks; Pass@1 is reported [19].
LeetCode	180 LeetCode tasks including 45 Easy, 91 Medium and 44 Hard; we convert to Python and grade with official test cases, reporting Pass@1 [10].
LiveCodeBench	Continually-updated contest benchmark. We use releases v4 with 776 tasks; metric is Pass@1 [26].

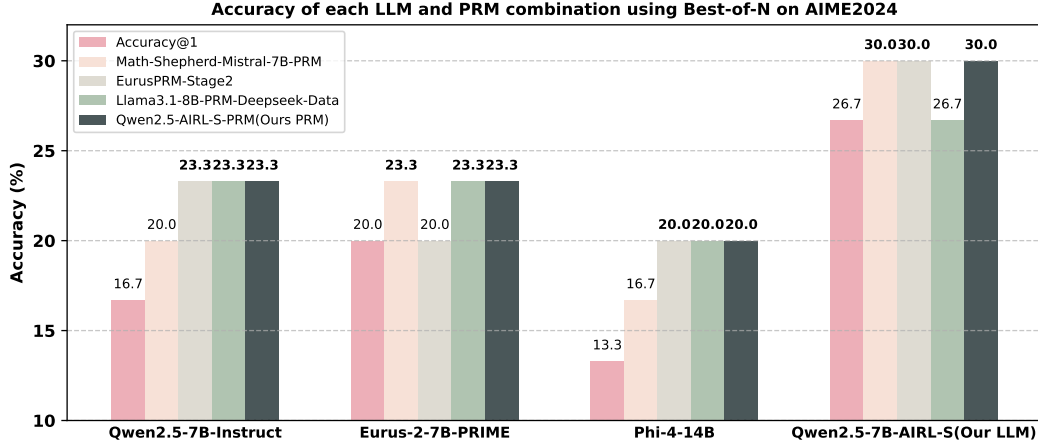


Figure 6: Performance of applying four PRMs on four different generation LLMs using Best-of-N with 64 rollouts on AIME2024

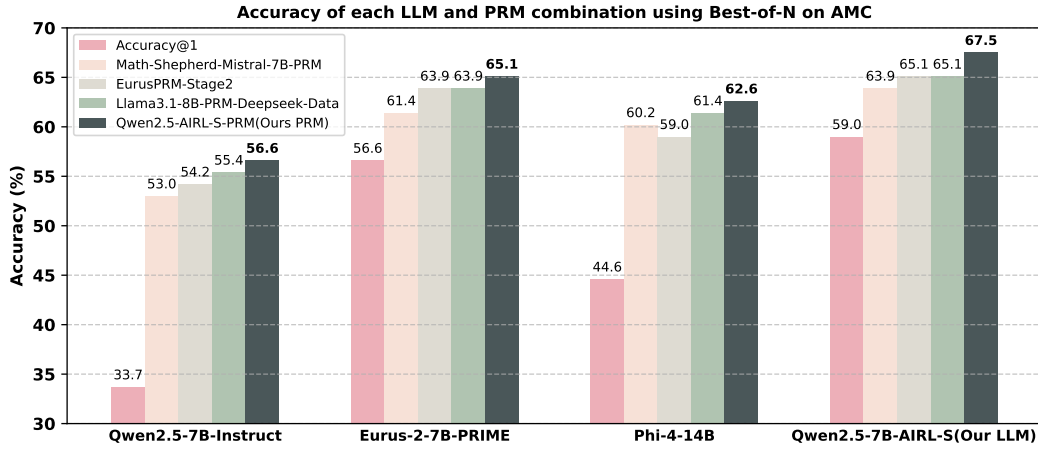


Figure 7: Performance of applying four PRMs on four different generation LLMs using Best-of-N with 64 rollouts on AMC

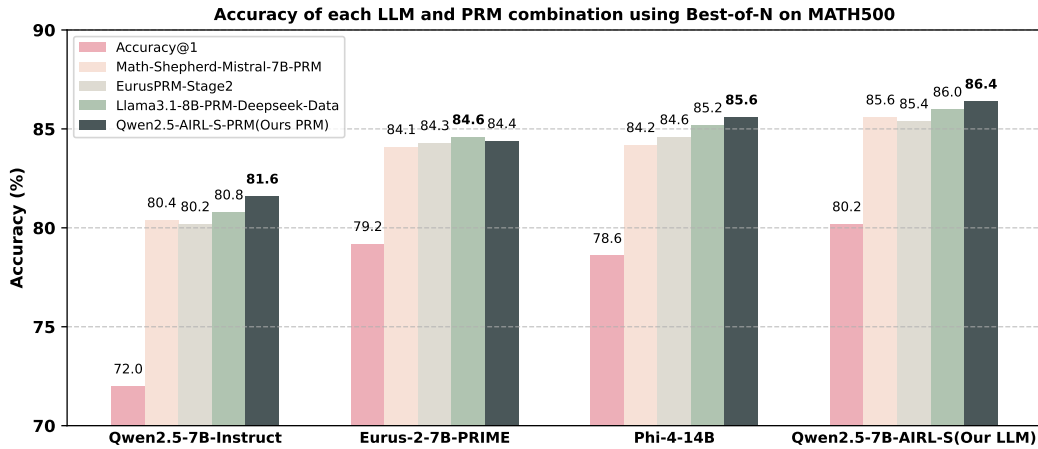


Figure 8: Performance of applying four PRMs on four different generation LLMs using Best-of-N with 64 rollouts on MATH500

(ii) *Educational equity*—a modest-sized model fine-tuned with our AIRL-S pipeline attains near-GPT-4o performance, enabling free or low-cost tutoring tools in under-resourced regions; (iii) *Software reliability*—the PRM can rank candidate patches during code repair, reducing latent defects in safety-critical systems.

Negative impacts and misuse scenarios. *Disinformation*: stronger mathematical/verbal reasoning could automate generation of persuasive but false technical content. *Fairness & bias*: reward models inherit biases present in demonstrations; these biases can be amplified by MCTS selection. *Dual-use*: chain-level reward shaping may help adversaries explore large search spaces (e.g. exploit generation, surveillance analytics). *Labour displacement*: automating competitive-programming-style tasks might displace junior software-engineering roles.