LLaMA-Berry: Pairwise Optimization for Olympiad-level Mathematical Reasoning via O1-like Monte Carlo Tree Search

Di Zhang[1,2*†], **Jianbo Wu**[3*], **Jingdi Lei**[2*†], **Tong Che**[4*]
Jiatong Li[5†], **Tong Xie**[6], **Xiaoshui Huang**[7], **Shufei Zhang**[2], **Marco Pavone**[8]
Yuqiang Li[2‡] , **Wanli Ouyang**[2], **Dongzhan Zhou**[2‡]
[1]Fudan University, [2]Shanghai Artificial Intelligence Laboratory
[3]University of California, Merced [4]Independent Researcher
[5]Hong Kong Polytechnic University, [6]University of New South Wales
[7]Shanghai Jiao Tong University, [8]Stanford University
{liyuqiang,zhoudongzhan}@pjlab.org.cn

Abstract

This paper presents LLaMA-Berry, an advanced mathematical reasoning framework to enhance the problem-solving ability of large language models (LLMs). The framework combines Monte Carlo Tree Search with Self-Refine (SR-MCTS) to optimize the reasoning paths and utilizes a pairwise reward model to evaluate different paths globally. By leveraging the self-critique and rewriting capabilities of LLMs, our SR-MCTS overcomes the inefficiencies and limitations of conventional stepwise and greedy search algorithms, enabling a more efficient exploration of solution spaces. To guide the search process, we propose the Pairwise Preference Reward Model (PPRM), which predicts pairwise preferences between solutions through instruction-following capabilities trained by Reinforcement Learning from Human Feedback (RLHF). Finally, the Enhanced Borda Count (EBC) method is adopted to synthesize pairwise preferences into global quantile scores for evaluations. This approach mitigates the challenges of scoring variability and non-independent distributions in mathematical reasoning tasks. The framework has been tested on general and advanced benchmarks, showing superior search efficiency and performance compared to existing open-source and closed-source methods, particularly in complex Olympiad-level benchmarks, including AIME24 and AMC23.

1 Introduction

Mathematical reasoning represents a great challenge in artificial intelligence, with broad applications across automated theorem proving, mathematical problem solving, and scientific discovery (Ahn et al., 2024). Recently, significant strides have been made by large language models (LLMs) like GPT-4 (Achiam et al., 2023) in general mathematical tasks involving arithmetic and geometric problem-solving (Cobbe et al., 2021; Sun et al., 2024; Ying et al., 2024). However, complex mathematical reasoning remains challenging, especially at the Olympiad-level benchmarks such as AIME (MAA, 2024).

An intuitive approach to improving problem-solving is to break solutions into step-by-step reasoning paths (Lightman et al., 2023; Luo et al., 2024a), as demonstrated in Chain-of-Thought (CoT Wei et al., 2022). While prompt-based methods can effectively facilitate the construction of such reasoning paths, they may still encounter challenges due to the lack of comprehensive feedback during the generation process, which can affect efficiency (Paul et al., 2023). In contrast to stepwise generation methods, another promising line of research treats the entire solution as an independent state, employing rewriting capabilities to refine the solutions, such as in Self-Refine (Madaan et al., 2023a) and Reflexion (Shinn et al., 2024). However, these approaches, while innovative, may occasionally face challenges like being susceptible to local optima or potentially drifting towards suboptimal solutions due to flawed feedback, which could impact their maximum potential performance.

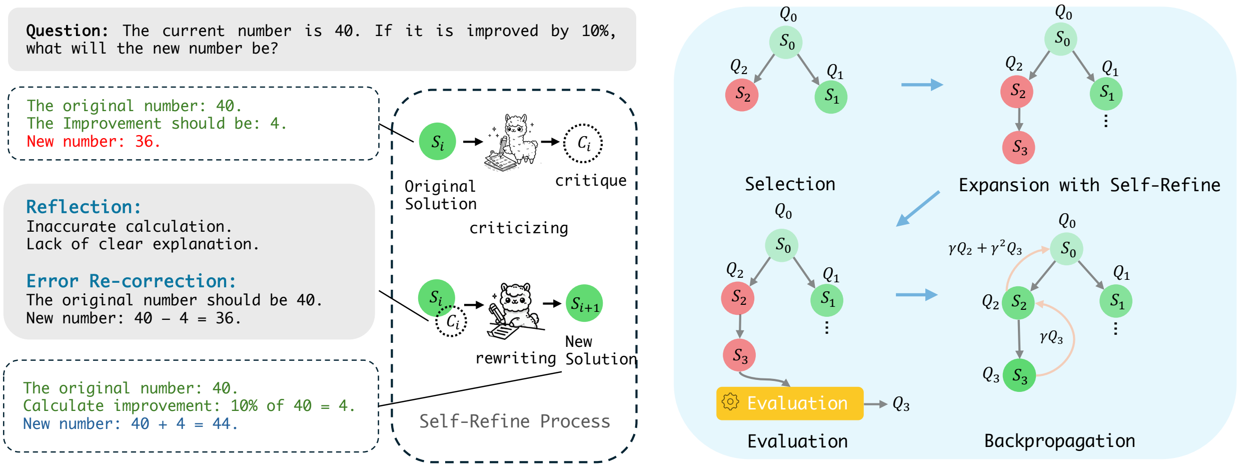
In addition to generating reasoning paths, effective solution evaluation is crucial, with models like the outcome reward model (ORM) and process reward model (PRM) (Uesato et al., 2022) serving as valuable examples. The ORM focuses on the correctness of the final answer in a reasoning path, while the PRM emphasizes the correctness of each step in the process. While both methods enable reward models to assign scalar scores, obtaining reliable labeled data for training these reward models remains a significant challenge. Moreover, the scoring standards for mathematical reasoning tasks can vary significantly, as each problem presents

[*] These authors contributed equally.

[†] This work was done during his internship at Shanghai Artificial Intelligence Laboratory.

[‡] Corresponding author

Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 7315–7337
April 29 - May 4, 2025 ©2025 Association for Computational Linguistics



Question: The current number is 40. If it is improved by 10%, what will the new number be?

The original number: 40.
The Improvement should be: 4.
New number: 36.

Reflection:
Inaccurate calculation.
Lack of clear explanation.

Error Re-correction:
The original number should be 40.
New number: 40 − 4 = 36.

The original number: 40.
Calculate improvement: 10% of 40 = 4.
New number: 40 + 4 = 44.

Figure 1: The main pipeline of LLaMA-Berry, where S_i stand for problem-solving solutions and C_i stands for critiques. The pipeline consists of four phases detailed in Section 2.2, including selection, expansion, evaluation, and backpropagation.

unique characteristics. This variation complicates the scaling of reward models and hinders their ability to capture local preference relations between solutions. Although trained using language models, these reward models have yet to fully leverage instruction-following capabilities, which may limit their effectiveness in handling more complex reasoning tasks (Zhang et al., 2024a).

To improve the efficiency of solution search in mathematical problems, we treat a complete solution as an independent state and apply Self-Refine to optimize previous solutions in order to obtain better ones. In the Self-Refine process, feedback from critiques is utilized to make the search more efficient compared to stepwise reasoning path generation. Furthermore, we incorporate Monte Carlo Tree Search (MCTS Kocsis and Szepesvári, 2006) to replace the iterative manner in naive Self-Refine, enhancing the solution search. MCTS leverages signals from the evaluation process to assess solutions and uses the Upper Confidence Bound applied to Trees (UCT) method to balance exploration and exploitation. This approach enables the search process to effectively exploit higher-quality solutions and explore those with greater potential for improvement, while avoiding getting trapped in suboptimal local minima.

In the evaluation process, utilizing the instruction-following capabilities trained by Reinforcement Learning from Human Feedback (RLHF Christiano et al., 2017), Pairwise Preference Reward Model (PPRM) transforms the absolute rewards calculation into preferences prediction between solutions to calculate rewards. The approach reduces the variability with scoring characteristics and thus leads to a more robust and consistent evaluation of different solutions. To overcome the locality limitations inherent in pairwise comparisons, we employ the Enhanced Borda Count (EBC) method to aggregate local preference evaluations into global quantile scores, leading to more informed decision-making and, ultimately, better solutions. Combining the PPRM and EBC method not only enables the reward model to learn a more robust reward signal but also captures the global characteristics of the solution space, ensuring more reliable comparisons.

Our contributions are summarized as follows: (1) We propose **SR-MCTS**, a novel Markov Decision Process (MDP) framework that treats entire solutions as states and Self-Refine as optimization action to perform advanced solution search with MCTS. (2) **PPRM** is developed to leverage the preference relationship between solutions to evaluate their quality, which avoids the volatility of absolute scores while providing a more guided exploration of optimal paths. We adopt the EBC method to convert the local preferences into global evaluations. (3) We verify the effectiveness of LLaMA-Berry on multiple benchmarks, which outperforms baseline approaches like ToT (Yao et al., 2024) and rStar (Qi et al., 2024) in both search efficiency and accuracy. Notably, LLaMA-Berry enhances the performance of LLaMA-3.1-8B, making it comparable to proprietary models, including GPT-4 Turbo on Olympiad-level mathematical reasoning **without additional training**.

2 Methodology

2.1 Preliminary

One of the core challenges in mathematical problem-solving is to generate and optimize reasoning paths to derive high-quality solutions. We formalize this process in a path-wise Markov Decision Process (MDP) framework, where each state s in the state space S represents a *complete solution* to a given problem, and the action space A consists of all feasible *rewriting* actions a that make transitions between states.

In the framework, we aim to quantify the expected reward $Q(s, a)$ from executing action a at state s, that is,

$$Q(s,a) = \mathbb{E}[R(s')|s' = T(s,a)], \tag{1}$$

where $T(s, a)$ indicates the transition from s to another solution s' via the rewriting action a. Our primary objective is to identify the optimal state s^* that represents the best solution. We can reach s^* by selecting actions that maximize the reward, guiding us toward the most desirable outcome, as demonstrated in Equation 2.

$$s^* = \arg\max_{s' \in S} Q(s') \tag{2}$$

2.2 Self-Refine applied to MCTS

As shown in Figure 1, SR-MCTS integrates Monte Carlo Tree Search (MCTS) with the Self-Refine mechanism to continuously evaluate and optimize the solution search. This integration leverages the iterative nature of MCTS and the self-improvement capabilities of LLMs, thereby improving the search outcomes.

Monte Carlo Tree Search (MCTS) is an effective method within the Markov Decision Processes (MDP) framework, employing states, actions, and value functions through sampling. The algorithm follows four key steps: selection, expansion, evaluation, and backpropagation. In the selection phase, the root node is expanded using the Upper Confidence Bound applied to Trees (UCT) algorithm, which selects a node s by balancing exploration and exploitation:

$$a = \arg\max_{a \in A(s)} \left(Q(s,a) + c \cdot \sqrt{\frac{\ln N(s)}{N(s,a)}} \right), \tag{3}$$

where $N(s)$ is the visitation count of node s, $N(s, a)$ is the action frequency, and c is a parameter controlling exploration. In the expansion phase, node s generates subsequent states s', added as new nodes in the tree $\mathcal{T}$. The evaluation phase typically uses simulations or heuristics to estimate the Q-values for these nodes. Finally, during backpropagation, the estimated Q-values are updated retroactively from the leaf nodes to the root. This iterative process allows MCTS to refine decision-making by balancing the exploration of new paths with the exploitation of known high-value paths.

Selection phase. The selection phase identifies a node s_i from the search tree $\mathcal{T}$ for expansion, where each node represents a complete solution state. The Upper Confidence Bound applied to Trees (UCT) algorithm is employed to select the optimal node, with dynamic pruning used to avoid local optima. A node s_i is considered fully explored when its child nodes reach a predefined limit, and at least one child node's Q value exceeds the Q value of s_i.

Expansion phase. In the expansion phase, as shown in Figure 1, the selected answer s_i is expanded by generating successor answers through a Self-Refine process, which includes a **Criticizing** and **Rewriting** process. The **Criticizing** process generates a critique $c_i = C(s_i)$ that identifies drawbacks (e.g., mathematical wrongs or logical faults) in the current chosen answer S_i, and then **Rewriting** process generates a new answer $s_{i+1} = R(s_i, c_i)$. In practice, to simplify the problem, we assume this process is deterministic, ensuring that the same original state of solutions s_i consistently produces the same successor state of solution s_{i+1}. The new state of solution s' is then added to the search tree $\mathcal{T}$ as a new node.

Evaluation phase. The evaluation phase calculates the value $Q(s')$ of the newly generated node s' using the Pairwise Preference Reward Model (PPRM). The evaluation involves two steps: global and local value assessments. The global value $Q_g(s')$ is determined by the quantile of s' in a win-loss preference matrix $\mathbf{M}$, which reflects the win-loss relationships between nodes. The local value $Q_l(s')$ is derived from comparisons with adjacent nodes in the search tree $\mathcal{T}$. The total value $Q(s')$ is then computed as a weighted combination of global and local values: $Q(s') = \alpha Q_g(s') + (1 - \alpha)Q_l(s')$, where α controls the relative influence of each component.

Backpropagation phase. In the backpropagation phase, the value $Q(s')$ of the new node is propagated back to its parent node s_i, updating Q value

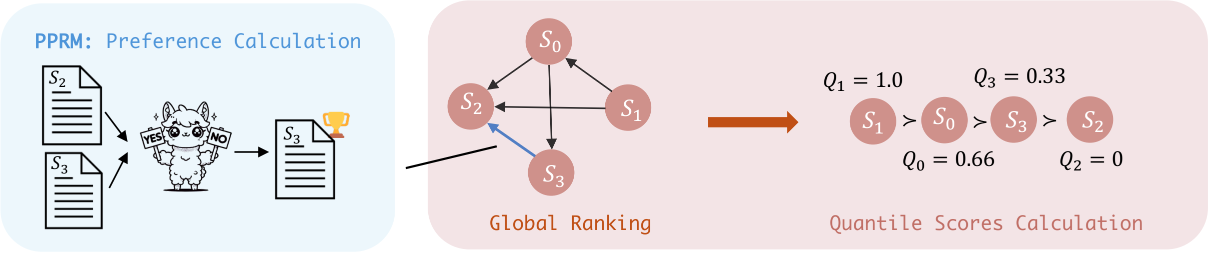


Figure 2: Preference prediction process of PPRM and global quantile score based on Enhanced Borda Count method.

of s_i as a discounted sum of its child nodes' Q values: $Q(s_i) = (1-\gamma)Q(s_i) + \gamma Q(s')$. The discount factor γ represents the importance of future rewards. This iterative update mechanism ensures that the values of parent nodes are progressively refined, enhancing the guidance for future selections.

Additionally, to control the growth of the search tree, the SR-MCTS method restricts the maximum number of rollouts N_{max} . The search process terminates when the restriction is reached, imposing limits on the unbounded expansion of the tree. The overarching objective of SR-MCTS is to maximize the expected highest Q value of all existing nodes S, guiding us towards the most desirable outcome s^*, ensuring that the search process efficiently converges to high-quality solutions.

2.3 Pairwise Preference Reward Model

Reliable evaluation of different solutions plays a crucial role in mathematical problem-solving tasks as it leads to better estimation of Q-values, thereby offering better guidance. Existing reward models typically evaluate solutions by giving absolute scores, such as process reward model (PRM Lightman et al., 2023) and outcome reward model (ORM Yu et al., 2023). However, the score-based reward models may fall short in leveraging the instruction-following capability of LLMs or effectively handling the variations in scoring standards, especially when the differences between solutions are subtle. To address this, we propose the Pairwise Preference Reward Model (PPRM), which leverages a comprehensive preference dataset incorporating substantial samples from both PRM and ORM approaches (Toshniwal et al., 2024; Lightman et al., 2023) to learn preference relationships among mathematical solutions.

For two solutions (a_1 and a_2) to a given mathematical problem, we use $a_1 \succ a_2$ to represent the situation where a_1 is preferred over a_2. PPRM predicts their relative quality using a pairwise partial ordering, represented by the following probability formula:

$$P(a_1 \succ a_2 \mid \phi) = \frac{e^{\phi(a_1)}}{e^{\phi(a_1)} + e^{\phi(a_2)}}, \tag{4}$$

where $P(a_1 \succ a_2 \mid \phi)$ denotes the probability of a partial ordering relation between solution a_1 and a_2, with ϕ representing the parameters of the reward model. In our method, $a_1 \succ a_2$ are represented by tokens of an LLM, and $P(a_1 \succ a_2 \mid \phi)$ is estimated using the logits value of tokens calculated by the LLM.

Then, inspired by advancements in Language Interface Fine-Tuning (LIFT Dinh et al., 2022), we frame the training process of PPRM as a question-answering task to leverage the instruction-following capabilities of LLMs. The model is tasked with answering the question, "For Question Q, is solution a_1 better than solution a_2?" as shown in Figure 2. To form a robust training objective, the predicted token labels $\hat{y}$ ('Yes' or 'No') are evaluated with ground truth label y using the indicator function $\mathbf{I}$:

$$\mathbf{I}(\hat{y}, y) = \begin{cases} 1, & \text{if } \hat{y} = y \\ 0, & \text{if } \hat{y} \neq y \end{cases} \tag{5}$$

Finally, a pairwise preference dataset $\mathcal{D}$ that contains millions of mathematical problem-solving solution pairs is converted into a dataset $\mathcal{D}'$ suitable for a question-answering task. We employ RLHF techniques to train the model to improve its performance in the partial-order prediction question-answering task. Subsequently, the Direct Preference Optimization (DPO Rafailov et al., 2024) method is utilized to find the optimal P_ϕ by maximizing the objective $\arg\max_\phi \mathbb{E}_P[\mathbf{I}(\hat{y}, y)]$. Please refer to Appendix C for details about training and inference of PPRM.

2.4 Enhanced Borda Count Method

Although PPRM allows us to directly compare the quality of two solutions, we still need to convert these local preferences into a cohesive global ranking to gain a comprehensive evaluation for the answers. This conversion process can be formalized as the global optimal ranking aggregation (GORA) problem related to Learning to Rank (LtR) methods. Further, we propose the Enhanced Borda Count (EBC) method based on the transitivity assumption of mathematical problem solutions, which integrates the naive Borda Count algorithm with a transitive closure of preferences calculated by the Floyd-Warshall (Warshall, 1962) algorithm. For formalized discussion, please refer to Appendix H.

Local preference calculation. First, the PPRM generates a preference matrix $\mathbf{M} \in \mathbb{R}^{n \times n}$ for all n problem solutions, where $\mathbf{M}_{i,j} = 1$ indicates that solution a_i is superior to solution a_j, and $\mathbf{M}_{i,j} = 0$ otherwise. This process can be represented as:

$$\mathbf{M}_{i,j} = \begin{cases} 1, & \text{if } P(a_i \succ a_j) \geq 0.5 \\ 0, & \text{if } P(a_i \succ a_j) < 0.5 \end{cases} \tag{6}$$

As shown in Figure 2, this matrix can be viewed as an adjacency matrix of a directed graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, where each solution a_i corresponds to a vertex v_i, and an edge $e = (v_i, v_j)$ exists if $\mathbf{M}_{i,j} = 1$, indicating that solution a_i is preferred over a_j.

Transitive closure. To simplify the problem, we adopt the assumption of transitivity for mathematical solutions, that is, if $v_i \succ v_k$ and $v_k \succ v_j$, then $v_i \succ v_j$. Under this assumption, the transitive closure $\mathbf{C}$ of a preference matrix can be computed through Floyd-Warshall algorithm, e.g., if $\mathbf{M}_{i,k} = 1$ and $\mathbf{M}_{k,j} = 1$, then $\mathbf{M}_{i,j} = 1$.

Borda count-based global ranking. Next, based on the transitive closure matrix $\mathbf{C}$, we apply the Enhanced Borda Count method for global ranking. The Enhanced Borda Count determines the ranking of each node by calculating its out-degree, which corresponds to the number of nodes it defeats. For each node v_i, the Borda(v_i) is defined as $\sum_{j=1}^{n} \mathbf{C}_{i,j}$, like the ranked node listed in Figure 2.

Nodes with higher Borda counts are ranked higher. However, in practice, cyclic preferences can cause efficiency issues with the naive Borda Count method. To further refine the ranking, we devise a re-ranking stage, where the logits generated by the PPRM are used for soft comparison among nodes with equal Borda counts. Specifically, for two nodes v_i and v_j with equal Borda counts, the soft comparison rule can be denoted as $v_i \succ v_j \iff P(a_i \succ a_j) > P(a_j \succ a_i)$. This process ensures that the ranking remains consistent and reasonable even in the presence of cycles or local ambiguities.

Global quantile score of solutions. Finally, the ranking is converted into a global quantile score Q_g for each solution v is $Q_g(v) = 1 - \frac{\text{rank}(v)-1}{|V|-1}$, where rank$(v)$ is the position of v in the ranking based on Borda counts, and $|V|$ is the total number of nodes. To reflect local advantages in the search tree structure, the local win rate $Q_l(v)$ for a node v is calculated in $\mathbf{C}$ with its children nodes Children$_v$ as follows:

$$Q_l[v] = \frac{\sum_{u \in \text{Children}[v]} C[u, v]}{|\text{Children}[v]|} \tag{7}$$

Finally, score $Q(v)$ for a solution is a weighted sum of local win rate $Q_l(v)$ and global quantile score Q_g.

3 Evaluation

3.1 Experiment Settings

Settings. To better evaluate the effectiveness of our approach, we select LLaMA-3.1-8B-Instruct model (Meta, 2024b) as the base model for SR-MCTS, **without any additional training**. We also train a Gemma2-2B-Instruct model (Google, 2024) as PPRM to provide reward signals during the search process. We develop the `Berry-Tree` inference framework to ensure robust and efficient inference, which supports advanced features, including fault tolerance, checkpoint recovery, multi-query concurrency, and automatic multi-server load balancing. Hyper-parameter settings are detailed in Appendix A.

Grading. We evaluate algorithm-generated answers using the correctness evaluation standards as in Lightman et al. (2023), focusing on format adherence and content accuracy. The model is provided with a prompt specifying the expected answer format. We score answers as consistent if they exactly match the ground truth, closely approximate it numerically, or are equivalent in symbolic form. To ensure a comprehensive and rigorous evaluation, we adopt **major@k** (Kuncheva, 2014) and **rm@k** (Yang et al., 2024c), which can be unified as the solved rate of problems (Lightman et al., 2023; Luo et al., 2024a). The evaluation methods and metric details are further outlined in Appendix B.

Benchmark Model	GSM8K	MATH	GaoKao2023En	OlympiadBench	College Math	MMLU STEM
Qwen2-7B-Instruct (Yang et al., 2024a)	85.7	52.9	36.4	21.3	24.5	68.2
Meta-Llama-3.1-8B-Instruct (Meta, 2024b)	76.6	47.2	30.1	15.4	33.8	60.5
Qwen2-72B-Instruct (Yang et al., 2024a)	93.2	69.0	58.7	33.2	43.2	84.4
Meta-Llama-3.1-70B-Instruct (Meta, 2024a)	94.1	65.7	54.0	27.7	42.5	80.4
DeepSeekMath-7B-RL (Shao et al., 2024)	88.2	52.4	43.6	19.0	37.5	64.8
Internlm2-math-plus-7b (Ying et al., 2024)	84.0	54.4	50.1	18.8	36.2	55.2
Mathstral-7B-v0.1 (Mistral AI, 2024)	84.9	56.6	46.0	21.5	33.7	64.0
NuminaMath-7B-CoT (Beeching et al., 2024b)	75.4	55.2	47.5	19.9	36.9	60.8
Qwen2-Math-7B-Instruct (Yang et al., 2024a)	89.9	75.1	62.1	38.2	45.9	63.8
NuminaMath-72B-CoT (Beeching et al., 2024a)	90.8	66.7	58.4	32.6	39.7	64.5
Qwen2-Math-72B-Instruct (Yang et al., 2024a)	96.7	84.0	68.3	43.0	47.9	79.9
Meta-Llama-3.1-8B-Instruct (Meta, 2024b)	$89.8_{\texttt{maj@8}}$	$54.8_{\texttt{maj@8}}$	$36.4_{\texttt{maj@8}}$	$24.8_{\texttt{maj@8}}$	$36.4_{\texttt{maj@8}}$	$68.3_{\texttt{maj@8}}$
+ LLaMA-Berry (Ours)@8	$94.9_{\texttt{rm@8}}$	$69.4_{\texttt{rm@8}}$	$61.6_{\texttt{rm@8}}$	$47.2_{\texttt{rm@8}}$	$63.7_{\texttt{rm@8}}$	$82.9_{\texttt{rm@8}}$
+LLaMA-Berry (Ours)@16	$96.1_{\texttt{rm@16}}$	$75.3_{\texttt{rm@16}}$	$68.6_{\texttt{rm@16}}$	$55.1_{\texttt{rm@16}}$	$68.9_{\texttt{rm@16}}$	$88.3_{\texttt{rm@16}}$

Table 1: Performance comparison of models across benchmarks of different difficulties, as represented by GaoKao2023En (Liao et al., 2024), College Math (Tang et al., 2024), and OlympiadBench (He et al., 2024), which range from high school to Olympiad levels. Scores denoted with subscripted notations, such as `maj@8`, represent specific metrics, with **major@8** as an example. Scores without subscripted notations reflect the model's greedy performance evaluated in a zero-shot CoT manner.

3.2 Benchmarks

General mathematical reasoning benchmarks. We summarize the results on general mathematical reasoning benchmarks in Table 1, which indicates that our method significantly boosts the base model's performance. The results consistently demonstrate improvements across various levels of difficulty. Specifically, the solved rate of problems in 16 rollouts of Meta-Llama-3.1-8B-Instruct (Meta, 2024b) has been improved by more than 35% on three benchmarks. Qwen2-Math-72B-Instruct (Yang et al., 2024b) exhibits the strongest mathematical reasoning capability among the competing methods, while our `LLaMA-Berry`, built on a base model with only 8B parameters, exceeds it in the solved rate of problems on four benchmarks. In particular, `LLaMA-Berry` reaches 55.1% on OlympiadBench and 68.9% on College Math, surpassing it by 11.9% and 21%, respectively.

Cutting-edge mathematical Olympiad benchmarks. In Table 2, we compare the performance of `LLaMA-Berry` with other leading models on Olympic-level benchmarks. The results demonstrate that `LLaMA-Berry` is highly competitive on these benchmarks, demonstrating its capability in complex reasoning. Notably, on the most challenging AIME2024 benchmark, our method boosts the base model's solving rate from 2/30 to 8/30, surpassing typical open-source models and commercial closed-source models, except for the OpenAI o1 series (OpenAI, 2024).

In addition to excelling in mathematical reasoning, our approach also excels across various science and engineering domains. For example, it achieves top performance on benchmarks such as MMLU STEM (Hendrycks et al., 2021a) in Table 1 and GPQA diamond (Rein et al., 2024) in Table 2. This demonstrates the method's robustness and versatility, which highlights its potential for broader applications in both research and practical scenes.

Model	Benchmarks			
	AIME24	AMC23	Math Odyssey	$\text{GPQA}_{\texttt{Diamond}}$
Claude 3 Opus	6.7	42.0	40.6	50.4
GPT 4 Turbo	3.3	–	47.0	38.8
GPT 4o	13.4	–	–	56.1
OpenAI o1 Preview	56.7	–	–	78.3
OpenAI o1	83.3	–	–	78.0
Gemini 1.5 Pro	6.7	–	45.0	–
Gemini Math-Specialized 1.5 Pro	23.3	–	55.8	–
Meta-LLaMA-3.1-8B-Instruct	6.7	15.7	41.7	30.4
Qwen2-Math-7B-Instruct	13.3	62.5	–	–
NuminaMath-72B CoT	3.3	52.5	–	–
Qwen2-Math-72B-Instruct	20.0	60.0	–	–
Meta-Llama-3.1-8B-Instruct	$13.3_{\texttt{maj@8}}$	$22.9_{\texttt{maj@8}}$	$44.2_{\texttt{maj@8}}$	$39.4_{\texttt{maj@8}}$
+ LLaMA-Berry (Ours)@8	$16.7_{\texttt{rm@8}}$	$48.2_{\texttt{rm@8}}$	$60.4_{\texttt{rm@8}}$	$77.3_{\texttt{rm@8}}$
+LLaMA-Berry (Ours)@16	$26.7_{\texttt{rm@16}}$	$54.2_{\texttt{rm@16}}$	$65.0_{\texttt{rm@16}}$	$92.4_{\texttt{rm@16}}$

Table 2: Performance comparison across multiple olympiad benchmarks, including AIME24 (MAA, 2024), AMC23 (MAA, 2023), Math Odyssey (Fang et al., 2024), and GPQA Diamond (Rein et al., 2024).

Comparison with other tree-based or CoT methods. We compare our algorithm with other tree-based reasoning methods and CoT-based methods on GSM8K (Cobbe et al., 2021), GSMHard (Gao et al., 2022), and MATH500 (Lightman et al., 2023) which is a representative and highly challenging 10% subset of MATH (Hendrycks et al., 2021b) benchmark. As shown in Table 3, the performance of RAP (Hao et al., 2023) and ToT (Yao et al., 2023) tends to degrade relative to more straightforward methods like Few-shot CoT and One-turn Self-

Method \ Benchmark	GSM8K	GSMHARD	MATH500
Zero-Shot CoT	68.4	14.9	5.8
Few-shot CoT	74.5	25.6	17.8
One-turn Self-Refine	75.7	26.5	25.0
Self-Cons@8	78.4	28.5	30.0
Self-Cons@64	83.2	30.3	33.0
Self-Cons@128	84.7	31.2	33.8
ToT@32	69.1	19.6	13.6
RAP@32	80.6	29.6	18.8
rStar@32	$88.7_{\texttt{maj@32}}$	$33.4_{\texttt{maj@32}}$	$38.3_{\texttt{maj@32}}$
LLaMA-Berry (Ours)@8	$86.4_{\texttt{maj@8}}$ $94.1_{\texttt{rm@8}}$	$30.2_{\texttt{maj@8}}$ $37.1_{\texttt{rm@8}}$	$35.2_{\texttt{maj@8}}$ $56.4_{\texttt{rm@8}}$
LLaMA-Berry (Ours)@16	$88.1_{\texttt{maj@16}}$ $96.4_{\texttt{rm@16}}$	$31.5_{\texttt{maj@16}}$ $41.1_{\texttt{rm@16}}$	$39.6_{\texttt{maj@16}}$ $63.8_{\texttt{rm@16}}$
Meta-Llama-3.1-70B-Instruct	91.0	50.0	66.0
Meta-Llama-3.1-70B-Instruct + LLaMA-Berry (Ours)@8	$91.7_{\texttt{maj@8}}$ $96.2_{\texttt{rm@8}}$	$52.3_{\texttt{maj@8}}$ $59.1_{\texttt{rm@8}}$	$70.0_{\texttt{maj@8}}$ $76.0_{\texttt{rm@8}}$

Table 3: Performance of different tree-based methods for LLaMA-3.1-8B-Instruct and LLaMA-3.1-70B-Instruct on GSM8K, GSMHARD, and MATH500 benchmarks. The 70B model is evaluated using a 10% subset of benchmarks due to the limitation of computation resources.

Refine when the difficulty increases from GSM8K to GSMHard. We suspect the reason could be the weak self-evaluation capability of LLMs, which may guide reasoning steps to the inefficient side. Moreover, tree-based methods can incur more computational overhead than straightforward methods. In contrast, rStar (Qi et al., 2024) and our method maintain a positive output performance trend, highlighting both approaches' higher search efficiency.

To make a fair comparison between the reported results on LLaMA-3-8B-instruct from rStar (Qi et al., 2024), self-consistency (Wang et al., 2022), and our algorithm, we also utilize the LLaMA-3-8B-instruct as the base model instead of the 3.1 version. We observe that our approach achieves on-par or even better performance with fewer rollouts. Specifically, our method achieves an accuracy of 88.1%, 31.5%, and 39.6% on GSM8K, GSMHARD, and MATH500 benchmarks, respectively, using the majority voting metric, which is among the same accuracy level as others while only consuming 1/2 of the rollout times of rStar and 1/8 of Self-consistency. This provides compelling validation of the efficacy of the EBC method and the aggressive exploration fostered by the dynamic pruning strategy.

3.3 Ablation Study

As shown in Table 4, we conduct ablation experiments to evaluate the key components of LLaMA-Berry, using the solved rate of problems (Luo et al., 2024a) as a metric across benchmarks of increasing difficulty: GSM8K, MATH500 and AIME2024. Major Voting method represents the base model's reasoning capabilities. CoT-based MCTS which is a step-level MCTS is a comparison with SR-MCTS method. SR-MCTS without PPRM refers to a basic version of our method that uses self-evaluation as the reward instead of PPRM and EBC.

Method \ Benchmark	GSM8K	MATH500	AIME2024
Multi-turn Self-Refine with Self-Verification	78.9	40.4	6.7
Major Voting with Random Repeated Sampling	$88.3_{\texttt{maj@8}}$	$46.8_{\texttt{maj@8}}$	$10_{\texttt{maj@8}}$
Step-level CoT-based MCTS (Process Self-Reward)	$70.7_{\texttt{rm@8}}$	$33.2_{\texttt{rm@8}}$	$6.7_{\texttt{rm@8}}$
SR-MCTS without PPRM (Outcome Self-Reward)	$82.0_{\texttt{rm@8}}$	$37.2_{\texttt{rm@8}}$	$6.7_{\texttt{rm@8}}$
SR-MCTS (with PPRM)	$95.0_{\texttt{rm@8}}$	$69.0_{\texttt{rm@8}}$	$16.7_{\texttt{rm@8}}$

Table 4: Ablation study comparing different methods on GSM8K, MATH500, and AIME2024 benchmarks with LLaMA-3.1-8B-Instruct.

When comparing the two distinct tree search node expansion strategies, namely step-level CoT and Self-Refine, it is evident that the Self-refine method surpasses the step-level CoT approach across all datasets under the same number of rollouts. This advantage is further amplified when employing PPRM, as exemplified by the improvement from 37.2% to 69.0% on the MATH500 dataset.

When comparing multi-turn Self-Refine with SR-MCTS methods, especially on the GSM8K dataset, the introduction of MCTS effectively mitigates the issue of solution degradation into suboptimal results caused by flawed critiques in iterative methods. As shown in Table 4, with rollouts of 8, SR-MCTS without PPRM improves the problem-solving rate by 3.1%, while SR-MCTS with PPRM further enhances the problem-solving rate by 16.1%. On the more challenging MATH500 dataset, the advantage of SR-MCTS with PPRM becomes even more pronounced, achieving a 28.6% improvement in the problem-solving rate. Furthermore, in comparing SR-MCTS without PPRM and SR-MCTS with PPRM, PPRM further boosts the solved rate of problems on GSM8K and MATH500 datasets, the solved rate of problems is elevated by 13% and 31.8%, respectively.

Notably, on the more challenging AIME2024 dataset, both Multi-turn Self-Refine and SR-MCTS without PPRM demonstrate limited search efficiency. However, SR-MCTS with PPRM can significantly improve the solved rate of problems from 6.7% (2/30) to 16.7% (5/30) with rollouts of 8. The results underscore the efficacy of combining the Self-Refine method with PPRM when addressing complex problems. The contrast between self-

Benchmark α	GSMHARD	MATH500
0.0 (global scores only)	$44.9_{\texttt{rm@8}}$	$68.2_{\texttt{rm@8}}$
0.9 (default)	$43.8_{\texttt{rm@8}}$	$69.0_{\texttt{rm@8}}$
1.0 (local scores only)	$44.2_{\texttt{rm@8}}$	$67.9_{\texttt{rm@8}}$

Table 5: The results with 8 rollouts of varying the hyper-parameter α in the reward mechanism of PPRM.

reward and PPRM underscores the importance of designing reward mechanisms that can more effectively guide the search process. PPRM provides a more holistic incentive to the model, thus fostering more effective problem-solving strategies.

Our ablation study on LLaMA-3-1-8B-Instruct reveals that the hyper-parameter α, which balances local and global reward scores in PPRM, has scenario-dependent impacts. For simpler tasks like GSMHARD, global scores dominate, as evidenced by the superior performance of LLaMA-Berry when prioritizing global metrics. This suggests that holistic evaluation is sufficient for straightforward problems. In contrast, complex tasks like MATH500 benefit from a balanced combination of local and global signals—the MATH500 benchmark shows a significant improvement (from 37.2% to 69.0%) when both metrics are optimally weighted. These results indicate that while α has a limited impact on simpler tasks, its tuning is crucial for complex tasks requiring both granular and global reasoning. These findings suggest that future work could focus on optimizing α adaptively for different task complexities.

3.4 Scaling Study

To explore the potentials and trends of the scaling with rollouts in inference-time, we depict the solved rate of problems with rollouts in three benchmarks with different difficulty levels. Analyzing the performance alongside Figure 3, the increment in the number of rollouts consistently enhances model performance across various benchmarks, and the extent of these improvements differs depending on the benchmark's complexity and the base model's reasoning capability. These curves underscore that the performance of the `LLaMA-Berry` framework benefits from scaling up rollouts during inference, similar to the observations in OpenAI (2024). However, there are ceiling limitations, as seen in the GSM8K dataset, which suggest that the base model's capabilities in both reasoning and refinement play a crucial role in determining the overall performance.

4 Related Works

Reward models for reasoning. Reliable reward models (Kang et al., 2024; Wang et al., 2023; Havrilla et al., 2024; Lightman et al., 2023; Ma et al., 2023) can effectively distinguish desirable responses from undesirable ones, which is especially important in complex reasoning. The outcome reward model (ORM) are trained with the final results of the reasoning paths. As the rewards are determined by the final answers, ORM may suffer from coarse supervision and misalignment issues. In contrast, process reward model (PRM) provides step-wise reward signals that are easier to interpret and guide the models to follow the CoTs. Therefore, PRM is generally considered to be more effective (Lightman et al., 2023). However, the success of PRM relies on extensive manually annotated data (Luo et al., 2024a; Havrilla et al., 2024), which is time-consuming and still faces the challenge of the volatility of absolute reward scores. Pairwise Preference Reward Model (PPRM) in `LLaMA-Berry` converts absolute scoring into preference prediction task, which then brings robust reward signals via EBC method.

Tree search reasoning. Sampling diverse reasoning paths (Brown et al., 2024) has demonstrated its effectiveness in enhancing the probability of finding the correct answers. Self-Consistency (Wang et al., 2022) samples a complete path each time while tree search methods like Tree-of-Thought (ToT) (Yao et al., 2023) and Monte Carlo Tree Search (MCTS) (Chen et al., 2024a,b; Luo et al., 2024b; Feng et al., 2023; Xie et al., 2024; Xu, 2023; Liu et al., 2023; Tian et al., 2024; Ding et al., 2023) extend multiple steps to optimize step answers and ultimately obtain the optimal solution. Additionally, Self-Refine (Madaan et al., 2023a) method has become a recent focus. Self-verification (Gero et al., 2023; Weng et al., 2022) and rStar (Qi et al., 2024) utilize the inherent capabilities of the model to iteratively explore and refine answers. However, the performance of Self-Refine is typically constrained by the inherent capabilities of the model, especially for small language models (SLMs) with significantly weaker Self-Refine abilities (Madaan et al., 2023b). Zhang et al. (2024b) suggests that the mathematical reasoning abilities of LLMs can be enhanced by treating the refinement process as a directed acyclic graph (DAG) through multi-agent

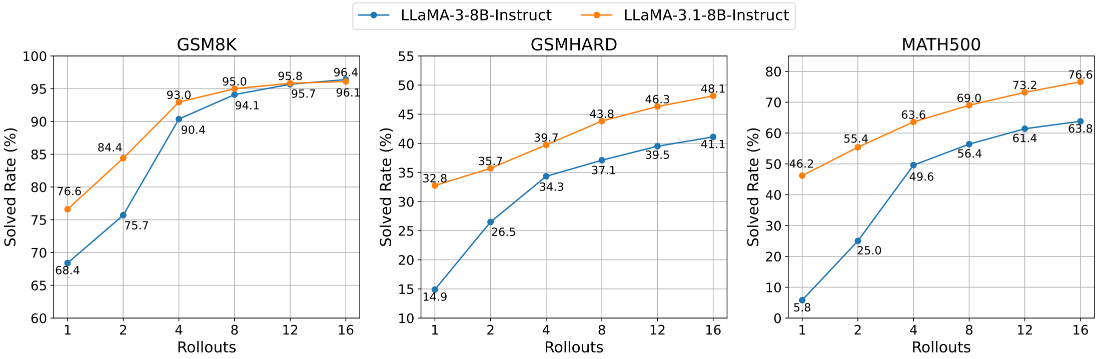


Figure 3: Scaling of inference-time rollouts.

collaboration. In our approach, we combine MCTS with Self-Refine to explore potential solutions and a global win-loss matrix is then constructed in the form of a directed graph to calculate the final quantile scores.

5 Conclusion

This research addresses challenges in complex mathematical reasoning, particularly at the Olympiad level, by enhancing the accuracy and efficiency of the search for reasoning paths. By introducing Self-Refine applied to Monte Carlo Tree Search (SR-MCTS), the `LLaMA-Berry` framework significantly improves the efficiency of solution generation by LLMs. Additionally, the Pairwise Preference Reward Model (PPRM) constructs preferences between solutions rather than merely scoring outcomes, calculating the final global quantile score using the enhanced Borda Count (EBC) method. Evaluation results demonstrate that `LLaMA-Berry` outperforms baseline approaches on benchmarks like GSM8K and MATH, and achieves competitive performance compared to closed-source models on Olympiad-level benchmarks such as AIME2024.

Limitation

The `LLaMA-Berry` framework has demonstrated strong performance in reasoning tasks, but there are still some challenges in practical applications. First, methods such as Monte Carlo Tree Search (MCTS) and Self-Refine have high computational costs. These techniques demand significant computational resources, which may limit the feasibility of deployment in environments with constrained computational capacity. As for summarizer of solutions, rule-based heuristics methods like self-consistency, major voting and mutual reasoning have shown a constraint to the ceiling search performance of MCTS. Thus, we aim to develop a learning-based summarizer as OpenAI (2024) does to further enhance the search efficiency.

Furthermore, the current evaluation of the `LLaMA-Berry` framework has primarily focused on mathematical reasoning benchmarks, resulting in a relatively narrow assessment scope. As a result, its performance in broader domains, such as general knowledge, symbolic logic tasks, and multimodal applications, has not been sufficiently validated. In future work, we aim to improve the framework by evaluating it on a more diverse set of tasks to enhance its applicability.

Lastly, most experiments so far have utilized relatively small open-source models, with limited testing on larger or closed-source models. In future research, we plan to investigate the performance of `LLaMA-Berry` on larger models, particularly addressing challenges related to scaling and performance optimization.

Acknowledgments

This work is supported by the Shanghai Municipal Science and Technology Major Project. This work is also supported by Shanghai Artificial Intelligence Laboratory.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. 2024. Large language models for mathematical reasoning: Progresses and challenges. *arXiv preprint arXiv:2402.00157*.

Anthropic. 2024. The Claude 3 Model Family: Opus, Sonnet, Haiku. Technical report, Anthropic. Accessed: 2024-09-19.

Edward Beeching, Shengyi Costa Huang, Albert Jiang, Jia Li, Benjamin Lipkin, Zihan Qina, Kashif Rasul, Ziju Shen, Roman Soletskyi, and Lewis Tunstall. 2024a. Numinamath 72b cot. https://huggingface.co/AI-MO/NuminaMath-72B-CoT.

Edward Beeching, Shengyi Costa Huang, Albert Jiang, Jia Li, Benjamin Lipkin, Zihan Qina, Kashif Rasul, Ziju Shen, Roman Soletskyi, and Lewis Tunstall. 2024b. Numinamath 7b cot. https://huggingface.co/AI-MO/NuminaMath-7B-CoT.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*.

Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. 2024a. Alphamath almost zero: process supervision without process. *arXiv preprint arXiv:2405.03553*.

Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. 2024b. Step-level value preference optimization for mathematical reasoning. *arXiv preprint arXiv:2406.10858*.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Ruomeng Ding, Chaoyun Zhang, Lu Wang, Yong Xu, Minghua Ma, Wei Zhang, Si Qin, Saravan Rajmohan, Qingwei Lin, and Dongmei Zhang. 2023. Everything of thoughts: Defying the law of penrose triangle for thought generation. *arXiv preprint arXiv:2311.04254*.

Tuan Dinh, Yuchen Zeng, Ruisu Zhang, Ziqian Lin, Michael Gira, Shashank Rajput, Jy-yong Sohn, Dimitris Papailiopoulos, and Kangwook Lee. 2022. Lift: Language-interfaced fine-tuning for non-language machine learning tasks. *Advances in Neural Information Processing Systems*, 35:11763–11784.

Meng Fang, Xiangpeng Wan, Fei Lu, Fei Xing, and Kai Zou. 2024. Mathodyssey: Benchmarking mathematical problem-solving skills in large language models using odyssey math data. *Preprint*, arXiv:2406.18321.

Xidong Feng, Ziyu Wan, Muning Wen, Stephen Marcus McAleer, Ying Wen, Weinan Zhang, and Jun Wang. 2023. Alphazero-like tree-search can guide large language model decoding and training. *arXiv preprint arXiv:2309.17179*.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2022. Pal: Program-aided language models. *arXiv preprint arXiv:2211.10435*.

Zelalem Gero, Chandan Singh, Hao Cheng, Tristan Naumann, Michel Galley, Jianfeng Gao, and Hoifung Poon. 2023. Self-verification improves few-shot clinical information extraction. *arXiv preprint arXiv:2306.00024*.

Google. 2024. Gemma-2b-it. https://huggingface.co/google/gemma-2b-it. Accessed: 2024-09-19.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*.

Alex Havrilla, Sharath Raparthy, Christoforus Nalmpantis, Jane Dwivedi-Yu, Maksym Zhuravinskyi, Eric Hambro, and Roberta Raileanu. 2024. Glore: When, where, and how to improve llm reasoning via global and local refinements. *arXiv preprint arXiv:2402.10963*.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Jikun Kang, Xin Zhe Li, Xi Chen, Amirreza Kazemi, Qianyi Sun, Boxing Chen, Dong Li, Xu He, Quan He, Feng Wen, Jianye Hao, and Jun Yao. 2024. Mindstar: Enhancing math reasoning in pre-trained llms at inference time. *arXiv preprint arXiv:2405.16265*.

Levente Kocsis and Csaba Szepesvári. 2006. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer.

Ludmila I Kuncheva. 2014. *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons.

Minpeng Liao, Wei Luo, Chengxi Li, Jing Wu, and Kai Fan. 2024. Mario: Math reasoning with code interpreter output – a reproducible pipeline. *arXiv preprint arXiv:2401.08190*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Jiacheng Liu, Andrew Cohen, Ramakanth Pasunuru, Yejin Choi, Hannaneh Hajishirzi, and Asli Celikyilmaz. 2023. Don't throw away your value model! generating more preferable text with value-guided monte-carlo tree search decoding. *arXiv preprint arXiv:2309.15028*.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. 2024a. Improve mathematical reasoning in language models by automated process supervision. *Preprint*, arXiv:2406.06592.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, et al. 2024b. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*.

Qianli Ma, Haotian Zhou, Tingkai Liu, Jianbo Yuan, Pengfei Liu, Yang You, and Hongxia Yang. 2023. Let's reward step by step: Step-level reward model as the navigators for reasoning. *arXiv preprint arXiv:2310.10080*.

MAA. 2023. American mathematics competitions. Online.

MAA. 2024. American invitational mathematics examination. Online.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023a. Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023b. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, pages 46534–46594. Curran Associates, Inc.

Meta. 2024a. Meta-llama 3.1-70b instruct. Accessed: 2024-09-03.

Meta. 2024b. Meta-llama 3.1-8b instruct. Accessed: 2024-09-03.

Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. 2017. SymPy: symbolic computing in Python. *PeerJ Computer Science*, 3:e103. Publisher: PeerJ Inc.

Mistral AI. 2024. Mathstral. `https://mistral.ai/news/mathstral`. Accessed: 2024-08-12.

OpenAI. 2024. Introducing openai o1-preview. Online.

OpenAI. 2024. Learning to reason with llms. `https://openai.com/index/learning-to-reason-with-llms/`. Accessed: 2024-09-19.

Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. 2023. Refiner: Reasoning feedback on intermediate representations. *arXiv preprint arXiv:2304.01904*.

Zhenting Qi, Mingyuan Ma, Jiahang Xu, Li Lyna Zhang, Fan Yang, and Mao Yang. 2024. Mutual reasoning makes smaller llms stronger problem-solvers. *arXiv preprint arXiv:2408.06195*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2024. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, YK Li, Yu Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Noah Shinn, Federico Cassano, Edward Berman, Karthik Narasimhan Ashwin Gopinath, and Shunyu Yao. 2024. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*.

Zhiqing Sun, Longhui Yu, Yikang Shen, Weiyang Liu, Yiming Yang, Sean Welleck, and Chuang Gan. 2024. Easy-to-hard generalization: Scalable alignment beyond human supervision. *arXiv preprint arXiv:2403.09472*.

Zhengyang Tang, Xingxing Zhang, Benyou Wan, and Furu Wei. 2024. Mathscale: Scaling instruction tuning for mathematical reasoning. *arXiv preprint arXiv:2403.02884*.

Ye Tian, Baolin Peng, Linfeng Song, Lifeng Jin, Dian Yu, Haitao Mi, and Dong Yu. 2024. Toward self-improvement of llms via imagination, searching, and criticizing. *arXiv preprint arXiv:2404.12253*.

Shubham Toshniwal, Ivan Moshkov, Sean Narenthiran, Daria Gitman, Fei Jia, and Igor Gitman. 2024. Openmathinstruct-1: A 1.8 million math instruction tuning dataset. *arXiv preprint arXiv:2402.10176*.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*.

Peiyi Wang, Lei Li, Zhihong Shao, R.X. Xu, Damai Dai, Yifei Li, Deli Chen, Y.Wu, and Zhifang Sui. 2023. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *arXiv preprint arXiv:2312.08935*.

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

Stephen Warshall. 1962. A theorem on boolean matrices. *J. ACM*, 9(1):11–12.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. 2022. Large language models are better reasoners with self-verification. *arXiv preprint arXiv:2212.09561*.

Yuxi Xie, Anirudh Goyal, Wenyue Zheng, Min-Yen Kan, Timothy P. Lillicrap, Kenji Kawaguchi, and Michael Shieh. 2024. Monte carlo tree search boosts reasoning via iterative preference learning. *arXiv preprint arXiv:2405.00451*.

Haotian Xu. 2023. No train still gain. unleash mathematical reasoning of large language models with monte carlo tree search guided by energy function. *arXiv preprint arXiv:2309.03224*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. 2024a. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. 2024b. Qwen2 technical report. *Preprint*, arXiv:2407.10671.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. 2024c. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

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

Fei Yu, Anningzhe Gao, and Benyou Wang. 2023. Outcome-supervised verifiers for planning in mathematical reasoning. *arXiv preprint arXiv:2311.09724*.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. 2024a. Generative verifiers: Reward modeling as next-token prediction. *arXiv preprint arXiv:2408.15240*.

Yifan Zhang, Yang Yuan, and Andrew Chi-Chih Yao. 2024b. On the diagram of thought. *arXiv preprint arXiv:2409.10038*.

A Hyper-paramerer settings

All hyper-parameter settings for Section 3 are listed in Table A1.

Hyper-parameter	Description	Value
α	Balance coefficient between local and global	0.9
c	Exploration coefficient in UCT function	1.4
γ	Discount factor for backpropagation	0.1
max_child	Dynamic pruning hyper-parameter in search	2

Table A1: Hyper-parameter settings

B Details of Grading and Metrics

Grading. We follow the correctness evaluation method proposed in PRM800K (Lightman et al., 2023) to score the answers generated by the algorithm. For the mathematical solutions proposed by the algorithm and their corresponding ground truth, we inform the model of the expected response format in a prompt. The answer's formula or value is extracted by matching the response format with predefined rules. If the model fails to follow the expected format in the prompt and the rule-based extraction fails, the solution is directly judged as inconsistent with ground truth.

For the extracted label, we score the answer based on the following criteria. The answer is considered consistent with the ground truth label and passes the evaluation if at least one of the criteria is met:

1. The answer label string is exactly equal to the ground truth label string in terms of literal value.

2. Both the answer label and the ground truth label can be converted to floating-point numbers, and the difference between the two values is less than 1×10^{-6}.

3. Following the criterion proposed in PRM800K (Lightman et al., 2023), we use the Sympy library (Meurer et al., 2017) to simplify the difference between the expression corresponding to the answer label, denoted as a, and the expression corresponding to the ground truth label, denoted as b. If the simplification yields $a - b \iff 0$, the criterion is satisfied.

Metrics. To provide a comprehensive and robust evaluation metric, we adopt **major@k** and **rm@k** as the evaluation metrics.

- **rm@k** represents reward model best-of-N among k sampled response (Yang et al., 2024c).

- **major@k** (Kuncheva, 2014), also abbreviated as **maj@k**, is defined as the fraction of tasks where a majority of the top k samples generated return the same correct solution. This metric focuses on consistency across multiple generated answers. The majority weight is calculated on the solution's extracted labels.

- The **solved rate of problems** (Lightman et al., 2023; Luo et al., 2024a) refers to the percentage of problems who have solutions meet the evaluation criteria.

For results without a subscript, the score represents the greedy evaluation using the default prompt of the base model. Results with a notation indicate the use of the corresponding prompt engineering technique, and those marked with **self-consistency** use the self-consistency aggregation method.

For closed-source models, we report the scores from existing official technical reports (Anthropic, 2024; Reid et al., 2024; OpenAI, 2024) or dataset-provided results, with no modifications.

C Details of PPRM Trainging

C.1 Overview

The design goal of the PPRM is to integrate the properties of both PRM and ORM, providing a more nuanced preference prediction between any two solution answers. We attempt to utilize reinforcement learning methods for training, leveraging the model's instruction-following capability to predict the relative merits of pairs of problem-solving answers. This will further enable the use of the EBC method to evaluate the global quantile scores of mathematical solutions.

C.2 Data Collection

Our data synthesis is derived from two datasets: PRM800K (Lightman et al., 2023) and OpenMathInstruct-1 (Toshniwal et al., 2024). The PRM800K dataset, collected from the MATH dataset, comprises a substantial number of step-divided problem-solving answers, with manual quality annotations for each step. We primarily utilize this dataset to generate pairs of answers for comparative analysis based on step-wise process quality. The OpenMathInstruct-1 dataset

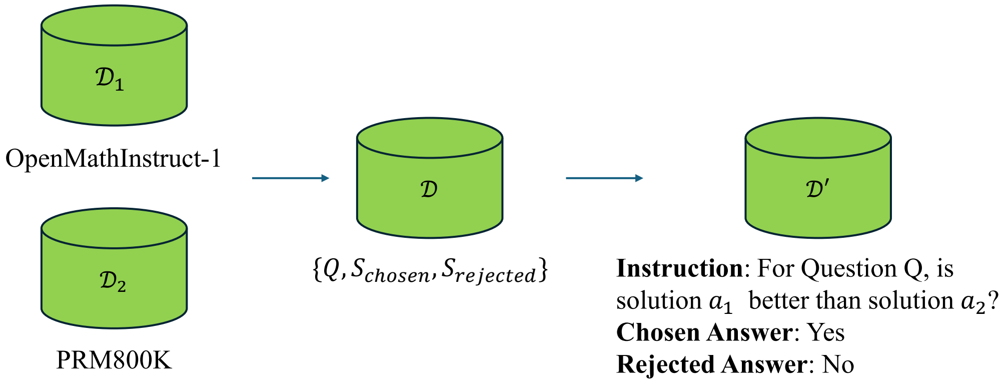


Figure A1: Dataset Construction of PPRM.

incorporates data from the GSM8K and MATH datasets, which have been manually annotated for outcome correctness. We use this dataset to synthesize pairs for comparative analysis based on outcome quality.

In processing PRM800K, for a given problem, we first sample steps of varying quality annotations from the step-wise dataset to construct a complete reasoning path. For pairs with the same final step annotation, paths composed of higher-quality steps are considered superior to those with lower-quality steps. In cases where the final step annotations are not identical, the reasoning path with the superior final step annotation is regarded as better.

During the processing of OpenMathInstruct-1, we exclusively utilize samples without Python interpreter calls. For the same problem, we sample pairs composed of outcomes with higher and lower quality annotations.

Notably, we filtered out any data samples from PRM800K and OpenMathInstruct-1 that may overlap with the GSM8K and MATH test sets, especially the PRM800K. Ultimately, we formed a dataset of 7,780,951 entries for training the PPRM model.

C.3 Direct Preference Pair Construction

For all pairs, we frame the inquiry as "For Question Q, is solution a_1 better than solution a_2?" If solution a_1 is deemed superior to solution a_2, we label it as 'Yes'; otherwise, it is labeled as 'No'.

In this manner, we transform the ordinal relationship prediction task into a question-answer format, employing the Direct Preference Optimization (DPO) method for model training through reinforcement learning from human feedback (RLHF). This approach aims to enhance the model's capability to follow instructions in predicting the relative merits of pairs of problem-solving answers.

C.4 RLHF Training

We apply the DPO method to train the Gemma2-2B-it model using RLHF which is shown in Figure A2. The loss function for DPO is structured as $\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}\left[\log\sigma\left(\beta\log\frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta\log\frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)\right]$, where σ is the logistic function, β is a parameter controlling the deviation from the base reference policy π_{ref}, namely the initial model π^{LLM}. In practice, the language model policy π_θ is also initialized to π^{LLM}. For one answer, denoted as $y_w \succ y_l|x$ where y_w and y_l denote the preferred and dispreferred completion amongst $(\hat{y}, y)$ respectively.

D Details of Berry-Tree Inference Framework

D.1 Overview

`Berry-Tree` is an inference framework specifically designed for complex multi-model reasoning tasks, addressing the need to improve inference efficiency and accuracy in intricate tree search processes of LLM's mathematical reasoning. This framework is particularly suited for large-scale reasoning tasks that involve the integration of multiple models and algorithms. By incorporating advanced tree search methods especially Monte Carlo Tree Search (MCTS), robust concurrency handling, and high-performance inference engines, `Berry-Tree` significantly enhances inference speed and resource utilization of LLM's mathematical reasoning process. This section provides a explanation of the system architecture and key tech-

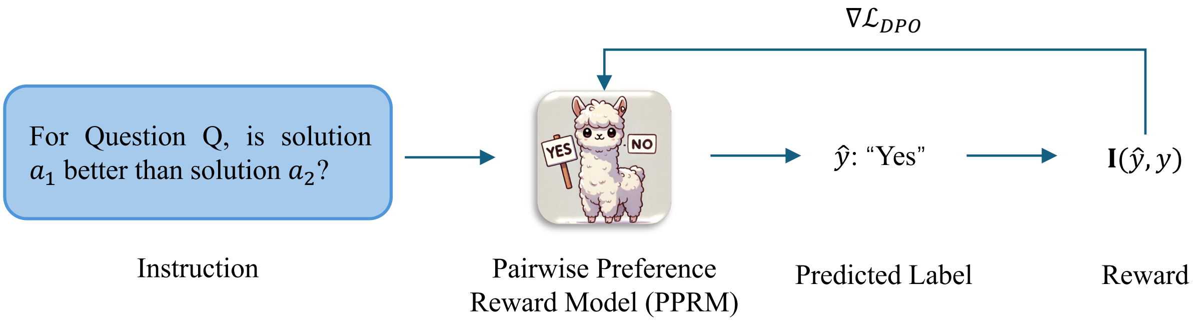

$\nabla\mathcal{L}_{DPO}$
For Question Q, is solution a_1 better than solution a_2?
YES
NO
$\hat{y}$: "Yes"
$\mathbf{I}(\hat{y}, y)$
Instruction
Pairwise Preference Reward Model (PPRM)
Predicted Label
Reward

Figure A2: RLHF training of PPRM.

nologies of Berry-Tree, along with a preliminary performance evaluation results.

D.2 System Architecture Overview

Figure A3 demonstrates the architecture of Berry-Tree which is divided into several layers, each handling different functional requirements. The **Data Management Layer** is responsible for the serialization and deserialization of data, ensuring efficient data read/write operations across models and systems and the ablity of recovering the search process from serialized data. The **Tree Search Methods Layer** incorporates MCTS (Monte Carlo Tree Search), ToT (Tree of Thoughts), and A* algorithms to optimize the inference process and explore multiple reasoning paths. Additionally, Berry-Tree includes a **Reliability Layer**, which ensures load balancing and failover support in highly concurrent scenarios, guaranteeing the stability of inference tasks. Finally, the **Inference Engines Layer** integrates efficient inference engines such as vLLM, FastAPI, and HuggingFace TGI to enable parallelized and efficient task processing.

D.3 Key Technical Components

Data Management. Berry-Tree employs serialization and deserialization techniques, specifically using formats including JSON and CSV, to efficiently store, transfer and recover checkpoint data from tree search reasoning processes. The framework stores this data along with hash values to ensure integrity and allows for quick restoration of tree search states in memory when needed. Furthermore, Berry-Tree leverages the HuggingFace Datasets library to handle core dataset inputs for both training and inference. It supports seamless loading of benchmark datasets such as GSM8K and MATH from the HuggingFace Hub, enhancing its flexibility and ensuring compatibility with diverse reasoning tasks.

Tree Search Methods. Berry-Tree support multiple tree search algorithms, with MCTS being central to handling large-scale complex reasoning tasks by leveraging random simulation and statistical analysis to optimize the search space. The Tree of Thoughts (ToT) extends the exploration depth and breadth of reasoning paths, helping the system manage uncertainty. And A* can offer a efficient heuristic search capabilities.

Reliability Design. To ensure the stability and continuity of inference tasks, Berry-Tree incorporates load balancing and failover mechanisms. During high-concurrency operations, the load balance componet distributes workloads across different inference servers, preventing server overload. The failover mechanism ensures that tasks can seamlessly recover and transition to backup servers in case of partial server failures.

Server Architecture. The framework's server architecture is divided into two segments: one dedicated to executing Large Language Model (LLM) inference and the other designated explicitly for handling PPRM (Pairwise Preference Reward Model) with EBC (Enhanced Borda Count) method. This modular design allows the framework to allocate computational resources flexibly, improving overall efficiency.

Inference Engines. Inference engines of Berry-Tree include VLLM , HuggingFace Transformers warpped by FastAPI, and HuggingFace TGI. These engines collectively enable the system to maintain high efficiency and stability while handling multi-model inference tasks, with robust support for high-concurrency demands.

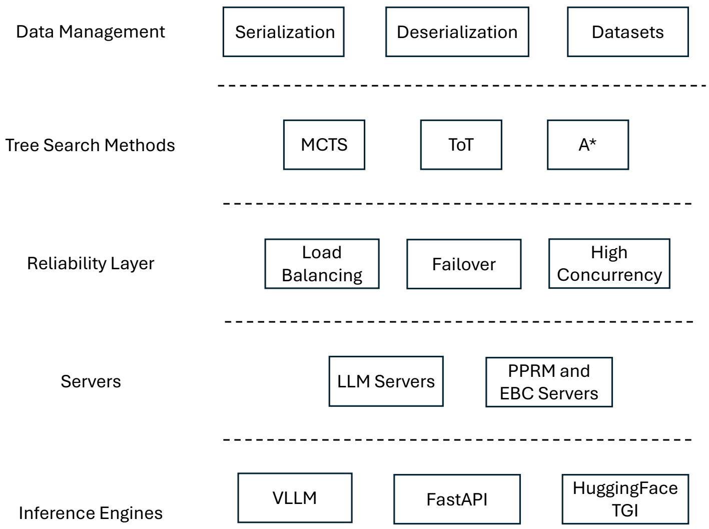


Figure A3: Architecture design of Berry-Tree

D.4 Preliminary Performance Evaluation

In a preliminary performance evaluation, we utilize 16 A100 GPUs to run the LLaMA3.1-8B-instruct model for large-scale inference tasks, while 4 A100 GPUs are used to run the Gemma2-2B-it model as PPRM servers. The test dataset consists of 1319 GSM8K test samples.

We conduct 16 rollouts to parallelize the inference tasks of LLaMA-Berry via Berry-Tree. The results indicate that the total inference time is 1 hour and 25 minutes, with an average inference time of approximately 3.87 seconds per sample. These results demonstrate a strong parallel inference capabilities of Berry-Tree under the given hardware configuration.

E Scaling Study on Inference-time Token Overheads

Comparing LLaMA-3-8B-Instruct and LLaMA-3.1-8B-Instruct across GSM8K, GSMHARD, and MATH500, LLaMA-3.1-8B-Instruct consistently consumes more tokens across all categories—Solutions, Critiques, and Overall. This Figure A4 result reflects LLaMA-3.1-8B-Instruct's tendency to generate more detailed and comprehensive outputs with increased overhead. While this likely improves solution quality, it also introduces greater resource demands and variability in token usage, highlighting a trade-off between accuracy and computational resources. LLaMA-3.1-8B-Instruct is thus better suited for tasks prioritizing precision over speed. As shown in Figure A5, Token overheads during inference also scale with task difficulty across different olympiad-level benchmarks with LLaMA-3.1-8B-Instruct. AIME2024 exhibits the highest token consumption with significant variability, reflecting the complexity of its solution paths. In contrast, GPQA Diamond shows lower overall overhead, while AMC2023 falls in between, with moderate token consumption and less variability than AIME2024 but still notable.

F Future Work

LLaMA-Berry holds significant potential for further development. First, we plan to enhance its multimodal capabilities, enabling it to better handle complex problems like VQA and mathematic geometric problems that require visual, auditory, or even tactile perception. For instance, in tasks involving visual reasoning, such as geometric problems, LLaMA-Berry could be extended to integrate image recognition and analysis capabilities, thereby assisting in solving challenges related to spatial relationships and shape recognition. Furthermore, we aim to generalize its application across other scientific domains, improving its performance in disciplines such as physics, chemistry, and biology. In physics, for example, it could be utilized to address complex micro- and macroscopic dynamics problems; in chemistry, it may assist in molecular structure prediction and drug design; and in biology, it could potentially be used for genome analysis and disease prediction. Moreover, LLaMA-Berry can also offer technical support for other interdisciplinary fields, such as meteorology, environmental science, and

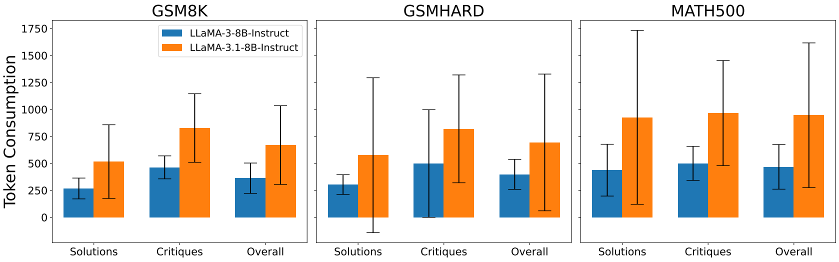


Figure A4: Average token consumption comparison across datasets, error bar stands for standard deviation.

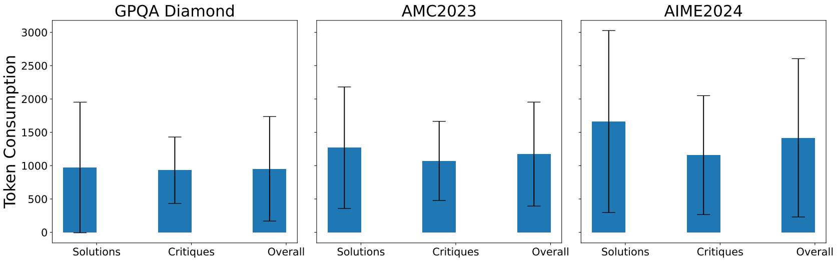


Figure A5: Average token consumption for LLaMA-3.1-8B-Instruct across olympiad-level datasets, error bar stands for standard deviation.

materials science, by integrating multimodal data to enhance predictive accuracy. At the same time, we will focus on how LLaMA-Berry can further enhance AI safety. For instance, LLaMA-Berry could be leveraged to design more robust safety and risk assessment mechanisms. Besides, the generation of responses could be guided by a safety-trained PPRM to produce outputs with higher safety standards. These advancements not only pave the way for broader scientific applications of LLaMA-Berry but also offer new possibilities for enhancing AI safety and promoting its widespread adoption.

G Case Study

The prompts utilized in LLaMA-Berry for the LLaMA-3.1-8B-Instruct model are presented in Figure A6. Additionally, Figure A7 provides a detailed breakdown of problem-solving examples derived from the GSM8K dataset.

H Convergence Analysis of the Enhanced Borda Count (EBC) Method

In this appendix, we present a formal discussion on how the quantile scores, as evaluated by the Enhanced Borda Count (EBC) method, converge to the true quantile scores of solutions within the actual quality distribution as the number of samples increases.

H.1 Definitions and Assumptions

Solution Set. Let $A = \{a_1, a_2, \ldots, a_n\}$ be a finite set of n solutions (answers).

True Quantile Scores. Each solution a_i has a true quality score $Q^*(a_i) \in \mathbb{R}$, drawn from a continuous distribution P_Q.

True Ranking. The true ranking R^* is induced by ordering the solutions in decreasing order of their true quantile scores $Q^*(a_i)$.

True Pairwise Preference Probabilities. The true probability that solution a_i is preferred over a_j is defined as:

$$P^*(a_i \succ a_j) = \mathbb{P}(Q^*(a_i) > Q^*(a_j)) + \tfrac{1}{2}\mathbb{P}(Q^*(a_i) = Q^*(a_j)). \tag{A1}$$

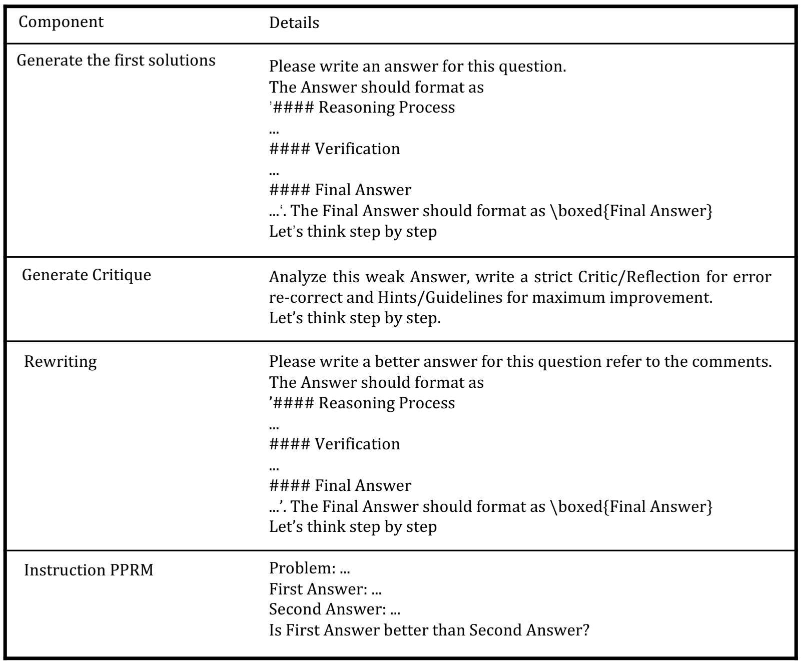
Given the continuity of P_Q, we have $\mathbb{P}(Q^*(a_i) = Q^*(a_j)) = 0$, so:

$$P^*(a_i \succ a_j) = \begin{cases} 1, & \text{if } Q^*(a_i) > Q^*(a_j), \\ 0, & \text{if } Q^*(a_i) < Q^*(a_j). \end{cases} \tag{A2}$$

Estimated Preference Probabilities. For T independent samples, the estimated preference probability is:

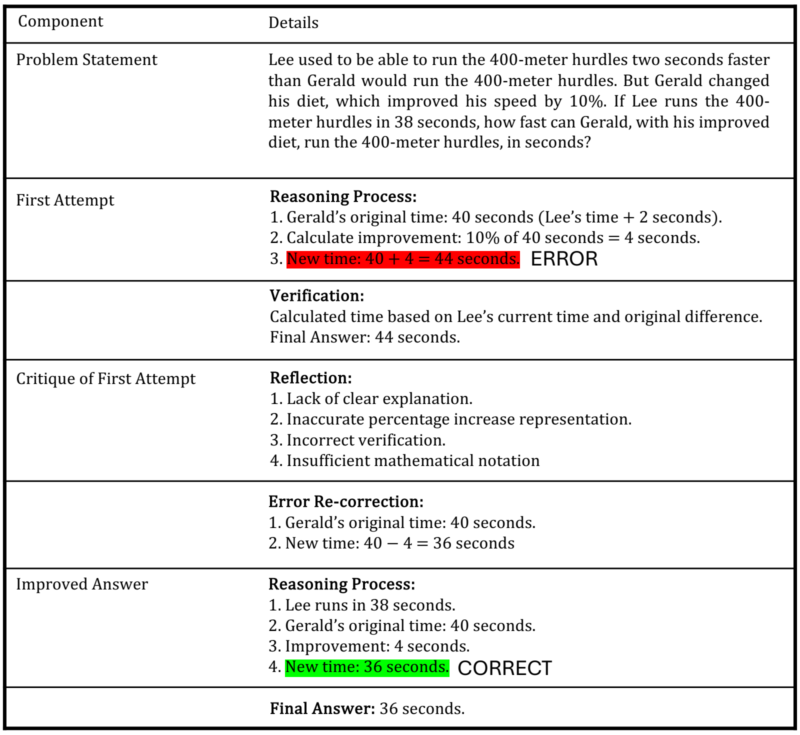
$$P_T(a_i \succ a_j) = \frac{1}{T}\sum_{t=1}^{T} X_t^{(i,j)}, \tag{A3}$$

where $X_t^{(i,j)}$ are independent Bernoulli random variables with success probability $P^*(a_i \succ a_j)$.



Component	Details
Generate the first solutions	Please write an answer for this question. The Answer should format as '#### Reasoning Process ... #### Verification ... #### Final Answer ...'. The Final Answer should format as \boxed{Final Answer} Let's think step by step
Generate Critique	Analyze this weak Answer, write a strict Critic/Reflection for error re-correct and Hints/Guidelines for maximum improvement. Let’s think step by step.
Rewriting	Please write a better answer for this question refer to the comments. The Answer should format as ’#### Reasoning Process ... #### Verification ... #### Final Answer ...’. The Final Answer should format as \boxed{Final Answer} Let’s think step by step
Instruction PPRM	Problem: ... First Answer: ... Second Answer: ... Is First Answer better than Second Answer?

Figure A6: Prompts for LLaMA-3.1-8B-Instruct



Component	Details
Problem Statement	Lee used to be able to run the 400-meter hurdles two seconds faster than Gerald would run the 400-meter hurdles. But Gerald changed his diet, which improved his speed by 10%. If Lee runs the 400-meter hurdles in 38 seconds, how fast can Gerald, with his improved diet, run the 400-meter hurdles, in seconds?
First Attempt	**Reasoning Process:** 1. Gerald’s original time: 40 seconds (Lee’s time + 2 seconds). 2. Calculate improvement: 10% of 40 seconds = 4 seconds. 3. New time: 40 + 4 = 44 seconds. ERROR
	Verification: Calculated time based on Lee’s current time and original difference. Final Answer: 44 seconds.
Critique of First Attempt	**Reflection:** 1. Lack of clear explanation. 2. Inaccurate percentage increase representation. 3. Incorrect verification. 4. Insufficient mathematical notation
	Error Re-correction: 1. Gerald’s original time: 40 seconds. 2. New time: 40 − 4 = 36 seconds
Improved Answer	**Reasoning Process:** 1. Lee runs in 38 seconds. 2. Gerald’s original time: 40 seconds. 3. Improvement: 4 seconds. 4. New time: 36 seconds. CORRECT
	Final Answer: 36 seconds.

Figure A7: Problem-solving example

Convergence Assumption. As $T \to \infty$, $P_T(a_i \succ a_j)$ converges almost surely to $P^*(a_i \succ a_j)$.

Preference Matrix. Construct the estimated preference matrix M_T and the true preference matrix M^* as:

$$
\begin{aligned}
M_T[i,j] &= \begin{cases} 1, & \text{if } P_T(a_i \succ a_j) \geq 0.5, \\ 0, & \text{otherwise.} \end{cases} \\
M^*[i,j] &= \begin{cases} 1, & \text{if } P^*(a_i \succ a_j) = 1, \\ 0, & \text{if } P^*(a_i \succ a_j) = 0. \end{cases}
\end{aligned}
\tag{A4}
$$

Transitive Closure. Let C_T and C^* be the transitive closures of M_T and M^*, respectively.

Borda Counts. Compute the Borda count for each solution:

$$B_T(a_i) = \sum_{j \neq i} C_T[i,j], \quad B^*(a_i) = \sum_{j \neq i} C^*[i,j]. \tag{A5}$$

Estimated Ranking and Quantile Scores.

- Estimated ranking R_T: Order solutions by decreasing $B_T(a_i)$.
- Estimated quantile score:

$$Q_g(a_i) = 1 - \frac{\text{rank}_T(a_i) - 1}{n-1}. \tag{A6}$$

- True quantile score:

$$Q_g^*(a_i) = 1 - \frac{\text{rank}^*(a_i) - 1}{n-1}. \tag{A7}$$

H.2 Objective

To prove that as $T \to \infty$, the estimated quantile scores $Q_g(a_i)$ converge almost surely to the true quantile scores $Q_g^*(a_i)$:

$$\lim_{T \to \infty} Q_g(a_i) = Q_g^*(a_i), \quad \forall a_i \in A. \tag{A8}$$

H.3 Proof

Convergence of Estimated Preference Probabilities.

By the Strong Law of Large Numbers (SLLN), since $X_t^{(i,j)}$ are i.i.d. Bernoulli random variables with success probability $P^*(a_i \succ a_j)$, we have:

$$\lim_{T \to \infty} P_T(a_i \succ a_j) = P^*(a_i \succ a_j) \quad \text{almost surely.} \tag{A9}$$

Convergence of Preference Matrix M_T to M^*.

Since $P_T(a_i \succ a_j)$ converges to $P^*(a_i \succ a_j)$ and $P^*(a_i \succ a_j) \in \{0, 1\}$, for sufficiently large T, we have:

$$M_T[i,j] = M^*[i,j], \quad \forall i \neq j, \tag{A10}$$

almost surely.

Justification: Because $P_T(a_i \succ a_j)$ converges to either 0 or 1, and $P_T(a_i \succ a_j) \neq 0.5$ almost surely for large T.

Convergence of Transitive Closure C_T to C^*.

The transitive closure is a deterministic function of the preference matrix. Therefore, since $M_T \to M^*$, it follows that:

$$C_T \to C^* \quad \text{as } T \to \infty, \tag{A11}$$

almost surely.

Convergence of Borda Counts $B_T(a_i)$ to $B^*(a_i)$.

Given that $C_T \to C^*$, the Borda counts converge:

$$B_T(a_i) = \sum_{j \neq i} C_T[i,j] \to B^*(a_i) = \sum_{j \neq i} C^*[i,j], \tag{A12}$$

almost surely.

Convergence of Estimated Ranking R_T to True Ranking R^*.

Since the Borda counts $B_T(a_i)$ converge to $B^*(a_i)$, and assuming that all $Q^*(a_i)$ are distinct (due to the continuity of P_Q), the rankings induced by $B_T(a_i)$ converge to the true rankings:

$$R_T \to R^* \quad \text{as } T \to \infty, \tag{A13}$$

almost surely.

Convergence of Quantile Scores $Q_g(a_i)$ to $Q_g^*(a_i)$.

Since $\text{rank}_T(a_i) \to \text{rank}^*(a_i)$, we have:

$$\begin{aligned} Q_g(a_i) = 1 - \frac{\text{rank}_T(a_i) - 1}{n-1} \to \\ Q_g^*(a_i) = 1 - \frac{\text{rank}^*(a_i) - 1}{n-1} \end{aligned} \tag{A14}$$

almost surely.

Conclusion.

Therefore, we have shown that:

$$\lim_{T \to \infty} Q_g(a_i) = Q_g^*(a_i), \quad \forall a_i \in A, \tag{A15}$$

which means that the EBC method's estimated quantile scores converge almost surely to the true quantile scores of the solutions.

H.4 Finite Sample Analysis and Convergence Rate

Lemma 1: Hoeffding's Inequality for Preference Probability Estimates.

For each pair (a_i, a_j), $P_T(a_i \succ a_j)$ is the sample mean of T i.i.d. Bernoulli trials with success probability $P^*(a_i \succ a_j)$. By Hoeffding's inequality:

$$\begin{aligned} \mathbb{P}\left(|P_T(a_i \succ a_j) - P^*(a_i \succ a_j)| \geq \epsilon\right) \\ \leq 2\exp(-2T\epsilon^2). \end{aligned} \tag{A16}$$

Lemma 2: Uniform Convergence over All Pairs.

Apply the union bound over all $N = n(n-1)/2$ pairs:

$$\begin{aligned} \mathbb{P}\left(\exists (i,j) : |P_T(a_i \succ a_j) - P^*(a_i \succ a_j)| \geq \epsilon\right) \\ \leq N \cdot 2\exp(-2T\epsilon^2). \end{aligned} \tag{A17}$$

Set the right-hand side equal to δ to solve for T:

$$T \geq \frac{1}{2\epsilon^2} \ln \left(\frac{2N}{\delta} \right). \tag{A18}$$

Lemma 3: Correctness of Preference Matrix with High Probability.

Given that $P^*(a_i \succ a_j) \in \{0, 1\}$, for any $0 < \epsilon < 0.5$, if:

$$|P_T(a_i \succ a_j) - P^*(a_i \succ a_j)| < \epsilon, \tag{A19}$$

then $M_T[i,j] = M^*[i,j]$ because $P_T(a_i \succ a_j)$ will be greater than 0.5 when $P^*(a_i \succ a_j) = 1$, and less than 0.5 when $P^*(a_i \succ a_j) = 0$.

Lemma 4: Probability of Correct Ranking.

From Lemma 2 and 3, with probability at least $1 - \delta$, $M_T = M^*$, and thus $C_T = C^*$, leading to $R_T = R^*$ and $Q_g(a_i) = Q_g^*(a_i)$.

Convergence Rate Analysis.

To achieve this with confidence level $1 - \delta$, the required number of samples per pair is:

$$T \geq \frac{1}{2\epsilon^2} \ln \left(\frac{n(n-1)}{\delta} \right). \tag{A20}$$

For small δ and $\epsilon < 0.5$, T scales logarithmically with the number of solutions n.

H.5 Addressing Practical Considerations

In practice, the true preference probabilities may not be exactly 0 or 1 due to noise or overlapping quality scores. To accommodate this:

- **Extended Preference Model**: Assume $P^*(a_i \succ a_j)$ is a strictly increasing function of $\Delta Q^*_{ij} = Q^*(a_i) - Q^*(a_j)$, such as:

  $$P^*(a_i \succ a_j) = F(\Delta Q^*_{ij}), \tag{A21}$$

  where F is a cumulative distribution function (CDF).

- **Margin Condition**: Define a margin $m > 0$ such that for all $i \neq j$:

  $$|P^*(a_i \succ a_j) - 0.5| \geq m. \tag{A22}$$

  This ensures a minimum separation between preference probabilities.

- **Modified Sample Complexity**: With the margin condition, Hoeffding's inequality becomes:

  $$\mathbb{P}\left(M_T[i,j] \neq M^*[i,j]\right) \leq 2\exp(-2Tm^2). \tag{A23}$$

  To achieve $\mathbb{P}\left(M_T = M^*\right) \geq 1 - \delta$, we need:

  $$T \geq \frac{1}{2m^2} \ln \left(\frac{n(n-1)}{\delta} \right). \tag{A24}$$

- **Convergence under Noise**: Even with noisy preference probabilities, as long as there is a margin $m > 0$, the convergence of $Q_g(a_i)$ to $Q_g^*(a_i)$ still holds with high probability for sufficiently large T.

H.6 Final Conclusion

We have provided a formal discussion showing that the estimated quantile scores $Q_g(a_i)$, obtained through the EBC method, converge almost surely to the true quantile scores $Q_g^*(a_i)$ as the number of samples T approaches infinity. The finite sample analysis demonstrates that the convergence rate depends logarithmically on the number of solutions and is inversely proportional to the square of the margin m between preference probabilities.

I Pseudo-code of main Algorithms

We present the process of Self-Refine applied to Monte Carlo Tree Search (SR-MCTS) Method in Algorithm 1 and provide overall pseudo-code for Ehanced Borda Count (EBC) Method in Algorithm 2.

Algorithm 1: Self-Refine applied to Monte Carlo Tree Search (SR-MCTS) Method

```
Input: Initial state $s_0$, search tree $\mathcal{T}$, max nodes $N_{max}$, exploration constant $c$
Output: Ranked solution list $S$
Initialize search tree $\mathcal{T}$ with root node $s_0$;
while number of nodes $N(\mathcal{T}) < N_{max}$ do
    ———Selection Phase———
    Select a node $s_i$ which met the dynamic pruning rule from $\mathcal{T}$ using UCT:
        $$a = \arg\max_{a \in A(s)} \left( Q(s,a) + c \cdot \sqrt{\frac{\ln N(s)}{N(s,a)}} \right)$$
    ———Expansion Phase———
    Expand $s_i$ by generating a successor node $s'$ using the rewriting process $R(s_i, c_i)$, where
      $c_i = C(s_i)$ is a critique of the current state;
    Add the new node $s'$ to $\mathcal{T}$;
    ———Evaluation Phase———
    Compute the value $Q(s')$ of the new node with Enhanced Borda Count (EBC) method:
        $$Q(s') = \alpha Q_g(s') + (1 - \alpha) Q_l(s')$$
      where $Q_g(s')$ is the global value from the win-loss matrix $M$ and $Q_l(s')$ is the local value
      from adjacent nodes in $\mathcal{T}$;
    ———Backpropagation Phase———
    Propagate $Q(s')$ back to its parent node $s_i$, updating $s_i$'s Q value:
        $$Q(s_i) = (1 - \gamma) Q(s_i) + \gamma Q(s')$$
    ———Check for tree growth limit———
    if $N(\mathcal{T}) \geq N_{max}$ then
        break;
    end
end
return Ranked solution list $S$
```

Algorithm 2: Ehanced Borda Count (EBC) Method

```
Data: $M$ (Binary Reward Matrix),$P$ (Pairwise Preference Reward model)
Result: $Q$ (quantile rewards), $R$ (ranked node list), $L$ (ranked layers of nodes)
Function FillTransitiveClosure($M$):
    $C \leftarrow$ all-zero matrix from size of $M$
    $C[i, j] \leftarrow -1$ for all $i, j$
    $C[i, j] \leftarrow$ sign$(M[i, j] - 0.5)$ if $M[i, j] \neq -1$ else $-1$
    for $k = 0$ to $|C| - 1$ do
        for $i = 0$ to $|C| - 1$ do
            for $j = 0$ to $|C| - 1$ do
                if $C[i, k] = C[k, j]$ then
                    $C[i, j] \leftarrow C[i, k]$
                end
            end
        end
    end
    return Updated $C$
Function BordaCount($C$):
    $D \leftarrow$ outdegree of each node from $C$
    $R \leftarrow$ argsort$(-D)$
    $L \leftarrow$ Define layers using unique values in $D$
    return $R$ , $L$
Function Rerank($R$, $L$, $C$, $P$):
    $G \leftarrow$ Group $R$ by $L$
    $R \leftarrow$ Sort each group in $G$ using local comparisons computed by $P$
    $L, C \leftarrow$ Update $L, C$ by $R$
    return $R$, $L$, Updated $C$
Function CalculateQuantileScores($R$, $L$):
    $Q \leftarrow \emptyset$
    $S_L \leftarrow \{1 - \frac{l}{\max(L)} : l \in L\}$
    $Q[x] \leftarrow S_L$[layer of $x$] for each $x \in R$
    return $Q$
Function EnhancedBordaCount($M$, $q$):
    $C \leftarrow$ FillTransitiveClosure($M$)
    $R$, $L \leftarrow$ BordaCount($C$)
    $R$, $L$, $C \leftarrow$ Rerank($R$, $L$, $C$,$P$)
    $Q \leftarrow$ CalculateQuantileScores($R$, $L$)
    return $Q$, $R$, $L$
```