Atom Reasoner: LLM Self-Preference Training via Monte Carlo Tree Search

Anonymous EMNLP submission

Abstract

Recent research focuses on utilizing more test-time computation to enhance the performance of Large Language Models in solving complex mathematical and logical reasoning tasks. However, these methods allocate more computational resources during the inference phase to explore the solution space by some tree search method such as Monte Carlo Tree Search, resulting in a significant increase in inference time. In this paper, We construct atom reasoning steps, which are subsequently utilized to develop MCTS for self-preference learning to enhance the reasoning capabilities of LLMs, without employing a larger model for data distillation. Extensive evaluations on various mathematic and common sense reasoning tasks demonstrate demonstrate remarkable performance improvements over existing models. For instance, our approach outperforms the Qwen2.5-7B-instruct baseline on MATH, GSM8K, ARC and SCIQ with substantial increases in accuracy to 50.0% (+14.2%), 92.1% (+10.4%), 89.6% (+13.3%), and 94.0 (+19.8%) respectively.

1 Introduction

Large language models (LLMs) have demonstrated impressive capabilities in a wide range of tasks and fields (OpenAI et al., 2024). Developing AI systems that can mimic human-like reasoning continues to be a central objective in the research community. A critical aspect of this process involves the collection of reasoning chains that reflect human-like thinking (Madaan et al., 2023). There are two dominant approaches for integrating these data: The first method generates reasoning chains by constructing linear data structures, breaking through solutions into step-by-step reasoning paths (Lightman et al., 2023; Luo et al., 2024), while the other method combines non-linear data structures, such as tree structures, to search for reasoning trajectories integrating the Monte Carlo Tree Search (MCTS) method (Qi et al., 2024). Current research often combines MCTS for preference data collection to train LLMs to enhance the model's ability for complex reasoning (Zhang et al., 2024b).

A typical approach is the chain-of-thought (CoT), which expands the reasoning space by generating additional tokens or solutions (Wei et al., 2022; Wang et al., 2023). Although straightforward and intuitive, recent studies have noted that the CoT method can frequently miss optimal reasoning pathways and display an automatic response style due to its emphasis on a single pathway (Besta et al., 2024). A striking example of the effectiveness of this iterative approach is AlphaZero (Silver et al., 2017), which demonstrates superhuman performance in multiple domains by integrating the advantages of the self-play system, RL techniques, and MCTS (Kocsis and Szepesvári, 2006; Chung et al., 2005). The success of AlphaZero highlights the possibilities that arise from the integration of these advanced techniques into LLMs.

However, the incorporation of MCTS in the collection of preference data for the enhancement of current policy is complex and requires thoughtful deliberation. One primary challenge lies in constructing human-like reasoning trajectories (Zhang et al., 2024b). Conventionally, reasoning trajectories are structured in a step-by-step strategy, which may lead to insufficient exploration of the action space (Qi et al., 2024). Another challenge is the critic or reward function for each intermediate reasoning step. This function is crucial for providing meaningful feedback on different rollouts generated by MCTS, thus guiding the policy improvement process (Liu et al., 2024a).

To address the issues above, evidence from LLM research indicates the superiority of atom reasoning actions as fundamental components for constructing chain-structured patterns (Wu et al., 2024). Inspired by the preceding conclusion, our approach defines five atom reasoning actions and utilizes

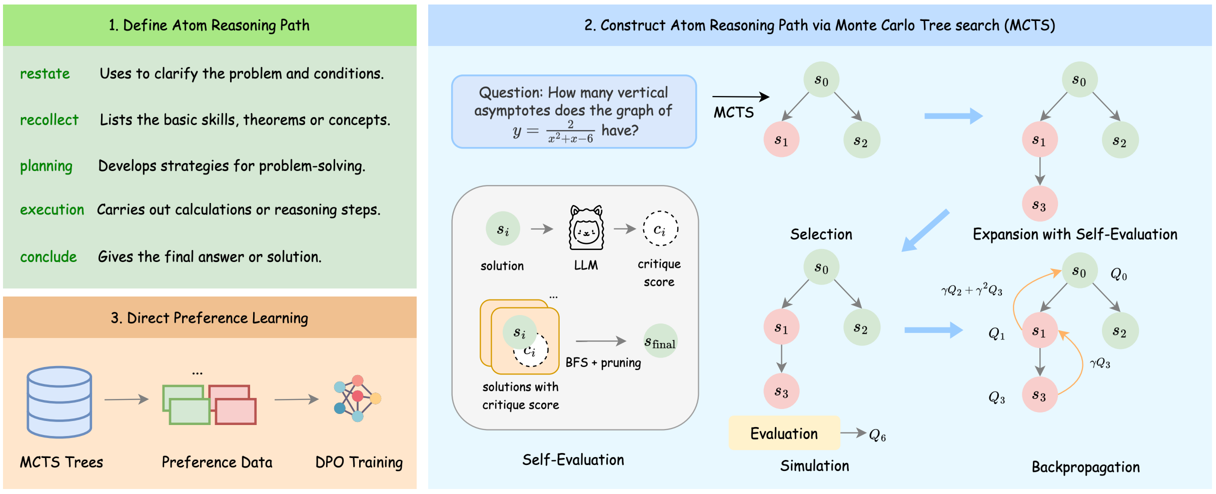
1. Define Atom Reasoning Path
restate
Uses to clarify the problem and conditions.
recollect
Lists the basic skills, theorems or concepts.
planning
Develops strategies for problem-solving.
execution
Carries out calculations or reasoning steps.
conclude
Gives the final answer or solution.
3. Direct Preference Learning
MCTS Trees
Preference Data
DPO Training
2. Construct Atom Reasoning Path via Monte Carlo Tree search (MCTS)
Question: How many vertical asymptotes does the graph of $y = \frac{2}{x^2+x-6}$ have?
MCTS
s_0
s_1
s_2
s_3
Selection
Expansion with Self-Evaluation
s_i
solution
LLM
c_i
critique score
solutions with critique score
BFS + pruning
s_{final}
Self-Evaluation
Evaluation
Q_6
Simulation
$\gamma Q_2 + \gamma^2 Q_3$
Q_0
Q_1
γQ_3
Q_3
Backpropagation

Figure 1: Monte Carlo Tree Search (MCTS) improves model performance through Direct Preference Learning (DPO). Our framework consists of two stages after the atom reasoning steps have been defined: MCTS constructs atom reasoning paths as preference data, and then performs DPO to enhance the model.

MCTS rollouts to collect preference data for Direct Preference Optimization (DPO) (Rafailov et al., 2024). Moreover, we implement self-evaluation, allowing the model to assess its own outputs, which creates a more efficient policy improvement pipeline by functioning as both the policy and the critic (Kadavath et al., 2022; Xie et al., 2023).

In summary, we present an algorithm based on MCTS that breaks down the entire reasoning path into five different atom reasoning steps. MCTS allows us to thoroughly explore the reasoning space, enabling LLMs to behave in a manner akin to human reasoning. we utilize the LLMs to evaluate the intermediate steps toward solving the problem, and prune the nodes with lower scores. During the training phrase, we select preferred and dispreferred paths according to the average scores assigned to each step during the self-evaluation process, constructing preference data for DPO training.

2 Related Work

2.1 Reasoning with LLMs

LLMs reasoning often requires breaking down complex questions into a series of sequential intermediate steps prior to generate the final answer, as illustrated by Chain-of-Thought (CoT) and its variations (Wei et al., 2022; Kojima et al., 2023). Following this, a variety of prompting techniques have been introduced to enhance the generated rationales (Zhou et al., 2023; Hao et al., 2023). Another group of research converts the linear reasoning structure into a non-linear format, such as a tree or graph, integrating thought evaluation with search algorithms like depth-first search (DFS) (Yao et al., 2024; Long, 2023). To generate more reasonable intermediate processes, LLMs themselves often serve as the evaluator to give feedback to intermediate states (Hao et al., 2023; Yao et al., 2024). Different from our Atom Reasoner, these methods require searching during inference, which significantly increases latency.

2.2 LLM Self-Improving

Reinforcement learning (RL) has been increasingly utilized for large language models (LLMs) by treating them as RL agents to align their outputs with human feedback (Christiano et al., 2023; Ouyang et al., 2022). For instance, reinforced self-training methods introduce mechanisms to curate new high-quality examples and iteratively enrich the training dataset for enhancing model performance (Gulcehre et al., 2023; Wang et al., 2024b). However, these methods generally depend on either an external reward model (Yang et al., 2024b; Aksitov et al., 2023) or labeled datasets (Gulcehre et al., 2023). In contrast, methods such as self-rewarding leverage LLMs themselves to assess the generated content, which aligns more closely with our approach. In contrast, methods such as self-rewarding leverage LLMs themselves to assess the generated content (Chen et al., 2024; Lee et al., 2024), which aligns more closely with our approach.

2.3 Monte Carlo Tree Search for LLMs

Monte Carlo Tree Search (MCTS) is a decision-making algorithm commonly employed in games and intricate decision-making processes (Browne et al., 2012; Chaslot et al., 2021). Recent research indicates that MCTS can improve the decoding process in LLMs by expanding the action space during the inference process.(Liu et al., 2024a; Qi et al., 2024). However, a major challenge associated with MCTS is the increased latency during inference, especially when dealing with complex reasoning tasks (Liu et al., 2023). While certain approaches have tried to enhance LLMs by utilizing reasoning paths discovered via MCTS (Tian et al., 2024; Feng et al., 2024), these methods still depend on manually labeled data to train distinct policy and reward models for investigating and assess potential reasoning step at the leaves of the tree (Jiang et al., 2024). In contrast, our approach eliminates the requirement for human annotations and simplifies the tuning process of LLMs without additional inference burden.

3 Preliminaries

3.1 Monte Carlo Tree Search

To enhance the reasoning capabilities of LLMs, we dissect the reasoning process into discrete steps, each represented by a token sequence of atom reasoning steps. We define s_t as the state at time t, which represents the prefix of the reasoning chain, with the addition of a new reasoning step a transitioning the state to s_{t+1}, where s_{t+1} is the concatenation of s_t and a. Utilizing the model as current policy π_θ, we sample candidate solution steps from its probability distribution $\pi_\theta(a|x, s_t)$. with x being the specific task input prompt. MCTS serves as an approximate policy improvement operator by leveraging the UCT strategy to search through the action space, thereby predicting the optimal state at the next step. The tree-structured search supports a balance between exploring diverse possibilities and exploiting promising paths, essential for navigating the vast search space in LLM reasoning.

The MCTS process begins from a root node s_0, as the query input. The algorithm follows four key steps: selection, expansion, and backup, which we will detail further.

Selection Phase. The goal of this phase is to identify nodes that strike a balance between search quality and computational efficiency. The selection is directed by two primary variables: $Q(s_t, a)$, the value function of taking action a at the state s_t, and $N(s_t)$, the visitation frequency of state s_t. To navigate the trade-off between exploring new nodes and exploiting visited ones, the Upper Confidence Bound applied to Trees (UCT) algorithm (Rosin, 2011) is employed to select the optimal node, with dynamic pruning used to avoid local optima. At node s_t, the choice of the subsequent node follows the formula:

$$a = \underset{a \in A(S)}{\arg\max} \left(Q(s_t, a) + c \cdot \sqrt{\frac{\ln N(s_t)}{N(s_t, a)}} \right) \quad (1)$$

Expansion Phase. Expansion occurs at a leaf node during the selection process to integrate new nodes. If the selected leaf node is not in a terminal state, the node will be expanded into k child node in depth with i as $\{s_{ij} \mid j \in [1, k]\}$ by decoding for one step using the policy model π_θ. Following (Xie et al., 2023), we define self-evaluation score $C(s_t)$ as Eq.2, where prompt$_{\text{eval}}$ denotes the evaluation prompt.

$$C(s_t) = \pi_\theta \left(\text{prompt}_{\text{eval}} \mid x, s_t\right) \quad (2)$$

Backup Phase. Once a terminal state is reached, we perform a bottom-up update that propagates from the terminal node up to the root. The visit count N, the state value V , and the transition value Q will be updated as follows:

$$Q(s_t, a) \leftarrow r(s_t, a) + \gamma V(s_{t+1}) \quad (3)$$

$$V(s_t) \leftarrow \sum_a Q(s_t, a) \quad (4)$$

$$N(s_t) \leftarrow N(s_t) + 1 \quad (5)$$

3.2 Direct Preference Optimization

As a sufficient quantity of preference data has been sampled through MCTS, we employ Direct Preference Optimization (DPO) (Rafailov et al., 2024) for training to update current policy π_θ. DPO is a method designed to directly optimize LLMs to align with preference data collected by human or AI feedback (Liu et al., 2024b; Zeng et al., 2024b). The standard RLHF paradigm trains a reward model (Ouyang et al., 2022) on the preference data and employs PPO (Schulman et al., 2017) to optimize the policy π_θ with the feedback provided by the reward model, where π_θ is also initialized to π_{sft} in practice. DPO avoids fitting a reward model by optimizing the policy π_θ using preferences directly.

After a pair of outcomes (y_1, y_2) are sampled from policy model $\pi(y|x)$ conditioned on input x, which are labeled to be (y_w, y_l) according to some preference density p as $\Pr\left[(y_w, y_l)\right] = p\left(y_1 > y_2 \mid x\right)$. Let the ground-truth reward function be r, then estimate the optimal policy π_θ by fitting the Bradley-Terry model (Bradley and Terry, 1952) on preference data. The DPO objective is formulated as follows, where σ is the logistic function, the hyperparameter β regulates the penalty imposed for the deviations from the base reference model π_{ref}.

$$
\begin{aligned}
&p(y_1 \succ y_2 \mid x) = \sigma\left(r(x, y_1) - r(x, y_2)\right) \\
&= \sigma\left(\beta \log\left(\frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)}\right) - \beta \log\left(\frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)\right)
\end{aligned} \tag{6}
$$

4 Method

In this paper, we introduce an approach for improving LLMs reasoning, centered around a direct preference learning process based on the reasoning paths derived from MCTS. The proposed method consists of two stages: first, generating reasoning steps through MCTS in different action spaces; second, using Q-value filtering of preference data for direct preference learning. To solve a complex problem, we formulate it as a multi-step reasoning generation task, which breaks the reasoning path into atom reasoning steps. In addressing the critical aspects of this methodology, three key challenges emerge: efficient decomposing the reasoning path, effective gathering of preference data and training with the DPO objective.

4.1 Define Atom Reasoning Steps

Current research frequently classifies reasoning into two cognitive processes: System 1 and System 2, thinking in fast and slow (Ji et al., 2025). "System 1" refers to the fast, automatic, and intuitive way of thinking, while "System 2" represents the slower, more deliberate, and analytical mode of thinking (Hagendorff et al., 2023). In light of the impressive capabilities of OpenAI's o1 model in complex reasoning, there is a growing focus among researchers on developing effective "System 2" approaches (Qin et al., 2024). Inspired by this, we introduce five atom reasoning steps to enhance the complex reasoning abilities LLMs. We represent the following steps using symbols z_1 to z_5, corresponding to `Restate`, `Recollect`, `Planning`, `Execution`, and `Conclusion`, respectively. The complete chain-of-thoughts can be expressed as $z = [z_1, \ldots, z_5]$. We define the atom reasoning steps as follows:

- **Restate:** Clarify the problem and the associated conditions, ensuring a clear understanding of what needs to be addressed.
- **Recollect:** List the basic mathematical skills, theorems, or concepts that may be needed to solve the problem.
- **Planning:** Develop steps or strategies to effectively address the problem, which will help identify actionable solutions.
- **Execution:** Carry out specific calculations or solution steps rigorously following the planning results above.
- **Conclusion:** Give the final answer in the format of "The answer is: <ANSWER>.", and will be compared with the gold answer.

4.2 Synthesizing Preference Data

As shown in Figure 1, our method of collecting preference data closely aligns with the MCTS inference process. In particular, the process is divided into three main components: 1) *Thought Generation*, which generate multiple candidate atom reasoning steps; 2) *Self Evaluation*, which evaluate each reasoning step by LLMs and select the top k with the highest scores; 3) *Rollout and collection*, which collect the preference data for direct preference learning.

Thought Generation. We define the partial solution state $s_{i-1} = [x, z_1, \ldots, z_{i-1}]$, where x represents the initial input containing the few-shot examples and the question to be answered, and $[z_1, \ldots, z_{i-1}]$ denotes a sequence of previous atom steps. we sample k candidate solutions for the next reasoning step as shown in Eq.7. Specifically, it follows the format of demonstrations, starting with the prefix like `<restate>` and end with `</restate>`, for instance. When the marker `</conclusion>` is generated, the model pauses its reasoning and the chain-of-thought is complete. As a result, We obtain a set S_i containing k new states as shown in Eq.8.

$$z_{ij} \sim \pi_\theta(z_i \mid x, s_{i-1}), \quad \text{for } j = 1, \ldots, k \tag{7}$$

$$S_i = \{s_{ij} = [x, z_{1j}, \ldots, z_{ij}] \mid j = 1, \ldots, k\} \tag{8}$$

Self Evaluation. Considering the set of candidate solution states $S_i = \{s_{ij}\}_{j=1}^{k}$, we employ LLMs as the generative reward model (GRM) to evaluate each state in each reasoning step, thus eliminating

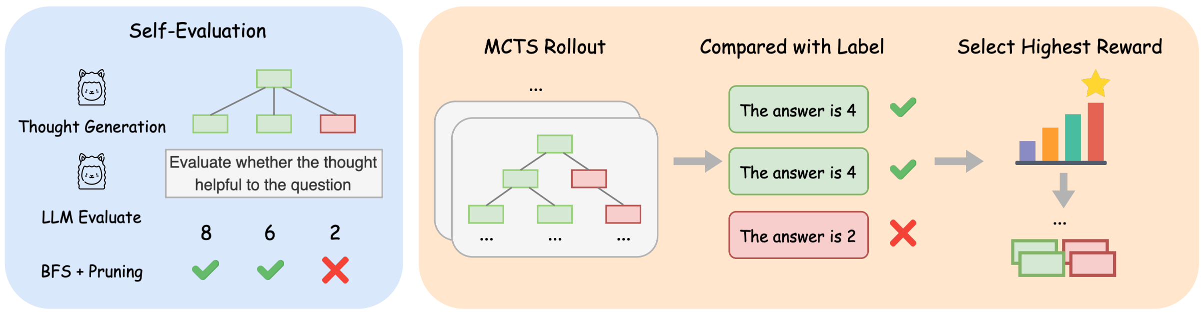

Figure 2: The framework of our method. The left part depicts the self-evaluation process at each reasoning step, while the right part shows the combination of preference data. The green box represents the preference reasoning path, whereas the red box indicates the dispreference one.

the requirement to train an external reward model or human annotations. A direct implementation of GRM involves the construction of assessment prompts and then using prior rules to assign scores to specific assessment tokens (e.g., `Yes` or `Not`) as the final reward score. (Mahan et al., 2024).

Inspired by this, We form the general assessment prompt for the evaluation as follows: `Evaluate whether the thought contributes to a partial or direct answer to the original question (likely/impossible)`. And then we assign a score, with `likely` = 10 and `impossible` = 1 to the assessment tokens. To reduce the randomness effect of the model, we perform multiple samplings and take the average of evaluation results.

Rollout and Collection. We use BFS with pruning as the search algorithm to select the reasoning steps. Once we obtain the reward for each candidate node, we select the top k nodes with the highest scores based on each solution's reward as child nodes for subsequent simulation or expansion phases. When the reasoning chain reaches the stop criteria, which includes the `</conclusion>`, the search algorithm returns the model answer, which will be compared with the ground truth y. And then caculate the $Q(s_t, a)$ for backpropogation of MCTS as Eq.9.

After complete a single iteration of the MCTS rollout, we split the reasoning paths into two sets according to the ground truth label (*i.e.*, correct solution set and incorrect solution set). We then rank the solution paths in each set by their average reward scores in descending order, ultimately selecting the top-k solutions from both sets, respectively. Considering high-scoring correct solutions tend to be of higher quality, while high-scoring incorrect solutions indicate challenging examples that the original reward model struggles to identify accurately. These two types of data constitute the preference dataset $\mathcal{D}$ and able to improving both the quality and difficulty of the training dataset.

$$Q(s_t, a) = \begin{cases} 1, & \text{if EXTRACT}(s_{t+1}) = y \\ 0, & \text{else} \end{cases} \tag{9}$$

4.3 Training with DPO Loss

Given the preference dataset $\mathcal{D}$ collected by MCTS, we finetune the policy model π_θ via DPO. For the final step of MCTS rollout, we get the preference data according to the ground truth. Considering prefered data y_w and dispreferred data y_l, to optimize the policy model π_θ on the pair of preference data (y_w, y_l), we can directly substitute it by Eq.6:

$$\begin{aligned} L(\pi_\theta; \pi_{\text{ref}}) = &-\log \sigma \left(\beta \log \left(\frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} \right) \right. \\ &\left. -\beta \log \left(\frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)} \right) \right) \end{aligned} \tag{10}$$

5 Experiments

5.1 Setups

Datasets. We assess the effectiveness of our method on mathematic and commonsense reasoning tasks. Our evaluation benchmarks encompass: 1) mathematic reasoning: GSM8K (Cobbe et al., 2021), which consists of grade school math word problems, and MATH (Hendrycks et al., 2021) featuring challenging competition math problems. 2) commonsense reasoning: ARC (only selected the challenge splits) (Clark et al., 2018) and SciQ (Welbl et al., 2017), containing science questions from student assessments. Performance evaluation is conducted using the corresponding validation sets of each dataset.

Table 1: Evaluation of the reasoning capabilities of our method in comparison to ICL methods and DPO baseline across four distinct reasoning benchmarks. The best results in each box are highlighted in **bold**.

MODEL	SETTING	MATHEMATIC		COMMONSENSE		AVERAGE
		MATH	GSM8K	ARC	SCIQ	
Llama-3.1-8B-Instruct	Zero-shot CoT	28.3	76.5	75.7	82.9	65.9
	Few-shot CoT	30.2	78.1	77.0	84.8	67.5
	CoT+SC@4	32.8	81.0	78.4	85.2	69.4
	DPO	35.6	80.7	82.5	89.4	72.1
	ours	**37.9**	**84.3**	**84.5**	**90.6**	**74.3**
Llama-3-8B-Instruct	Zero-shot CoT	6.8	68.4	73.0	81.1	57.3
	Few-shot CoT	17.6	73.8	75.6	84.7	62.9
	CoT+SC@4	26.1	76.3	77.4	89.2	67.3
	DPO	32.7	79.6	80.2	89.4	70.5
	ours	**33.4**	**82.7**	**82.2**	**91.1**	**72.4**
Qwen2.5-7B-instruct	Zero-shot CoT	35.8	81.7	76.3	78.4	68.1
	Few-shot CoT	39.6	87.2	78.1	83.1	72.0
	CoT+SC@4	44.1	90.6	80.5	89.6	76.2
	DPO	47.9	91.4	86.2	88.3	78.5
	ours	**50.0**	**92.1**	**89.6**	**94.0**	**81.4**
Qwen2-7B-instruct	Zero-shot CoT	34.9	74.3	73.8	76.2	64.8
	Few-shot CoT	38.0	80.4	76.4	82.3	69.3
	CoT+SC@4	41.2	83.6	78.9	87.6	72.8
	DPO	42.5	85.0	83.1	88.7	74.8
	ours	**44.5**	**86.1**	**85.0**	**90.8**	**76.6**
ChatGLM4-9B	Zero-shot CoT	34.2	78.0	83.5	86.4	70.5
	Few-shot CoT	37.6	81.4	87.1	91.9	74.5
	CoT+SC@4	36.1	84.7	88.6	92.1	75.4
	DPO	38.3	84.6	93.4	93.0	77.3
	ours	**41.6**	**85.9**	**94.3**	**93.5**	**78.8**

Models. Our method represents a versatile approach that can be applied to different LLMs. In our experiments, we evaluate its effectiveness using state-of-the-art opensource models: Llama3-8B-Instruct (Grattafiori et al., 2024), Llama-3.1-8B-Instruct (Meta, 2024), Qwen2-7B-Instruct (Yang et al., 2024a), Qwen2.5-7B-Instruct (Qwen, 2024), GLM-4-9B-Chat (Zeng et al., 2024a).

Baselines. We evaluate our method to three strong categories of baseline: 1) In-context Learning method (ICL method), including zero-shot CoT (Kojima et al., 2023), few-shot CoT (Wei et al., 2023) and CoT + SC (Wang et al., 2023). 2) Tree-based method, including ToT (Yao et al., 2023), RAP (Hao et al., 2023),LiteSearch(Wang et al., 2024a), ReST-MCTS* (Zhang et al., 2024a) 3) Synthesis methods, Self-Taught Reasoner (STaR) (Zelikman et al., 2022). 4) DPO baseline, we performed multiple samplings and and compared each reault with the final answer, reasoning paths leading to correct answers were considered positive samples, whereas those leading to incorrect answers were considered negative samples.

Implementation Details. The MCTS rollouts and supervised preference learning experiments are conducted with a maximum of 8 × 100GB GPUs (NVIDIA H20). For hyperparameter of MCTS rollouts, we set the discount factor $\gamma = 1.0$, rollout numbers $n = 16$, temperature set to 0.6, top p set to 0.95, top k set to 40, and max tokens set to 1024. To improve inference speed, we use vLLM [1] as the inference framework. For DPO learning, we use Low-Rank Adaptation (LoRA) (Hu et al., 2021) as our training method. We choose the learning rates 5e-6 with a cosine learning rate scheduler. The maximum sequence length of models is 1024. And we train the model with a batch size of 64. We follow the DPO paper to set the KL constraint parameter β as 0.1.

[1] https://github.com/vllm-project/vllm.

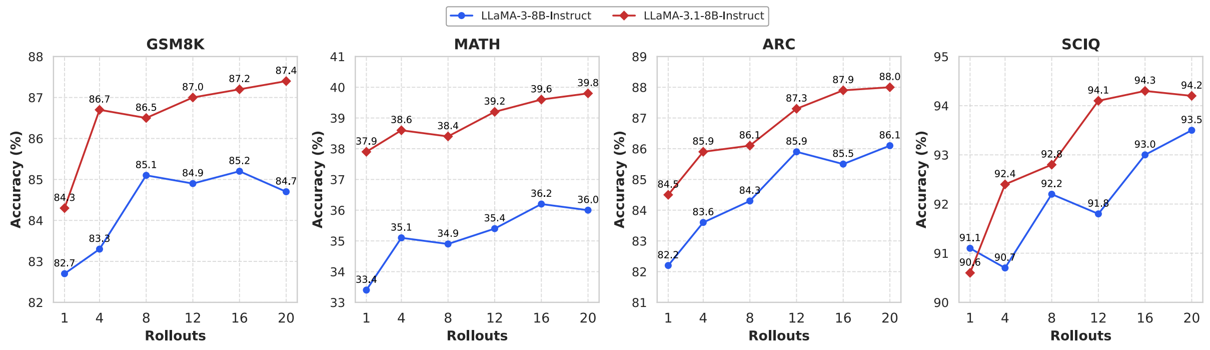

Figure 3: Results on test-time compute scaling show that the increment in the number of rollouts overall improves model performance across different benchmarks.

5.2 Main Results

As shown in Table 1, ICL method serves to boost the performance of the original LLMs to some extent, given that these tasks require improved step-by-step reasoning skills. Among these methods, CoT + SC deliver more superior performance as it effectively expand the solution space for challenging reasoning tasks. Furthermore, the DPO baseline achieved a notable improvement compared to ICL method. As a result, our method exhibits the most substantial performance improvements both on ICL method and DPO baseline. For example, Qwen2.5-7B-instruct achieve absolute accuracy increases of 2.3% from ICL to DPO, while increase of 1.9% from DPO to ours. And Llama-3.1-8B-Instruct achieve 2.7% from ICL to DPO as well as 2.2% from DPO to ours. These results underscore our approach's potential to efficiently guide LLMs in generating and selecting optimal solutions.

Table 2: Evaluating the reasoning capabilities of our method against tree-based methods and synthesis methods.

Model	Setting	MATH	GSM8K	ARC	SCIQ
Llama3.1-8B	ToT	31.4	78.3	75.9	80.3
	RAP	34.6	82.9	75.2	77.5
	ReST-MCTS*	35.2	80.3	76.3	79.1
	LiteSearch	33.5	81.9	74.5	78.7
	STaR	34.8	83.2	77.9	80.4
	Ours	**37.9**	**84.3**	**84.5**	**90.6**
Qwen2.5-7B	ToT	43.9	88.7	82.1	87.8
	RAP	44.3	90.2	83.4	90.5
	ReST-MCTS*	46.5	91.8	84.0	89.3
	LiteSearch	45.4	89.2	85.5	88.2
	STaR	47.1	91.4	86.3	91.7
	Ours	**50.0**	**92.1**	**89.6**	**94.0**
ChatGLM4-9B	ToT	35.8	81.2	86.6	87.2
	RAP	38.9	82.3	87.9	91.1
	ReST-MCTS*	40.3	85.6	92.1	88.7
	LiteSearch	39.2	84.9	88.3	92.4
	STaR	41.1	84.3	93.5	92.8
	Ours	**41.6**	**85.9**	**94.3**	**93.5**

Furthermore, we compare our approach with other tree-based reasoning methods and synthesis methods on GSM8K and MATH. As shown in Table 2, our method reveals superior performance and impressive generalizability across diverse models and datasets. Remarkably, as the complexity of benchmarks increases, existing methods such as ToT and RAP experience considerable challenges. However, methods like STaR and LiteSearch show performance comparable to ours on Llama3-8B-Instruct, performance persist also on model like Llama3.1-8B-Instruct. Consequently, our method demonstrates a clear performance improvement compared to other baselines.

5.3 Further Analysis

Test-Time Compute Scaling. To investigate the potential and emerging trends of scaling with rollouts during inference time, we present the pass rates of problem-solving by increasing rollouts across three benchmarks of varying difficulty levels. As shown in Figure 3, the increment in the number of rollouts overall improves model performance across different benchmarks, and degree of these improvements varies based on the complexity of the benchmark and the reasoning abilities of the base model. The results above highlight that our framework's performance improves with an increased number of rollouts during inference. However, there are ceiling limitations reveals in the mathematical task, which suggest that the abilities of the base model in reasoning are essential to the overall performance.

Ablation Study. We ablate the impact of atom reasoning steps on our MCTS-based approach. Figure 4 shows performance comparisons across mathematic and commonsense reasoning tasks under different settings. Our method, which focuses on

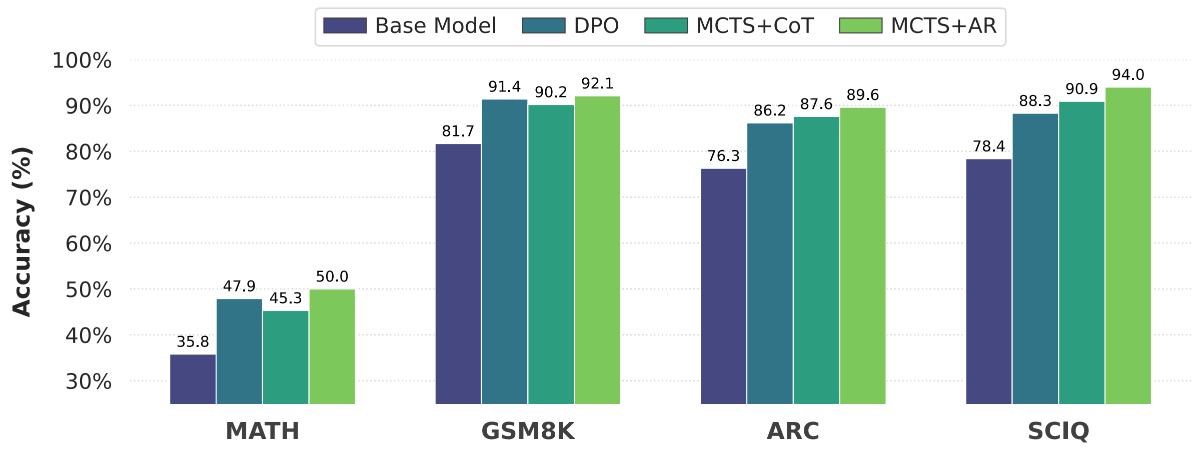

Figure 4: Abltation study on MCTS+CoT v.s. MCTS+AR. We also compare the accuracy of the reasoning paths collected via MCTS by chain-of-thought and atom reasoning steps on Qwen2.5-7B-instruct.

constructing reasoning paths through atom actions, consistently outperforms both the base model and the CoT counterpart. For example, we achieve 89.6% on ARC and 94.0% on SciQ, surpassing 84.6% and 89.9% of the CoT counterpart, and 76.3% and 78.4% of the base model. This result indicates that atom actions significantly enhance the reasoning capabilities of the model.

Complexity Level Analysis. As shown in Table 3, we showcase the results of Few-shot CoT, Few-shot CoT+SC, and our method on the MATH dataset, assessed across various levels of difficulty. Compared to the first two methods, our approach improved performance at all levels. Notably, at the more difficult levels 4-5, our method exhibits an average improvement of 6.8% compared to Few-shot CoT, while Few-shot CoT+SC shows only a 1.5% improvement. This indicates that atom reasoning steps, as a form of slow thinking, have the potential to tackle more challenging problems and enhance reasoning performance.

METHOD	1	2	3	4	5	AVERAGE
CoT	54.1	46.7	44.3	36.1	28.4	39.6
CoT+SC	58.7	50.1	48.3	38.4	29.0	44.1
ours	63.1	58.4	55.3	43.6	34.5	50.0

Table 3: Performance variations across different difficulty levels on MATH with Qwen2.5-7B-Instruct as base model.

6 Conclusion

In this paper, we propose MCTS-enhanced self-preference learning framework, utilizing MCTS to generate high quality reasoning paths for DPO training. MCTS balances quality exploitation and diversity exploration to produce high-quality training data, efficiently pushing the ceiling performance of the LLM on various reasoning tasks. Additionally, the introduction of atom reasoning steps and self-evaluation mechanism improves the reliability and human-like quality of the model's reasoning These results demonstrate the effectiveness and efficiency of Atom Reasoner, offering a scalable framework for developing large language models capable of handling intricate reasoning tasks.

7 Limitations

We acknowledge that this work has several limitations that require further exploration. First of all, we optimize the entire reasoning path for DPO, disregarding the optimization of each atomic step, which may encounter longest common prefix (LCP) gradient cancelation (Zhang et al., 2024c). Furthermore, the preference datasets are collected ahead of training and our method is purely offline, as π_θ cannot get feedback on its own generations over training.This may lead to significant distribution shift between the policy that generated the dataset and the policy being aligned (Guo et al., 2024).

Ethics Statement

Our proposed method Atom Reasoner effectively improves the ability of complex problems reasoning of LLMs without the need for larger models for data distillation. Therefore, this process does not entail additional risks.

We utilize a variety of open-source English datasets for training, including MATH, GSM8K, ARC and SCIQ. And we utilize a variety of open-source model for fintuning, including Qwen2.5-7B-Instruct and so on. We acknowledge that there may be inherent biases present within these datasets and the model.

References

Renat Aksitov, Sobhan Miryoosefi, Zonglin Li, Daliang Li, Sheila Babayan, Kavya Kopparapu, Zachary Fisher, Ruiqi Guo, Sushant Prakash, Pranesh Srinivasan, Manzil Zaheer, Felix Yu, and Sanjiv Kumar. 2023. Rest meets react: Self-improvement for multi-step reasoning llm agent.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of thoughts: Solving elaborate problems with large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690.

Bradley and Terry. 1952. Rank analysis of incomplete block designs: The method of paired comparisons. *Biometrika*, 39(3-4):324–345.

Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. 2012. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43.

Guillaume Chaslot, Sander Bakkes, Istvan Szita, and Pieter Spronck. 2021. Monte-carlo tree search: A new framework for game ai. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 4(1):216–217.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. 2024. Self-play fine-tuning converts weak language models to strong language models.

Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2023. Deep reinforcement learning from human preferences.

Michael Chung, Michael Buro, and Jonathan Schaeffer. 2005. Monte carlo planning in rts games.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems.

Xidong Feng, Ziyu Wan, Muning Wen, Stephen Marcus McAleer, Ying Wen, Weinan Zhang, and Jun Wang. 2024. Alphazero-like tree-search can guide large language model decoding and training.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, and Abhinav Pandey. 2024. The llama 3 herd of models.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, Wolfgang Macherey, Arnaud Doucet, Orhan Firat, and Nando de Freitas. 2023. Reinforced self-training (rest) for language modeling.

Shangmin Guo, Biao Zhang, Tianlin Liu, Tianqi Liu, Misha Khalman, Felipe Llinares, Alexandre Rame, Thomas Mesnard, Yao Zhao, Bilal Piot, Johan Ferret, and Mathieu Blondel. 2024. Direct language model alignment from online ai feedback.

Thilo Hagendorff, Sarah Fabi, and Michal Kosinski. 2023. Human-like intuitive behavior and reasoning biases emerged in large language models but disappeared in chatgpt. *Nature Computational Science*, 3(10):833–838.

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.

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

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models.

Yixin Ji, Juntao Li, Hai Ye, Kaixin Wu, Jia Xu, Linjian Mo, and Min Zhang. 2025. Test-time computing: from system-1 thinking to system-2 thinking.

Jinhao Jiang, Zhipeng Chen, Yingqian Min, Jie Chen, Xiaoxue Cheng, Jiapeng Wang, Yiru Tang, Haoxiang Sun, Jia Deng, Wayne Xin Zhao, Zheng Liu, Dong Yan, Jian Xie, Zhongyuan Wang, and Ji-Rong Wen. 2024. Technical report: Enhancing llm reasoning with reward-guided tree search.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared

Kaplan. 2022. Language models (mostly) know what they know.

Levente Kocsis and Csaba Szepesvári. 2006. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006*, pages 282–293, Berlin, Heidelberg. Springer Berlin Heidelberg.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2023. Large language models are zero-shot reasoners.

Nicholas Lee, Thanakul Wattanawong, Sehoon Kim, Karttikeya Mangalam, Sheng Shen, Gopala Anumanchipalli, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. 2024. Llm2llm: Boosting llms with novel iterative data enhancement.

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.

Jiacheng Liu, Andrew Cohen, Ramakanth Pasunuru, Yejin Choi, Hannaneh Hajishirzi, and Asli Celikyilmaz. 2024a. Don't throw away your value model! generating more preferable text with value-guided monte-carlo tree search decoding.

Zichen Liu, Siyi Li, Wee Sun Lee, Shuicheng Yan, and Zhongwen Xu. 2023. Efficient offline policy optimization with a learned model.

Zixuan Liu, Xiaolin Sun, and Zizhan Zheng. 2024b. Enhancing llm safety via constrained direct preference optimization.

Jieyi Long. 2023. Large language model guided tree-of-thought.

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

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. 2023. Self-refine: Iterative refinement with self-feedback.

Dakota Mahan, Duy Van Phung, Rafael Rafailov, Chase Blagden, Nathan Lile, Louis Castricato, Jan-Philipp Fränken, Chelsea Finn, and Alon Albalak. 2024. Generative reward models.

Meta. 2024. Introducing llama 3.1.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, and Igor Babuschkin. 2024. Gpt-4 technical report.

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 Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback.

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

Yiwei Qin, Xuefeng Li, Haoyang Zou, Yixiu Liu, Shijie Xia, Zhen Huang, Yixin Ye, Weizhe Yuan, Hector Liu, Yuanzhi Li, and Pengfei Liu. 2024. O1 replication journey: A strategic progress report – part 1.

Qwen. 2024. Qwen2.5: A party of foundation models.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model.

Christopher D. Rosin. 2011. Multi-armed bandits with episode context. *Annals of Mathematics and Artificial Intelligence*, 61:203–230.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. 2017. Mastering chess and shogi by self-play with a general reinforcement learning algorithm.

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.

Ante Wang, Linfeng Song, Ye Tian, Baolin Peng, Dian Yu, Haitao Mi, Jinsong Su, and Dong Yu. 2024a. Litesearch: Efficacious tree search for llm.

Chaojie Wang, Yanchen Deng, Zhiyi Lyu, Liang Zeng, Jujie He, Shuicheng Yan, and Bo An. 2024b. Q*: Improving multi-step reasoning for llms with deliberative planning.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions.

Jinyang Wu, Mingkuan Feng, Shuai Zhang, Feihu Che, Zengqi Wen, and Jianhua Tao. 2024. Beyond examples: High-level automated reasoning paradigm in in-context learning via mcts.

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, Xu Zhao, Min-Yen Kan, Junxian He, and Qizhe Xie. 2023. Self-evaluation guided beam search for reasoning.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, and Chang Zhou. 2024a. Qwen2 technical report.

Zonghan Yang, Peng Li, Ming Yan, Ji Zhang, Fei Huang, and Yang Liu. 2024b. React meets actre: When language agents enjoy training data autonomy.

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.

Yao Yao, Zuchao Li, and Hai Zhao. 2024. Beyond chain-of-thought, effective graph-of-thought reasoning in language models.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. 2022. Star: Bootstrapping reasoning with reasoning.

Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, Hongning Wang, Jiadai Sun, Jiajie Zhang, Jiale Cheng, Jiayi Gui, Jie Tang, Jing Zhang, Jingyu Sun, Juanzi Li, Lei Zhao, Lindong Wu, Lucen Zhong, Mingdao Liu, Minlie Huang, Peng Zhang, Qinkai Zheng, Rui Lu, Shuaiqi Duan, Shudan Zhang, Shulin Cao, Shuxun Yang, Weng Lam Tam, Wenyi Zhao, Xiao Liu, Xiao Xia, Xiaohan Zhang, Xiaotao Gu, Xin Lv, Xinghan Liu, Xinyi Liu, Xinyue Yang, Xixuan Song, Xunkai Zhang, Yifan An, Yifan Xu, Yilin Niu, Yuantao Yang, Yueyan Li, Yushi Bai, Yuxiao Dong, Zehan Qi, Zhaoyu Wang, Zhen Yang, Zhengxiao Du, Zhenyu Hou, and Zihan Wang. 2024a. Chatglm: A family of large language models from glm-130b to glm-4 all tools.

Yongcheng Zeng, Guoqing Liu, Weiyu Ma, Ning Yang, Haifeng Zhang, and Jun Wang. 2024b. Token-level direct preference optimization.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024a. Rest-mcts*: Llm self-training via process reward guided tree search.

Di Zhang, Jianbo Wu, Jingdi Lei, Tong Che, Jiatong Li, Tong Xie, Xiaoshui Huang, Shufei Zhang, Marco Pavone, Yuqiang Li, Wanli Ouyang, and Dongzhan Zhou. 2024b. Llama-berry: Pairwise optimization for o1-like olympiad-level mathematical reasoning.

Xuan Zhang, Chao Du, Tianyu Pang, Qian Liu, Wei Gao, and Min Lin. 2024c. Chain of preference optimization: Improving chain-of-thought reasoning in llms.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. 2023. Least-to-most prompting enables complex reasoning in large language models.