# Monte Carlo Tree Search Boosts Reasoning via Iterative Preference Learning

Anonymous ACL submission

## Abstract

We introduce an approach aimed at enhancing the reasoning capabilities of Large Language Models (LLMs) through an iterative preference learning process inspired by the successful strategy employed by AlphaZero. Our work leverages Monte Carlo Tree Search (MCTS) to iteratively collect preference data, utilizing its look-ahead ability to break down instance-level rewards into more granular step-level signals. To enhance consistency in intermediate steps, we combine outcome validation and stepwise self-evaluation, continually updating the quality assessment of newly generated data. The proposed algorithm employs Direct Preference Optimization (DPO) to update the LLM policy using this newly generated step-level preference data. Theoretical analysis reveals the importance of using on-policy sampled data for successful self-improving. Extensive evaluations on various arithmetic and commonsense reasoning tasks demonstrate remarkable performance improvements over existing models. For instance, our approach outperforms the Mistral-7B Supervised Fine-Tuning (SFT) baseline on GSM8K, MATH, and ARC-C, with substantial increases in accuracy to $81.8\%$ (+$5.9\%$), $34.7\%$ (+$5.8\%$), and $76.4\%$ (+$15.8\%$), respectively. Additionally, our research delves into the training and inference compute tradeoff, providing insights into how our method effectively maximizes performance gains.

## 1 Introduction

Development of Large Language Models (LLMs), has seen a pivotal shift towards aligning these models more closely with human values and preferences (Stiennon et al., 2020; Ouyang et al., 2022; Bai et al., 2022a). A critical aspect of this process involves the utilization of preference data. There are two prevailing methodologies for incorporating this data: the first entails the construction of a reward model based on preferences, which is then integrated into a Reinforcement Learning (RL) framework to update the policy (Christiano et al., 2017; Bai et al., 2022b); the second, more stable and scalable method, directly applies preferences to update the model's policy (Rafailov et al., 2023).

In this context, the concept of "iterative" development is a key, especially when contrasted with the conventional Reinforcement Learning from Human Feedback (RLHF) paradigm (Christiano et al., 2017; Stiennon et al., 2020; Ouyang et al., 2022; Bai et al., 2022a), where the reward model is often trained offline and remains static. An iterative approach proposes a dynamic and continuous refinement process (Zelikman et al., 2022; Gülçehre et al., 2023; Huang et al., 2023; Yuan et al., 2024). It involves a cycle that begins with the current policy, progresses through the collection and analysis of data to generate new preference data, and uses this data to update the policy. This approach underlines the importance of ongoing adaptation in LLMs, highlighting the potential for these models to become more attuned to the complexities of human decision-making and reasoning.

A compelling illustration of the success of such an iterative approach can be seen in the case of AlphaZero (Silver et al., 2017) for its superhuman performance across various domains, which combines the strengths of neural networks, RL techniques, and Monte Carlo Tree Search (MCTS) (Coulom, 2006; Kocsis and Szepesvári, 2006). The integration of MCTS as a *policy improvement operator* that transforms the current policy into an improved policy (Grill et al., 2020). The effectiveness of AlphaZero underscores the potential of combining these advanced techniques in LLMs. By integrating MCTS into the iterative process of policy development, it is plausible to achieve significant strides in LLMs, particularly in the realm of reasoning and decision-making aligned with human-like preferences (Zhu et al., 2023; Hao et al., 2023).

The integration of MCTS in collecting preference data to improve the current policy iteratively

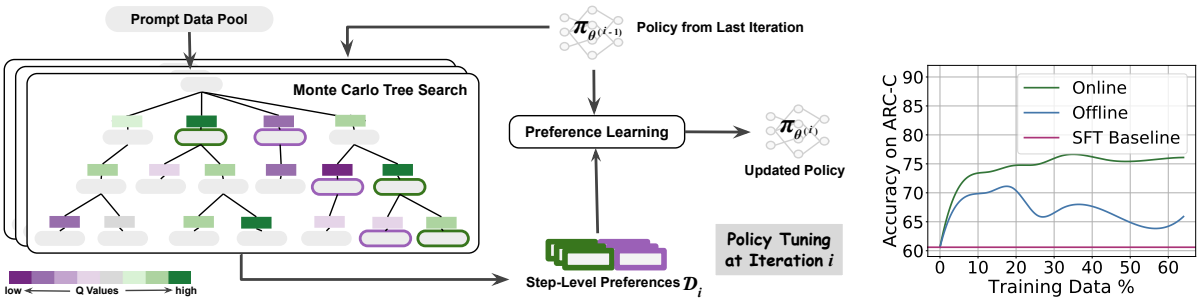


Figure 1: Monte Carlo Tree Search (MCTS) boosts model performance via iterative preference learning. Each iteration of our framework (on the left) consists of two stages: MCTS to collect step-level preferences and preference learning to update the policy. Specifically, we use action values $Q$ estimated by MCTS to assign the preferences, where steps of higher and lower $Q$ values will be labeled as positive and negative data, respectively. The scale of $Q$ is visualized in the colormap. We show the advantage of the online manner in our iterative learning framework using the validation accuracy curves as training progresses on the right. The performance of ARC-C validation illustrates the effectiveness and efficiency of our proposed method compared to its offline variant.

is nuanced and demands careful consideration. One primary challenge lies in determining the appropriate granularity for applying MCTS. Conventionally, preference data is collected at the instance level. The instance-level approach employs sparse supervision, which can lose important information and may not optimally leverage the potential of MCTS in improving the LLMs (Wu et al., 2023). Another challenge is the reliance of MCTS on a critic or a learned reward function. This function is crucial for providing meaningful feedback on different rollouts generated by MCTS, thus guiding the policy improvement process (Liu et al., 2023a).

Addressing this granularity issue, evidence from LLM research indicates the superiority of process-level or stepwise evaluations over instance-level ones (Lightman et al., 2023; Li et al., 2023; Xie et al., 2023; Yao et al., 2023). Our approach utilizes MCTS rollouts for step-level guidance, aligning with a more granular application of MCTS. Moreover, we employ self-evaluation, where the model assesses its outputs, fostering a more efficient policy improvement pipeline by acting as both policy and critic (Kadavath et al., 2022; Xie et al., 2023). This method streamlines the process and ensures more cohesive policy updates, aligning with the iterative nature of policy enhancement and potentially leading to more robust and aligned LLMs.

To summarize, we propose an algorithm based on Monte Carlo Tree Search (MCTS) that breaks down the instance-level preference signals into step-level. MCTS allows us to use the current LLM policy to generate preference data instead of a predetermined set of human preference data, enabling the LLM to receive real-time training signals. During training, we generate sequences of text on the fly and label the preference via MCTS based on feedback from self-evaluation (Figure 1). To update the LLM policy using the preference data, we use Direct Preference Optimization (DPO) (Rafailov et al., 2023). We extensively evaluate the proposed approach on various arithmetic and commonsense reasoning tasks and observe significant performance improvements. For instance, the proposed approach outperforms the Mistral-7B SFT baseline by $81.8\%$ (+5.9%), $34.7\%$ (+5.8%), and $76.4\%$ (+15.8%) on GSM8K, MATH, and SciQ, respectively. Further analysis of the training and test compute tradeoff shows that our method can effectively push the performance gains in a more efficient way compared to sampling-only approaches.

## 2 MCTS-Enhanced Iterative Preference Learning

In this paper, we introduce an approach for improving LLM reasoning, centered around an iterative preference learning process. The proposed method begins with an initial policy $\pi_{\theta^{(0)}}$, and a dataset of prompts $\mathcal{D}_{\mathcal{P}}$. Each iteration $i$ involves selecting a batch of prompts from $\mathcal{D}_{\mathcal{P}}$, from which the model, guided by its current policy $\pi_{\theta^{(i-1)}}$, generates potential responses for each prompt. We then apply a set of dynamically evolving reward criteria to extract preference data $\mathcal{D}_i$ from these responses. The model's policy is subsequently tuned using this preference data, leading to an updated policy $\pi_{\theta^{(i)}}$, for the next iteration. This cycle of sampling, response generation, preference extraction, and policy tuning is repeated, allowing for continuous self-improvement and alignment with evolving preferences. In addressing the critical aspects of

this methodology, two key challenges emerge: the effective collection of preference data and the process of updating the policy post-collection.

We draw upon the concept that MCTS can act as an approximate policy improvement operator, transforming the current policy into an improved one. Our work leverages MCTS to iteratively collect preference data, utilizing its look-ahead ability to break down instance-level rewards into more granular step-level signals. To enhance consistency in intermediate steps, we incorporate stepwise self-evaluation, continually updating the quality assessment of newly generated data. This process, as depicted in Figure 1, enables MCTS to balance quality exploitation and diversity exploration during preference data sampling at each iteration. Detailed in section 2.1, our approach utilizes MCTS for step-level preference data collection. Once this data is collected, the policy is updated using DPO, as outlined in section 2.2. Our method can be viewed as an online version of DPO, where the updated policy is iteratively employed to collect preferences via MCTS. Our methodology, thus, not only addresses the challenges in preference data collection and policy updating but also introduces a dynamic, iterative framework that significantly enhances LLM reasoning.

### 2.1 MCTS for Step-Level Preference

To transform instance-level rewards into granular, step-level signals, we dissect the reasoning process into discrete steps, each represented by a token sequence. We define the state at step $t$, $s_t$, as 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's current policy $\pi_\theta$, we sample candidate steps from its probability distribution $\pi_\theta(a \mid x, s_t)$[1], with $x$ being the task's input prompt. MCTS serves as an approximate policy improvement operator by leveraging its look-ahead capability to predict the expected future reward. This prediction is refined through stepwise self-evaluation (Kadavath et al., 2022; Xie et al., 2023), enhancing process consistency and decision accuracy. 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 sentence start or incomplete response, and unfolds in three iterative stages: selection, expansion, and backup, which we detail further.

**Select.** The objective of this phase is to identify nodes that balance search quality and computational efficiency. The selection is guided by two key variables: $Q(s_t, a)$, the value of taking action $a$ in state $s_t$, and $N(s_t)$, the visitation frequency of state $s_t$. These variables are crucial for updating the search strategy, as explained in the backup section. To navigate the trade-off between exploring new nodes and exploiting visited ones, we employ the Predictor + Upper Confidence bounds applied to Trees (PUCT) (Rosin, 2011). At node $s_t$, the choice of the subsequent node follows the formula:

$$s_{t+1}{}^* = \arg\max_{s_t} \Big[ Q(s_t, a) + c_{\text{puct}} \cdot p(a \mid s_t) \frac{\sqrt{N(s_t)}}{1 + N(s_{t+1})} \Big] \tag{1}$$

where $p(a \mid s_t) = \pi_\theta(a \mid x, s_t)/|a|^\lambda$ denotes the policy $\pi_\theta$'s probability distribution for generating a step $a$, adjusted by a $\lambda$-weighted length penalty to prevent overly long reasoning chains.

**Expand.** Expansion occurs at a leaf node during the selection process to integrate new nodes and assess rewards. The reward $r(s_t, a)$ for executing step $a$ in state $s_t$ is quantified by the reward difference between states $R(s_t)$ and $R(s_{t+1})$, highlighting the advantage of action $a$ at $s_t$. As defined in Eq. (2), reward computation merges outcome correctness $\mathcal{O}$ with self-evaluation $\mathcal{C}$. We assign the outcome correctness to be $1$, $-1$, and $0$ for correct terminal, incorrect terminal, and unfinished intermediate states, respectively. Following Xie et al. (2023), we define self-evaluation as Eq. (3), where A denotes the confidence score in token-level probability for the option indicating correctness[2]. Future rewards are anticipated by simulating upcoming scenarios through roll-outs, following the selection and expansion process until reaching a terminal state[3].

$$R(s_t) = \mathcal{O}(s_t) + \mathcal{C}(s_t) \tag{2}$$

$$\mathcal{C}(s_t) = \pi_\theta(\mathrm{A} \mid \text{prompt}_{\text{eval}}, x, s_t) \tag{3}$$

[1]For tasks (*e.g.*, MATH) where the initial policy performs poorly, we also include the ground-truth reasoning steps for training. We detail the step definition for different tasks with examples in Appendices C and D.

[2]We show an example of evaluation prompt in Table 6.

[3]The terminal state is reached when the whole response is complete or exceeds the maximum length.

**Algorithm 1 . MCTS-Enhanced Iterative Preference Learning.** Given an initial policy $\pi_{\theta^{(0)}} = \pi_{\text{sft}}$, our algorithm iteratively conducts step-level preference data sampling via MCTS and preference learning via DPO to update the policy.

**Input:** $\mathcal{D}_{\mathcal{P}}$: prompt dataset; $q(\cdot \mid x)$: MCTS sampling strategy that constructs a tree-structured set of possible responses given a prompt $x$, where $q_\pi$ represents that the strategy is based on the policy $\pi$ for both response generation and self-evaluation; $\ell_i(x, y_w, y_l; \theta)$: loss function of preference learning at the $i$-th iteration, where the corresponding sampling policy is $\pi^{(i)}$; $M$: number of iterations; $B$: number of samples per iteration; $T$: average number of steps per sample
Train $\pi_\theta$ on $\mathcal{D}_{\mathcal{P}}$ using step-level preference learning.
**for** $i = 1$ **to** $M$ **do**
  $\pi^{(i)} \leftarrow \pi_\theta \leftarrow \pi_{\theta^{(i-1)}}$
  Sample a batch of $B$ samples from $\mathcal{D}_{\mathcal{P}}$ as $\mathcal{D}_{\mathcal{P}}^{(i)}$.
  /* MCTS for Step-Level Preference Data Collection */
  For each $x \in \mathcal{D}_{\mathcal{P}}^{(i)}$, elicit a search tree of depth $T$ via $q_{\pi_\theta}(\cdot \mid x)$.
  Collect a batch of preferences $\mathcal{D}_i = \{ \{(x^j, y_l^{(j,t)}, y_l^{(j,t)})|_{t=1}^{T}\}|_{j=1}^{B}$ s.t. $x^j \sim \mathcal{D}_{\mathcal{P}}^{(i)}, y_w^{(j,t)} \neq y_w^{(j,t)} \sim q_{\pi_\theta}(\cdot \mid x^j)$ $\}$, where $y_w^{(j,t)}$ and $y_l^{(j,t)}$ is the nodes at depth $t$, with the highest and lowest $Q$ values, respectively, among all the children nodes of their parent node.
  /* Preference Learning for Policy Improvement */
  Optimize $\theta$ by minimizing $J(\theta) = \mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}_i} \ell_i(x, y_w, y_l; \theta)$.
  Obtain the updated policy $\pi_{\theta^{(i)}}$
**end for**
$\pi_\theta \leftarrow \pi_{\theta^{(M)}}$
**Output:** Policy $\pi_\theta$

**Backup.** Once a terminal state is reached, we carry out a bottom-up update from the terminal node back to the root. We update the visit count $N$, the state value $V$, and the transition value $Q$:

$$Q(s_t, a) \leftarrow r(s_t, a) + \gamma V(s_{t+1}) \tag{4}$$

$$V(s_t) \leftarrow \sum_a N(s_{t+1}) Q(s_t, a) / \sum_a N(s_{t+1}) \tag{5}$$

$$N(s_t) \leftarrow N(s_t) + 1 \tag{6}$$

where $\gamma$ is the discount for future state values.

For each step in the response generation, we conduct $K$ iterations of MCTS to construct the search tree while updating $Q$ values and visit counts $N$. To balance the diversity, quality, and efficiency of the tree construction, we initialize the search breadth as $b_1$ and anneal it to be a smaller $b_2 < b_1$ for the subsequent steps. We use the result $Q$ value corresponding to each candidate step to label its preference, where higher $Q$ values indicate preferred next steps. For a result search tree of depth $T$, we obtain $T$ pairs of step-level preference data. Specifically, we select the candidate steps of highest and lowest $Q$ values as positive and negative samples at each tree depth, respectively. The parent node selected at each tree depth has the highest value calculated by multiplying its visit count and the range of its children nodes' visit counts, indicating both the quality and diversity of the generations.

## 2.2 Iterative Preference Learning

Given the step-level preferences collected via MCTS, we tune the policy via DPO (Rafailov et al., 2023). Considering the noise in the preference labels determined by $Q$ values, we employ the conservative version of DPO (Mitchell, 2023) and use the visit counts simulated in MCTS to apply adaptive label smoothing on each preference pair. Using the shorthand $h_{\pi_\theta}^{y_w,y_l} = \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}$, at the $i$-th iteration, given a batch of preference data $\mathcal{D}_i$ sampled with the latest policy $\pi_{\theta^{(i-1)}}$, we denote the policy objective $\ell_i(\theta)$ as follows:

$$\ell_i(\theta) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}_i} \Big[ (1 - \alpha_{x,y_w,y_l}) \log \sigma(\beta\, h_{\pi_\theta}^{y_w,y_l}) + \alpha_{x,y_w,y_l} \log \sigma(-\beta h_{\pi_\theta}^{y_w,y_l}) \Big] \tag{7}$$

where $y_w$ and $y_l$ represent the step-level preferred and dispreferred responses, respectively, and the hyperparameter $\beta$ scales the KL constraint. Here, $\alpha_{x,y_w,y_l}$ is a label smoothing variable calculated using the visit counts at the corresponding states of the preference data $y_w$, $y_l$ in the search tree:

$$\alpha_{x,y_w,y_l} = \frac{1}{N(x, y_w)/N(x, y_l) + 1} \tag{8}$$

where $N(x, y_w)$ and $N(x, y_l)$ represent the states taking the actions of generating $y_w$ and $y_l$, respectively, from their previous state as input $x$.

After optimization, we obtain the updated policy $\pi_{\theta^{(i)}}$ and repeat the data collection process in Section 2.1 to iteratively update the LLM policy. We

outline the full algorithm of our MCTS-enhanced Iterative Preference Learning in Algorithm 1.

## 3 Experiments

We evaluate the effectiveness of MCTS-enhanced iterative preference learning on arithmetic and commonsense reasoning tasks. We employ Mistral-7B (Jiang et al., 2023) as the base pre-trained model. We conduct supervised training using Arithmo [4] which comprises approximately 540K mathematical and coding problems. Detailed information regarding the task formats, specific implementation procedures, and parameter settings of our experiments can be found in Appendix C.

**Datasets.** We aim to demonstrate the effectiveness and versatility of our approach by focusing on two types of reasoning: arithmetic and commonsense reasoning. For arithmetic reasoning, we utilize two datasets: GSM8K (Cobbe et al., 2021), which consists of grade school math word problems, and MATH (Hendrycks et al., 2021), featuring challenging competition math problems. Specifically, in the GSM8K dataset, we assess both chain-of-thought (CoT) and program-of-thought (PoT) reasoning abilities. We integrate the training data from GSM8K and MATH to construct the prompt data for our preference learning framework, aligning with a subset of the Arithmo data used for Supervised Fine-Tuning (SFT). This approach allows us to evaluate whether our method enhances reasoning abilities on specific arithmetic tasks. For commonsense reasoning, we use four multiple-choice datasets: ARC (easy and challenge splits) (Clark et al., 2018), focusing on science exams; AI2Science (elementary and middle splits) (Clark et al., 2018), containing science questions from student assessments; OpenBookQA (OBQA) (Mihaylov et al., 2018), which involves open book exams requiring broad common knowledge; and CommonSenseQA (CSQA) (Talmor et al., 2019), featuring commonsense questions necessitating prior world knowledge. The diversity of these datasets, with different splits representing various grade levels, enables a comprehensive assessment of our method's generalizability in learning various reasoning tasks through self-distillation. Performance evaluation is conducted using the corresponding validation sets of each dataset. Furthermore, we employ an unseen evaluation using the validation set of an additional dataset, SciQ (Welbl et al., 2017), following the approach of Liu et al. (2023b), to further test our model's ability to generalize to novel reasoning contexts.

**Baselines.** Our study involves a comparative evaluation of our method against several prominent approaches and fair comparison against variants including instance-level iterative preference learning and offline MCTS-enhanced learning. We use instance-level sampling as a counterpart of step-level preference collection via MCTS. For a fair comparison, we also apply self-evaluation and correctness assessment and control the number of samples under a comparable compute budget with MCTS in instance-level sampling. The offline version uses the initial policy for sampling rather than the updated one at each iteration.

We contrast our approach with the Self-Taught Reasoner (STaR)(Zelikman et al., 2022), an iterated learning model based on instance-level rationale generation, and Crystal(Liu et al., 2023b), an RL-tuned model with a focus on knowledge introspection in commonsense reasoning. Considering the variation in base models used by these methods, we include comparisons with Direct Tuning, which entails fine-tuning base models directly bypassing chain-of-thought reasoning. In the context of arithmetic reasoning tasks, our analysis includes Language Model Self-Improvement (LMSI)(Huang et al., 2023), a self-training method using self-consistency to gather positive data, and Math-Shepherd(Wang et al., 2023a), which integrates process supervision within Proximal Policy Optimization (PPO). To account for differences in base models and experimental setups across these methods, we also present result performance of SFT models as baselines for each respective approach.

### 3.1 Main Results

**Arithmetic Reasoning.** In Table 1, we present a comparative analysis of performance gains in arithmetic reasoning tasks. Our method demonstrates substantial improvements, notably on GSM8K, increasing from $75.9\% \rightarrow 81.8\%$, and on MATH, enhancing from $28.9\% \rightarrow 34.7\%$. When compared to Math-Shepherd, which also utilizes process supervision in preference learning, our approach achieves similar performance enhancements without the necessity of training separate reward or value networks. This suggests the potential of integrating trained reward model signals into our MCTS stage to further augment performance. On

[4] https://huggingface.co/datasets/akjindal53244/Arithmo-Data

| Approach | Base Model | Conceptual Comparison | | | | GSM8K | MATH |
|---|---|---|---|---|---|---|---|
| | | NR | OG | OF | NS | | |
| LMSI | PaLM-540B | ✓ | ✓ | ✗ | ✗ | 73.5 | – |
| SFT (MetaMath) | Mistral-7B | – | – | – | – | 77.7 | 28.2 |
| Math-Shepherd | | ✗ | ✓ | ✗ | ✓ | 84.1$\uparrow_{6.4}$ | 33.0$\uparrow_{4.8}$ |
| SFT (Arithmo) | Mistral-7B | – | – | – | – | 75.9 | 28.9 |
| MCTS Offline-DPO | | ✓ | ✗ | ✗ | ✓ | 79.9 | 31.9 |
| Instance-level Online-DPO | | ✓ | ✓ | ✓ | ✓ | 79.9 | 31.9 |
| Ours | | ✓ | ✓ | ✓ | ✓ | 80.7 | 32.2 |
| Ours (w/ G.T.) | | ✓ | ✓ | ✓ | ✓ | 81.8$\uparrow_{5.9}$ | 34.7$\uparrow_{5.8}$ |

Table 1: Result comparison (accuracy %) on arithmetic tasks. We supervised fine-tune the base model Mistral-7B on Arithmo data, while Math-Shepherd (Wang et al., 2023a) use MetaMATH (Yu et al., 2023b) for SFT. We highlight the advantages of our approach via conceptual comparison with other methods, where NR, OG, OF, and NS represent "w/o Reward Model", "On-policy Generation", "Online Feedback", and "w/ Negative Samples".

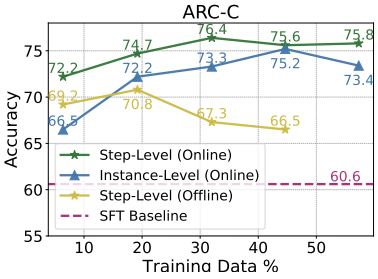


Figure 2: Performance on the validation set of ARC-C via training with different settings.

| Approach | Base Model | Conceptual Comparison | | | | ARC-c | AI2Sci-m | CSQA | SciQ | Train Data Used (%) |
|---|---|---|---|---|---|---|---|---|---|---|
| | | NR | OG | OF | NS | | | | | |
| CoT Tuning | GPT-3-curie (6.7B) | ✓ | ✗ | ✗ | ✗ | – | – | 56.8 | – | 100 |
| Direct Tuning | GPT-J (6B) | ✓ | ✗ | ✗ | ✗ | – | – | 60.0 | – | 100 |
| STaR | | ✓ | ✓ | ✓ | ✗ | – | – | 72.5 | – | 86.7 |
| Direct TUning | T5-11B | ✓ | ✗ | ✗ | ✗ | 72.9 | 84.0 | 82.0 | 83.2 | 100 |
| Crystal | | ✗ | ✓ | ✓ | ✓ | 73.2 | 84.8 | **82.3** | 85.3 | 100 |
| SFT Base (Arithmo) | Mistral-7B | – | – | – | – | 60.6 | 70.9 | 54.1 | 80.8 | – |
| Direct Tuning | | ✓ | ✗ | ✗ | ✗ | 73.9 | 85.2 | 79.3 | 86.4 | 100 |
| MCTS Offline-DPO | | ✓ | ✗ | ✗ | ✓ | 70.8 | 82.6 | 68.5 | 87.4 | 19.2 |
| Instance-level Online-DPO | | ✓ | ✓ | ✓ | ✓ | 75.3 | 87.3 | 63.1 | 87.6 | 45.6 |
| Ours | | ✓ | ✓ | ✓ | ✓ | 76.4$\uparrow_{15.8}$ | 88.2$\uparrow_{17.3}$ | 74.8$\uparrow_{19.3}$ | 88.5$\uparrow_{7.7}$ | 47.8 |

Table 2: Result comparisons (accuracy %) on commonsense reasoning tasks. The results based on GPT-3-curie (Brown et al., 2020) and T5 (Raffel et al., 2020) are reported from Liu et al. (2023b). For CSQA, we also include the GPT-J (Wang and Komatsuzaki, 2021) results reported by Zelikman et al. (2022). We follow Liu et al. (2023b) to combine the training data of ARC, AI2Sci, OBQA, and CSQA for training , while STaR (Zelikman et al., 2022) only use CSQA for training.

the other hand, we observe significant performance gain on MATH when incorporating the ground-truth solutions in the MCTS process for preference data collection, illustrating an effective way to refine the preference data quality with G.T. guidance.

**Commonsense Reasoning.** In Table 2, we report the performance on commonsense reasoning tasks, where our method shows consistent improvements. Notably, we achieve absolute accuracy increases of 2.5%, 3.0%, and 2.1% on ARC-Challenge (ARC-C), AI2Sci-Middle (AI2Sci-M), and SciQ, respectively, surpassing the results of direct tuning. However, in tasks like OBQA and CSQA, our method, focusing on intermediate reasoning refinement, is less efficient compared to direct tuning. Despite significant improvements over the Supervised Fine-Tuning (SFT) baseline (for instance, from 59.8% to 79.2% on OBQA, and from 54.1% to 74.8% on CSQA), the gains are modest relative to direct tuning.

This discrepancy could be attributed to the base model's lack of specific knowledge, where eliciting intermediate reasoning chains may introduce increased uncertainty in model generations, leading to incorrect predictions. We delve deeper into this issue of hallucination and its implications in our qualitative analysis, as detailed in Section 3.2.

### 3.2 Further Analysis

**Training- vs. Test- Time Compute Scaling.** Our method integrates MCTS with preference learning, aiming to enhance both preference quality and policy reasoning via step-level alignment. We analyze the impact of training-time compute scaling versus increased inference-time sampling.

We measure success by the pass rate, indicating the percentage of correctly elicited answers. Figure 3 displays the cumulative pass rate at each checkpoint, aggregating the pass rates up to that point. For test-time scaling, we increase the number of sampled reasoning chains. Additionally, we compare the inference performance of our checkpoints with a sampling-only method, self-consistency, to assess their potential performance ceilings. The pass rate curves on ARC-C, SciQ, and MATH datasets reveal that our MCTS-enhanced approach yields a higher training compute scaling exponent. This effect is particularly pronounced on the unseen SciQ dataset, highlighting our method's efficiency and effectiveness in enhancing specific reasoning abilities with broad applicability. Inference-time performance analysis shows higher performance upper bounds of our method on ARC-C and SciQ. For instance,

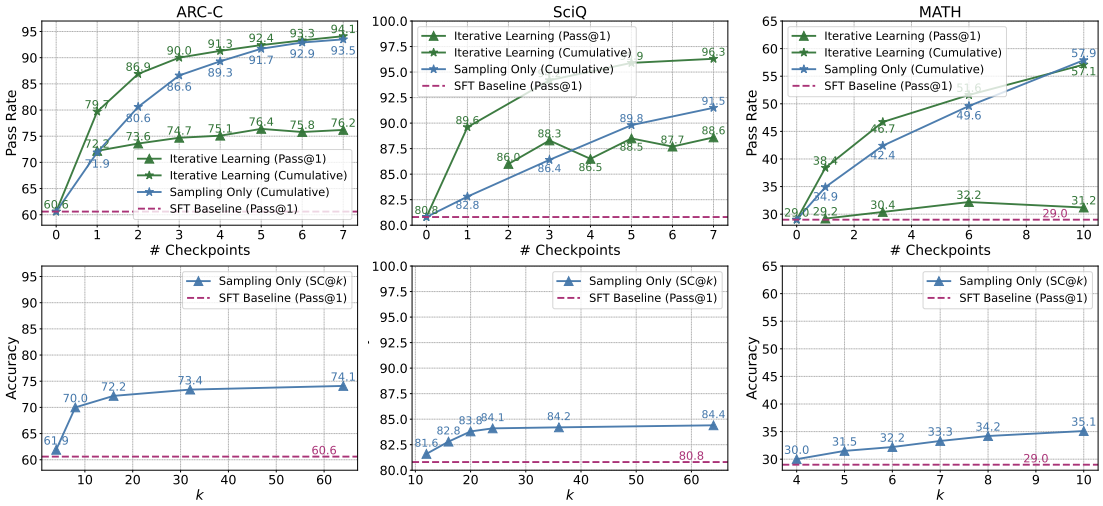


Figure 3: Training- vs. Test- Time Compute Scaling on ARC-C, SciQ, and MATH evaluation sets. The cumulative pass rate of our iterative learning method can be seen as the pass rate of an ensemble of different model checkpoints. We use greedy decoding to obtain the inference time performance of our method of iterative learning.

| Approach | GSM8K | | MATH | | ARC-C | |
|---|---|---|---|---|---|---|
| | AUC | Accuracy | AUC | Accuracy | AUC | Accuracy |
| w/ example answer | 74.7 | 72.5 | 76.6 | 48.8 | 65.2 | 57.5 |
| w/o example answer | 62.0 | 69.5 | 48.1 | 42.3 | 55.8 | 48.4 |

Table 3: Ablation of "EXAMPLE ANSWER" in self-evaluation on GSM8K, MATH, and ARC-C. We report both AUC score and accuracy (%) to compare the discriminative abilities of self-evaluation scores.

while self-consistency on SciQ plateaus at around 84%, our framework pushes performance to 88.6%. However, on MATH, the sampling-only approach outperforms training compute scaling: more sampling consistently enhances performance beyond 35%, whereas post-training performance hovers around 32.2%. This observation suggests that in-domain SFT already aligns the model well with task-specific requirements.

**Functions of Self-Evaluation Mechanism.** As illustrated in Section 2.1, the self-evaluation score inherently revises the $Q$ value estimation for subsequent preference data collection. In practice, we find that the ground-truth information, *i.e.*, the "EXAMPLE ANSWER" in Table 6, is crucial to ensure the reliability of self-evaluation. We now compare the score distribution and discriminative abilities when including v.s. excluding this ground-truth information in Table 3. With this information , the accuracy of self-evaluation significantly improves across GSM8K, MATH, and ARC-C datasets.

**Ablation Study.** We ablate the impact of step-level supervision signals and the online learning aspect of our MCTS-based approach. Tables 1 and 2 shows performance comparisons across commonsense and arithmetic reasoning tasks under different settings. Our method, focusing on step-level online preference learning, consistently outperforms both instance-level and offline approaches in commonsense reasoning. For example, we achieve 76.4% on ARC-C and 88.5% on SciQ, surpassing 70.8% and 87.4% of the offline variant, and 75.3% and 87.6% of the instance-level approach.

In arithmetic reasoning, performance differences among settings are less pronounced for challenging task such as MATH without the incorporation of ground-truth solutions (*e.g.*, 32.2% for our method (w/o G.T.) vs. 31.9% and 32.9% for offline and instance-level on MATH). The comparable performance of offline learning aligns with our theoretical analysis that offline approaches can be effective when the initial policy is already well-tuned with high-quality, in-domain data. We further interpret how G.T. guidance integration to enhance the effectiveness of our framework in Figure 4. With G.T. supervision, the accuracy of training signals improve significantly from 81.2% to 97.9%, leading to substantial performance gain on model performance. This also explains the similar performance (w/o G.T.) between corresponding using step- and instance-level supervision, where our step-level approach shows effectiveness in narrowing the gap between accuracies of corresponding supervisions.

**Training Dynamics in Iterative Learning.** As shown in Figure 2, online learning exhibits cyclic

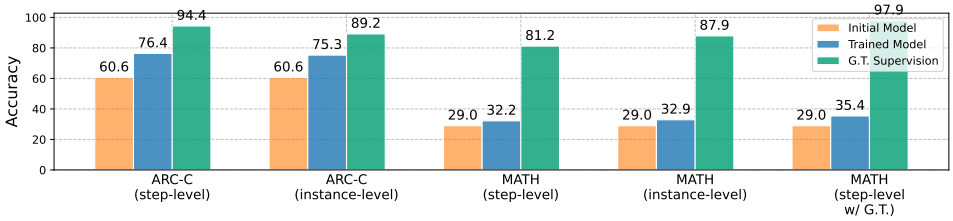


Figure 4: Abltation study on step- vs. instance-level supervision, online vs. offline learning, and w/ vs. w/o G.T. supervision. We also compare the accuracy of the training signals collected via MCTS or instance-level sampling.

| Task | Inputs and Outputs | Answer |
|---|---|---|
| OBQA | **Question**: As the rain forest is deforested the atmosphere will increase with:<br>**Answer Choices**: (A) oxygen (B) nitrogen (C) carbon (D) rain<br>**SFT Baseline**: As the rain forest is deforested the atmosphere will increase with carbon. The answer is C.<br>**Ours**: When trees are cut down, the carbon stored in the trees is released into the atmosphere as carbon dioxide. This carbon dioxide then increases the carbon content of the atmosphere. Since carbon is the main ingredient in plant life, the increased carbon content of the atmosphere will lead to increased plant growth, which will lead to increased photosynthesis, which will lead to increased oxygen production. Therefore, oxygen is the gas that will increase in the atmosphere as the rain forest is deforested. The answer is: A. | (C) |
| AI2Sci | **Question**: Which pair of instruments can be used to measure wind chill?<br>**Answer Choices**: (A) barometer and wind vane (B) wind vane and anemometer (C) thermometer and barometer (D) anemometer and thermometer<br>**SFT Baseline**: Wind chill is the effect of wind on the apparent temperature, it is the temperature that the human body feels due to the movement of air. It is an index of the discomfort felt by the human body due to low temperature and high wind speed. It is measured by an instrument called wind chill meter. The answer is B<br>**Ours**: To measure wind chill, you would need an anemometer and a thermometer. The anemometer measures the wind speed, and the thermometer measures the temperature. Together, these two instruments can be used to calculate the wind chill factor. The answer is: D | (D) |
| MATH | **Question**: Let $f(x) = x^7 + ax^3 + cx$ If $f(5) = 3$, then find $f(-5)$.<br>**SFT Baseline**: We know that $f(5) = 5^7 + 5a + 5c = 3$. We want to find $f(-5)$, which is $(-5)^7 + (-5a) + (-5c)$. Since $(-5)^7 = -5^7 = -3$, we have $f(-5) = -3 - 5a - 5c$. We can rewrite this as $f(-5) = -3 - 5(-3) - 5c = -3 + 15 - 5c = \boxed{12 - 5c}$.<br>**Ours**: We know that $f(5) = 5^7 + 5a + 5c = 3$. Since $f(x)$ is a polynomial, we can use the fact that $f(-5) = (-5)^7 - 5a - 5c = \boxed{-3}$. | -3 |

Table 4: Qualitative Analysis of model outputs on OBQA, AI2Sci, and MATH. We highlight correct and incorrect keywords. The predicted answers on MATH are boxed, where our method solves the complex problem while SFT baseline failed to elicit a complete reasoning chain.

performance fluctuations, with validation performance peaking before dipping. We conduct theoretical analysis on this in Appendix B and shows that continuous policy updates with the latest models can lead to periodic knowledge loss due to insufficient optimization in iterative updates. We further probe these phenomena qualitatively next.

**Qualitative Analysis.** Our qualitative analysis in Table 4 examines the impact of step-level supervision on intermediate reasoning correctness across different tasks. In OBQA, the implementation of MCTS, as discussed in Section 3.1, often leads to longer reasoning chains. This can introduce errors in commonsense reasoning tasks, as seen in our OBQA example, where an extended chain results in an incorrect final prediction. Conversely, in the MATH dataset, our approach successfully guides the model to rectify mistakes and formulates accurate, extended reasoning chains, demonstrating its effectiveness in complex math word problems. This analysis underscores the need to balance reasoning chain length and logical coherence, particularly in tasks with higher uncertainty, such as commonsense reasoning.

## 4 Related Work

Various studies focus on self-improvement to exploit the model's capability. One line of work focuses on collecting high-quality positive data from model generations guided by static reward heuristic (Zelikman et al., 2022; Gülçehre et al., 2023; Polu et al., 2023). Recently, Yuan et al. (2024) utilize the continuously updated LLM self-rewarding to collect both positive and negative data for preference learning. Different from prior works at instance-level alignment, we leverage MCTS as a policy improvement operator to iteratively facilitate step-level preference learning. We discuss additional related work in Appendix A.

## 5 Conclusion

In this paper, we propose MCTS-enhanced iterative preference learning, utilizing MCTS as a policy improvement operator to enhance LLM alignment via step-level preference learning. MCTS balances quality exploitation and diversity exploration to produce high-quality training data, pushing the ceiling performance of the LLM on various reasoning tasks. We hope our proposed approach can inspire future research on LLM alignment from both data-centric and algorithm-improving aspects: to explore searching strategies and utilization of history data and policies to augment and diversify training examples; to employ offline-online trade-off to address the problem of cyclic performance change of the online learning framework.

## Limitations

The main limitations of this work is the consideration regarding computational cost as the MCTS process is bottlenecked by LLM inference time. While automatic data generation and labeling via MCTS illustrates an efficient way than human preference annotation, the increasing cost and time from sampling may constrain the scalability of our framework. On the other hand, the work only focuses on the application of MCTS in the context of LLM reasoning. We leave it to future work in exploring the generalizability of our MCTS-enhanced iterative preference learning approach across more general tasks and models.

## Ethics Statement

This paper mainly focuses on improving LLM reasoning via MCTS-enhanced Iterative Preference Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted regarding the ethical concern here.

## References


Thomas Anthony, Zheng Tian, and David Barber. 2017. Thinking fast and slow with deep learning and tree search. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5360–5370.

Mohammad Gheshlaghi Azar, Mark Rowland, Bilal Piot, Daniel Guo, Daniele Calandriello, Michal Valko, and Rémi Munos. 2023. A general theoretical paradigm to understand learning from human preferences. *CoRR*, abs/2310.12036.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom B. Brown, Jack Clark, Sam McCandlish, Chris Olah, Benjamin Mann, and Jared Kaplan. 2022a. Training a helpful and harmless assistant with reinforcement learning from human feedback. *CoRR*, abs/2204.05862.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. 2022b. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*.

Ralph Allan Bradley and Milton E Terry. 1952. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4299–4307.

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. *CoRR*, abs/1803.05457.

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. *CoRR*, abs/2110.14168.

Rémi Coulom. 2006. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pages 72–83. Springer.

Jean-Bastien Grill, Florent Altché, Yunhao Tang, Thomas Hubert, Michal Valko, Ioannis Antonoglou, and Rémi Munos. 2020. Monte-carlo tree search as regularized policy optimization. In *International Conference on Machine Learning*, pages 3769–3778. PMLR.

Çaglar Gülçehre, 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. *CoRR*, abs/2308.08998.

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. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 8154–8173. Association for Computational Linguistics.

Junxian He, Jiatao Gu, Jiajun Shen, and Marc'Aurelio Ranzato. 2020. Revisiting self-training for neural sequence generation. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

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. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.

Jiaxin Huang, Shixiang Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2023. Large language models can self-improve. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 1051–1068. Association for Computational Linguistics.

H. J. Scudder III. 1965. Probability of error of some adaptive pattern-recognition machines. *IEEE Trans. Inf. Theory*, 11(3):363–371.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *CoRR*, abs/2310.06825.

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. *CoRR*, abs/2207.05221.

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

Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Lu, Thomas Mesnard, Colton Bishop, Victor Carbune, and Abhinav Rastogi. 2023. RLAIF: scaling reinforcement learning from human feedback with AI feedback. *CoRR*, abs/2309.00267.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2023. Making language models better reasoners with step-aware verifier. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5315–5333.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *CoRR*, abs/2305.20050.

Jiacheng Liu, Andrew Cohen, Ramakanth Pasunuru, Yejin Choi, Hannaneh Hajishirzi, and Asli Celikyilmaz. 2023a. Making PPO even better: Value-guided monte-carlo tree search decoding. *CoRR*, abs/2309.15028.

Jiacheng Liu, Ramakanth Pasunuru, Hannaneh Hajishirzi, Yejin Choi, and Asli Celikyilmaz. 2023b. Crystal: Introspective reasoners reinforced with self-feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 11557–11572, Singapore. Association for Computational Linguistics.

Tianqi Liu, Yao Zhao, Rishabh Joshi, Misha Khalman, Mohammad Saleh, Peter J. Liu, and Jialu Liu. 2023c. Statistical rejection sampling improves preference optimization. *CoRR*, abs/2309.06657.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? A new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2381–2391. Association for Computational Linguistics.

Eric Mitchell. 2023. A note on dpo with noisy preferences & relationship to ipo.

Long Ouyang, Jeffrey 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 F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *NeurIPS*.

Daniel S Park, Yu Zhang, Ye Jia, Wei Han, Chung-Cheng Chiu, Bo Li, Yonghui Wu, and Quoc V Le. 2020. Improved noisy student training for automatic speech recognition. *arXiv preprint arXiv:2005.09629*.

Stanislas Polu, Jesse Michael Han, Kunhao Zheng, Mantas Baksys, Igor Babuschkin, and Ilya Sutskever. 2023. Formal mathematics statement curriculum learning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *CoRR*, abs/2305.18290.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Jie Ren, Yao Zhao, Tu Vu, Peter J Liu, and Balaji Lakshminarayanan. 2023. Self-evaluation improves selective generation in large language models. *arXiv preprint arXiv:2312.09300*.

Christopher D Rosin. 2011. Multi-armed bandits with episode context. *Annals of Mathematics and Artificial Intelligence*, 61(3):203–230.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy P. Lillicrap, Karen Simonyan, and Demis Hassabis. 2017. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *CoRR*, abs/1712.01815.

Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F. Christiano. 2020. Learning to summarize from human feedback. *CoRR*, abs/2009.01325.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

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

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax.

Peiyi Wang, Lei Li, Zhihong Shao, RX Xu, Damai Dai, Yifei Li, Deli Chen, Y Wu, and Zhifang Sui. 2023a. Math-shepherd: A label-free step-by-step verifier for llms in mathematical reasoning. *arXiv preprint arXiv:2312.08935*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. 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 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text, NUT@EMNLP 2017, Copenhagen, Denmark, September 7, 2017*, pages 94–106. Association for Computational Linguistics.

Zeqiu Wu, Yushi Hu, Weijia Shi, Nouha Dziri, Alane Suhr, Prithviraj Ammanabrolu, Noah A. Smith, Mari Ostendorf, and Hannaneh Hajishirzi. 2023. Fine-grained human feedback gives better rewards for language model training. *CoRR*, abs/2306.01693.

Qizhe Xie, Minh-Thang Luong, Eduard H. Hovy, and Quoc V. Le. 2020. Self-training with noisy student improves imagenet classification. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 10684–10695. Computer Vision Foundation / IEEE.

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, Xu Zhao, Min-Yen Kan, Junxian He, and Qizhe Xie. 2023. Decomposition enhances reasoning via self-evaluation guided decoding. *CoRR*, abs/2305.00633.

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. *CoRR*, abs/2305.10601.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd Annual Meeting of the Association for Computational Linguistics, 26-30 June 1995, MIT, Cambridge, Massachusetts, USA, Proceedings*, pages 189–196. Morgan Kaufmann Publishers / ACL.

Fei Yu, Anningzhe Gao, and Benyou Wang. 2023a. Outcome-supervised verifiers for planning in mathematical reasoning. *CoRR*, abs/2311.09724.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023b. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. 2024. Self-rewarding language models. *arXiv preprint arXiv:2401.10020*.

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

Tianjun Zhang, Fangchen Liu, Justin Wong, Pieter Abbeel, and Joseph E. Gonzalez. 2023. The wisdom of hindsight makes language models better instruction followers. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 41414–41428. PMLR.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *CoRR*, abs/2306.05685.

Xinyu Zhu, Junjie Wang, Lin Zhang, Yuxiang Zhang, Yongfeng Huang, Ruyi Gan, Jiaxing Zhang, and Yujiu Yang. 2023. Solving math word problems via cooperative reasoning induced language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 4471–4485. Association for Computational Linguistics.

Barret Zoph, Golnaz Ghiasi, Tsung-Yi Lin, Yin Cui, Hanxiao Liu, Ekin Dogus Cubuk, and Quoc Le. 2020. Rethinking pre-training and self-training. *Advances in neural information processing systems*, 33:3833–3845.

## A Related Work

**Iterated Learning.** Typical iterated learning operates in a multi-agent scenario, consisting of a loop where an apprentice self-plays, learns from expert feedback, and replaces the current expert for the new iteration (Anthony et al., 2017). Polu et al. (2023) apply expert iteration on formal mathematical reasoning to conduct proof search interleaved with learning. Zelikman et al. (2022) avoid the need for training a separate value function by directly assessing the final outcomes of reasoning to filter generated examples for iterated learning. Recently, Yuan et al. (2024) leverage the technique of LLM-as-a-Judge (Zheng et al., 2023) and introduce self-rewarding language models to improve LLM alignment with self-feedback. Differently, we combine the feedback of outcome assessment and LLM self-evaluation and further decompose them into fine-grained signals via MCTS for step-level iterative preference learning.

**Self-Training.** Self-training uses unlabeled data to improve model training by assigning pseudo labels from a learned labeler (III, 1965; Yarowsky, 1995; Xie et al., 2020; He et al., 2020; Park et al., 2020; Zoph et al., 2020). Recent research has explored several alternatives to label the examples. Zelikman et al. (2022) and Gülçehre et al. (2023) use static reward heuristic to curate high-quality examples, while Huang et al. (2023) collect high-confidence outputs as training data via chain-of-thought prompting (Wei et al., 2022) and self-consistency (Wang et al., 2023b). Lee et al. (2023) and Yuan et al. (2024) utilize the off-the-shelf LLM to reward its generations for preference learning. To mitigate the noise from the sparse instance-level signals, we further refine the preference labels via stepwise tree search and LLM self-evaluation.

**Preference Learning.** The empirical achievements of LLMs have identified the benefits of RL techniques to better align with human preferences (Touvron et al., 2023; Stiennon et al., 2020; Ouyang et al., 2022; Bai et al., 2022a). The standard preference learning process learns a reward model to provide feedback in online RL (Schulman et al., 2017). Recently, a variety of studies avoid training separate reward or value networks by hindsight instruction relabeling (Zhang et al., 2023), direct preference optimization (Rafailov et al., 2023) and LLM self-evaluation (Ren et al., 2023). We further explore automatic supervision with MCTS to collect step-level preferences by breaking down outcome correctness integrated with self-evaluation. Our approach enables the continual collection of better-quality data via iterative learning, mitigating the limit of preference data when using a frozen reward model or offline learning algorithms.

**Guided Search for Reasoning.** Recent works improve the LLM reasoning ability by eliciting the intermediate reasoning chain (Wei et al., 2022) and breaking it down into multiple steps via searching (Yao et al., 2023; Hao et al., 2023; Yu et al., 2023a). The decomposition of the reasoning process has also been shown effective in reinforcement learning. Lightman et al. (2023) and Li et al. (2023) apply process supervision to train more reliable reward models than outcome supervision in mathematical reasoning (Uesato et al., 2022). Wang et al. (2023a) reinforce LLMs step-by-step with process supervision data automatically collected via model sampling and annotation. We leverage the look-ahead ability of MCTS and integrate it with step-by-step self-evaluation to provide refined process supervision for reasoning. This improves the generalization ability of our framework to update the policy via real-time collected preferences iteratively.

## B Theoretical Analysis of Online DPO

Our approach can be viewed as an online version of DPO, where we iteratively use the updated policy to sample preferences via MCTS. In this section, we provide theoretical analysis to interpret the advantages of our online learning framework compared to the conventional alignment techniques that critically depend on offline preference data. We review the typical RLHF and DPO paradigms in Appendix B.

**Preliminaries.** A typical alignment technique begins with a policy $\pi_{\text{sft}}(y \mid x)$ supervisedly fine-tuned on high-quality data from the target domain, where $x$ and $y$ are the prompt and the response, respectively.

The SFT policy is used to sample pairs of responses $(y_1, y_2) \sim \pi_{\text{sft}}(y \mid x)$ with prompts $x$, which will be further labeled as pairwise preference data $y_w \succ y_l \mid x$, where $y_w$ and $y_l$ represent the preferred and dispreferred responses respectively. 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 $\pi_\theta$ with the feedback provided by the reward model, where $\pi_\theta$ is also initialized to $\pi_{\text{sft}}$ in practice. DPO avoids fitting a reward model by optimizing the policy $\pi_\theta$ using preferences directly.

Given a reward function $r(x, y)$ and prompt distribution $\mathcal{P}$, RLHF and DPO optimize the KL-constrained reward maximization objective as follows:

$$\max_{\pi} \mathbb{E}_{x\sim\mathcal{P},y\sim\pi}[r(x,y)] - \beta\mathbb{D}_{\text{KL}}[\pi(y \mid x) \,\|\, \pi_{\text{sft}}(y \mid x)] \tag{9}$$

where $\beta$ scales the strength of the KL constraint. Let the ground-truth reward function be $r^*$, then Rafailov et al. (2023) estimate the optimal policy $\pi^*$ by fitting the Bradley-Terry model (Bradley and Terry, 1952) on preference data:

$$\begin{aligned} &p^*(y_1 \succ y_1 \mid x) = \sigma(r^*(x, y_1) - r^*(x, y_2)) \\ =&\frac{1}{1 + \exp\left(\beta \log \frac{\pi^*(y_2|x)}{\pi_{\text{sft}}(y_2|x)} - \beta \log \frac{\pi^*(y_1|x)}{\pi_{\text{sft}}(y_1|x)}\right)} \end{aligned} \tag{10}$$

As the maximum likelihood estimator (MLE) of the optimal policy requires preferences sampled from the target policy (Liu et al., 2023c), DPO uses a fixed, potentially optimal but unknown policy to collect preference data of good quality. This discrepancy can be a problem when the sampling policy differs dramatically from the current policy. Moreover, the absence of a reward model in DPO presents challenges in learning from additional policy-generated data that lacks explicit preference indicators.

We now consider the following abstract formulation for clean theoretical insights to analyze our online setting of preference learning. Given a prompt $x$, there exist $n$ possible suboptimal responses $\{\bar{y}_1, \ldots, \bar{y}_n\} = Y$ and an optimal outcome $y^*$. As specified in Equation 7, at the $i$-th iteration, a pair of responses $(y, y')$ are sampled from some sampling policy $\pi^{(i)}$ without replacement so that $y \neq y'$ as $y \sim \pi^{(i)}(\cdot \mid x)$ and $y' \sim \pi^{(i)}(\cdot \mid x, y)$. Then, these are labeled to be $y_w$ and $y_l$ according to the preference. Define $\Theta$ be a set of all global optimizers of the preference loss for all $M$ iterations, *i.e.*, for any $\theta \in \Theta$, $\ell_i(\theta) = 0$ for all $i \in \{1, 2, \cdots, M\}$. Similarly, let $\theta^{(i)}$ be a parameter vector such that $\ell_j(\theta^{(i)}) = 0$ for all $j \in \{1, 2, \cdots, i-1\}$ for $i \geq 1$ whereas $\theta^{(0)}$ is the initial parameter vector.

This abstract formulation covers both the offline and online settings. The offline setting in previous works is obtained by setting $\pi^{(i)} = \pi$ for some fixed distribution $\pi$. The online setting is obtained by setting $\pi^{(i)} = \pi_{\theta^{(i-1)}}$ where $\pi_{\theta^{(i-1)}}$ is the latest policy at beginning of the $i$-th iteration.

The following theorem shows that the offline setting can fail with high probability if the sampling policy $\pi^{(i)}$ differs too much from the current policy $\pi_{\theta^{(i-1)}}$:

**Theorem B.1** (Offline setting can fail with high probability)**.** *Let $\pi$ be any distribution for which there exists $\bar{y} \in Y$ such that $\pi(\bar{y} \mid x), \pi(\bar{y} \mid x, y) \leq \epsilon$ for all $y \in (Y \setminus \bar{y}) \cup \{y^*\}$ and $\pi_{\theta^{(i-1)}}(\bar{y} \mid x) \geq c$ for some $i \in \{1, 2, \cdots, M\}$. Set $\pi^{(i)} = \pi$ for all $i \in \{1, 2, \cdots, M\}$. Then, there exists $\theta \in \Theta$ such that with probability at least $1 - 2\epsilon M$ (over the samples of $\pi^{(i)} = \pi$), the following holds: $\pi_\theta(y^* \mid x) \leq 1 - c$.*

If the current policy and the sampling policy differ too much, it is possible that $\epsilon = 0$ and $c \approx 1.0$, for which Theorem B.1 can conclude $\pi_\theta(y^* \mid x) \approx 0$ with probability 1 for any number of steps $M$. When $\epsilon \neq 0$, the lower bound of the failure probability decreases towards zero as we increase $M$. Thus, it is important to make sure that $\epsilon \neq 0$ and $\epsilon$ is not too low. This is achieved by using the online setting, *i.e.*, $\pi^{(i)} = \pi_{\theta^{(i)}}$. Therefore, Theorem B.1 motivates us to use the online setting. Theorem B.2 confirms that we can indeed avoid this failure case in the online setting.

**Theorem B.2** (Online setting can avoid offline failure case)**.** *Let $\pi^{(i)} = \pi_{\theta^{(i-1)}}$. Then, for any $\theta \in \Theta$, it holds that $\pi_\theta(y^* \mid x) = 1$ if $M \geq n + 1$.*

See Appendix B for the proofs of Theorems B.1 and B.2. As suggested by the theorems, a better sampling policy is to use both the latest policy and the optimal policy for preference sampling. However, since we cannot access the optimal policy $\pi^*$ in practice, we adopt online DPO via sampling from the

latest policy $\pi_{\theta^{(i-1)}}$. The key insight of our iterative preference learning approach is that online DPO is proven to enable us to converge to an optimal policy even if it is inaccessible to sample outputs. We provide further discussion and additional insights in Appendix B.

**Additional details on labeling outcomes.** After a pair of outcomes $(y^{(i)}, y'^{(i)})$ are sampled from some sampling policy $\pi^{(i)}$, these are labeled to be $y_w^{(i)}$ and $y_l^{(i)}$ according to some preference density $p$. That is, $\Pr[(y_w^{(i)}, y_l^{(i)}) = (y^{(i)}, y'^{(i)})] = p(y^{(i)} \succ y'^{(i)} \mid x)$ and $\Pr[(y_w^{(i)}, y_l^{(i)}) = (y'^{(i)}, y^{(i)})] = 1 - p(y^{(i)} \succ y'^{(i)} \mid x)$. For simplicity, a preference density is set to be $p(y^* \succ \bar{y} \mid x) = 1$ for every optima-suboptimal pairs $(y^*, \bar{y})$ for all $\bar{y} \in Y$. We do not specify the preference density for other pairs, *i.e.*, $p(\bar{y} \succ \bar{y}' \mid x)$ is arbitrary for $(\bar{y}, \bar{y}') \in Y \times Y$.

**Abstract formulation for both offline and online settings.** Our abstract formulation covers both the offline and online settings. The offline setting in previous papers is obtained by setting $\pi^{(i)}$ to be a single distribution fixed over $i \in \{1, 2, \cdots, M\}$, *e.g.*, an initial policy, an optimal policy, or an empirical data distribution of a given preference data. In the case of the empirical data distribution, the preference density $p$ is set to the function outputting only 0 or 1 to recover the given preference data. The online setting is obtained by setting $\pi^{(i)} = \pi_{\theta^{(i-1)}}$ where $\pi_{\theta^{(i-1)}}$ is the latest policy at the beginning of the $i$-th iteration, i.e., for $i \geq 1$, $\theta^{(i)}$ satisfies $\ell_j(\theta^{(i)}) = 0$ for $j \in \{1, 2, \cdots, i-1\}$ and $\theta^{(0)}$ is the initialization. Thus, we can analyze both offline and online settings with this abstract framework.

**Proof of Theorem B.1.**

*Proof.* The intuition behind the proof of Theorem B.1 is that the current policy $\pi_{\theta^{(i)}}$ may not be corrected if a fixed sampling policy $\pi$ never samples a suboptimal output $\bar{y} \in Y$ whose probability is high for the current policy $\pi_{\theta^{(i)}}$. Let $\bar{y}$ be the suboptimal output such that $\pi(\bar{y} \mid x) \leq \epsilon$ and $\pi_{\theta^{(i)}}(\bar{y} \mid x) \geq c$ for some $i \in \{1, 2, \cdots, M\}$. Denote preferences sampled by policy $\pi^{(i)}$ as $(y_w^{(i)}, y_l^{(i)})$. From the definition of the logistic function, we can rewrite

$$\begin{aligned}
\ell_i(\theta) &= -\log \sigma \left( \beta \log \frac{\pi_\theta(y_w^{(i)} \mid x)}{\pi_{\text{ref}}(y_w^{(i)} \mid x)} - \beta \log \frac{\pi_\theta(y_l^{(i)} \mid x)}{\pi_{\text{ref}}(y_l^{(i)} \mid x)} \right) \\
&= -\log \frac{1}{1 + \exp(\beta \log \frac{\pi_\theta(y_l^{(i)}|x)}{\pi_{\text{ref}}(y_l^{(i)}|x)} - \beta \log \frac{\pi_\theta(y_w^{(i)}|x)}{\pi_{\text{ref}}(y_w^{(i)}|x)})} \\
&= -\log \frac{\exp(\beta \log \frac{\pi_\theta(y_w^{(i)}|x)}{\pi_{\text{ref}}(y_w^{(i)}|x)})}{\exp(\beta \log \frac{\pi_\theta(y_w^{(i)}|x)}{\pi_{\text{ref}}(y_w^{(i)}|x)}) + \exp(\beta \log \frac{\pi_\theta(y_l^{(i)}|x)}{\pi_{\text{ref}}(y_l^{(i)}|x)})} \\
&= -\log \frac{\frac{\pi_\theta(y_w^{(i)}|x)^\beta}{\pi_{\text{ref}}(y_w^{(i)}|x)^\beta}}{\frac{\pi_\theta(y_w^{(i)}|x)^\beta}{\pi_{\text{ref}}(y_w^{(i)}|x)^\beta} + \frac{\pi_\theta(y_l^{(i)}|x)^\beta}{\pi_{\text{ref}}(y_l^{(i)}|x)^\beta}} \\
&= -\log \frac{\pi_\theta(y_w^{(i)} \mid x)^\beta}{\pi_\theta(y_w^{(i)} \mid x)^\beta + \pi_\theta(y_l^{(i)} \mid x)^\beta (\frac{\pi_{\text{ref}}(y_w^{(i)}|x)}{\pi_{\text{ref}}(y_l^{(i)}|x)})^\beta}.
\end{aligned}$$

From this equation, we observe that $\ell_i(\theta)$ can be minimized to be zero by minimizing $\pi_\theta(y_l^{(i)} \mid x)$ to be zero without maximizing $\pi_\theta(y_w^{(i)} \mid x)$. That is, for any $\beta > 0$, if $\pi_\theta(y_l^{(i)} \mid x) = 0$,

$$\ell_i(\theta) = -\log \frac{\pi_\theta(y_w^{(i)} \mid x)^\beta}{\pi_\theta(y_w^{(i)} \mid x)^\beta + 0} = -\log 1 = 0.$$

Thus, even if we sample $y^*$ with the optimal policy, $\ell_i(\theta)$ can be minimized without maximizing $\pi_\theta(y^* \mid x)$ and minimizing $\pi_\theta(\bar{y}|x)$ for $\bar{y} \neq y_l^{(i)}$. Thus, if $\bar{y} \neq y_l^{(i)}$ for all $i \in \{1, 2, \cdots, M\}$, there exists $\theta$ such that

$\ell_i(\theta) \leq 0$ for all $i = 1, \ldots, M$, and

$$\pi_\theta(\bar{y} \mid x) \geq c,$$

because of the condition that $\pi_\theta(\bar{y} \mid x) \geq c$ for some $i \in \{1, 2, \cdots, M\}$: i.e., $\pi_\theta(\bar{y} \mid x)$ is never minimized from the $i$-th iteration while minimizing $\ell_i(\theta)$ arbitrarily well, if $\bar{y}$ is never sampled.

Therefore, if $\bar{y}$ is never sampled over $m$ iterations, since the probabilities sums up to one, we have

$$\pi_\theta(y^* \mid x) \leq 1 - \pi_\theta(\bar{y}|x) \leq 1 - c.$$

Moreover,

$$\Pr[\ \bar{y} \text{ being never sampled over } m \text{ iterations }] \geq (1 - 2\epsilon)^m \geq 1 - 2\epsilon m,$$

where the last line follows Bernoulli's inequality. By combining the above two equations, it holds that

$$\Pr[\pi_\theta(y^*|x) \leq 1 - c] \geq 1 - 2\epsilon M.$$

□

**Proof of Theorem B.2.**

*Proof.* From the proof of Theorem B.1, we have

$$\ell_i(\theta) = -\log \frac{\pi_\theta(y_w^{(i)} \mid x)^\beta}{\pi_\theta(y_w^{(i)} \mid x)^\beta + \pi_\theta(y_l^{(i)} \mid x)^\beta (\frac{\pi_{\text{ref}}(y_w^{(i)}|x)}{\pi_{\text{ref}}(y_l^{(i)}|x)})^\beta}.$$

For $\alpha \geq 0$ and $\beta > 0$, the condition $\ell_i(\theta) \leq \alpha$ implies that

$$\begin{aligned}
&-\log \frac{\pi_\theta(y_w^{(i)} \mid x)^\beta}{\pi_\theta(y_w^{(i)} \mid x)^\beta + \pi_\theta(y_l^{(i)} \mid x)^\beta (\frac{\pi_{\text{ref}}(y_w^{(i)}|x)}{\pi_{\text{ref}}(y_l^{(i)}|x)})^\beta} \leq \alpha \\
&\iff \frac{\pi_\theta(y_w^{(i)} \mid x)^\beta}{\pi_\theta(y_w^{(i)} \mid x)^\beta + \pi_\theta(y_l^{(i)} \mid x)^\beta (\frac{\pi_{\text{ref}}(y_w^{(i)}|x)}{\pi_{\text{ref}}(y_l^{(i)}|x)})^\beta} \geq \exp(-\alpha) \\
&\iff \pi_\theta(y_w^{(i)} \mid x)^\beta \geq \exp(-\alpha)\pi_\theta(y_w^{(i)} \mid x)^\beta + \exp(-\alpha)\pi_\theta(y_l^{(i)} \mid x)^\beta \left(\frac{\pi_{\text{ref}}(y_w^{(i)} \mid x)}{\pi_{\text{ref}}(y_l^{(i)}|x)}\right)^\beta \\
&\iff \pi_\theta(y_w^{(i)} \mid x)^\beta [1 - \exp(-\alpha)] \geq \pi_\theta(y_l^{(i)} \mid x)^\beta \exp(-\alpha) \left(\frac{\pi_{\text{ref}}(y_w^{(i)} \mid x)}{\pi_{\text{ref}}(y_l^{(i)} \mid x)}\right)^\beta \\
&\iff \pi_\theta(y_w^{(i)} \mid x)^\beta [1 - \exp(-\alpha)] \geq \pi_\theta(y_l^{(i)} \mid x)^\beta \exp(-\alpha) \left(\frac{\pi_{\text{ref}}(y_w^{(i)} \mid x)}{\pi_{\text{ref}}(y_l^{(i)} \mid x)}\right)^\beta \\
&\iff \pi_\theta(y_w^{(i)} \mid x)(\exp(\alpha) - 1)^{1/\beta} \left(\frac{\pi_{\text{ref}}(y_l^{(i)} \mid x)}{\pi_{\text{ref}}(y_w^{(i)} \mid x)}\right) \geq \pi_\theta(y_l^{(i)} \mid x).
\end{aligned}$$

Since $\pi_\theta(y_w^{(i)} \mid x) \leq 1$, this implies that

$$\begin{aligned}
\pi_\theta(y_l^{(i)} \mid x) &\leq \pi_\theta(y_w^{(i)} \mid x)(\exp(\alpha) - 1)^{1/\beta} \frac{\pi_{\text{ref}}(y_l^{(i)} \mid x)}{\pi_{\text{ref}}(y_w^{(i)} \mid x)} \\
&\leq (\exp(\alpha) - 1)^{1/\beta} \frac{\pi_{\text{ref}}(y_l^{(i)} \mid x)}{\pi_{\text{ref}}(y_w^{(i)} \mid x)}.
\end{aligned}$$

Thus, while we can prove a similar statement for $\alpha > 0$ with this equation, we set $\alpha = 0$ for this theorem for a cleaner insight, yielding the following: the condition $\ell_i(\theta) \leq 0$ implies that

$$\pi_\theta(y_l^{(i)} \mid x) = 0.$$

Since $y^{(i)}$ and $y'^{(i)}$ are sampled from $\pi_{\theta^{(i)}}$ without replacement, this means that we have $\pi_{\theta^{(i+k)}}(y_l^{(i)} \mid x) = 0$ for all $k \geq 1$ from the definition of $\pi_{\theta^{(i)}}$: i.e., $\pi_{\theta^{(i)}}$ is the policy such that $\ell_j(\theta^{(i)}) = 0$ for all $j = 1, \ldots, i-1$. Since $\pi_{\theta^{(i+k)}}$ is then used to sample $y^{(i)}$ and $y'^{(i)}$ in the followings iterations for $k \geq 1$, we will never sample this $y_l^{(i)}$ again. Thus, at each iteration, we always sample pairs of $y$ and $y'$ such that these do not include an output judged to be not preferred in a previous iteration. This implies that at each iteration, we increase the number of suboptimal samples $\bar{y} \in Y$ such that $\pi_{\theta^{(i)}}(\bar{y} \mid x) = 0$. In other words, we have

$$|\{\bar{y} \in Y \mid \pi_{\theta^{(i)}}(\bar{y} \mid x) = 0\}| \geq i - 1.$$

Thus,

$$\pi_{\theta^{(i)}}(y^* \mid x) = 1 - \sum_{j=1}^{n} \pi_{\theta^{(i)}}(\bar{y}_j \mid x) = 1 - \sum_{j \in S} \pi_{\theta^{(i)}}(\bar{y}_j \mid x).$$

where $|S| \leq n + 1 - i$. Therefore, $\pi_{\theta^{(i)}}(y^* \mid x) = 1$ when $i \geq n + 1$.

□

**Additional discussion.** We list the additional insights gained from the theoritical analysis.

- The proofs of Theorems B.1–B.2 suggest that a better sampling policy is to use both the current policy and the optimal policy at the same time in the preference learning loss, i.e., sample $y \sim \pi^*$ and $y' \sim \pi_{\theta^{(i-1)}}$. This avoids the failure case of Theorem B.1 and improves the convergence speed in Theorem B.2. However, since we cannot access the optimal policy $\pi^*$ in practice, Theorems B.1–B.2 motivate online DPO. Online DPO is proven to enable us to converge to an optimal policy even if we cannot sample outputs from the optimal policy.

- The proofs of Theorems B.1–B.2 suggest that if we can sample from the optimal policy, then we can also use the samples of the optimal policy with the negative log-likelihood loss $-\log \pi_\theta(y^* \mid x)$ instead of DPO loss to avoid the failure case.

- The proofs of Theorems B.1–B.2 suggest that in the online setting, we should minimize the DPO loss to a certain low degree per iteration, i.e., we should take several rounds of minimization of DPO loss per online iteration, instead of only taking one round of minimization per iteration. This is because the proofs of Theorems B.1–B.2 show that we can get into the cyclic situation in the online setting if the DPO loss is not minimized sufficiently per iteration. For example, we can sample $\bar{y}_1$ and $\bar{y}_2$ in one iteration and $\bar{y}_2$ and $\bar{y}_3$ in another iteration where $\bar{y}_1 \succ \bar{y}_2 \succ \bar{y}_3$. If the probability of sampling $\bar{y}_2$ is not minimized sufficiently in the first iteration, it can be sampled again in the second iteration, where the probability of sampling $\bar{y}_2$ can be increased as $\bar{y}_2 \succ \bar{y}_3$. Then, this can repeat indefinitely. Thus, it is important to minimize DPO loss with several optimizer iterations per iteration.

## C Implementation Details

We use Mistral-7B as our base pre-trained model. The supervised fine-tuning and preference learning experiments are conducted with a maximum of $4 \times 40$GB GPUs (NVIDIA A100).

We choose the learning rates 5e-6 and 1e-6 for SFT and DPO training, respectively, with a cosine learning rate scheduler. The maximum sequence length of models is $512$. We train the model with a batch size of $128$ and $32$ for SFT and DPO, respectively. For DPO, we follow the DPO paper to set the KL constraint parameter $\beta$ as $0.1$. Each sample in DPO is a set of step-level preference data decomposed by MCTS. We set the max length for each step as $64$. The number of MCTS iterations is set as $K = 5$ for all tasks.

For arithmetic reasoning, we combine the problems in GSM8K and MATH training sets as the prompt data containing a total of 24K samples for preference learning. For each sample, we conduct MCTS with an initial breadth of $b_1 = 5$ and decrease it to $b_2 = 3$ for the subsequent steps, with a maximum search depth $d = 4$. It takes about 2 minutes per sample to collect the step-level preferences via MCTS. This requires about 30 A100 days of compute to train one whole epoch. In practice, we can adopt an early stop when the performance saturates, which usually only needs $30\%$ of the training data.

For commonsense reasoning, we combine the training data of ARC, AI2Science, OBQA, and CSQA, which produces a total of 12K samples. As the model generations are more diversified on these tasks, we set the initial breadth as $b_1 = 4$ and decrease it to $b_2 = 2$ for subsequent steps. As the intermediate reasoning chains are relatively shorter than those in arithmetic reasoning, we set the maximum search depth $d = 3$. Likewise, we also adopt an early stop at around $50\%$ of the training progress where the performance saturates.

**Hyperparameter Tuning of MCTS.** We compare the performance in commonsense reasoning when employing different searching breadths in MCTS. Table 5 shows how different search heuristics impact learning performance. O2 produces better performance, highlighting the importance of increasing the search space at the beginning point of MCTS. One can efficiently reduce compute while maintaining good performance by using a small search space for the subsequent steps. For future work, we will explore the hyperparameter settings in MCTS, including the search breadth, depth, number of steps, and iteration time, to probe the cost–performance tradeoff of our MCTS-enhanced iterative learning framework.

| APPROACH | ARC-E | ARC-C | AI2SCI-E | AI2SCI-M | OBQA | CSQA | SCIQ |
|---|---|---|---|---|---|---|---|
| SFT BASELINE | 69.2 | 60.6 | 74.9 | 70.9 | 59.8 | 54.1 | 80.8 |
| O1 ($b_1 = 3, b_2 = 3$) | 88.4 | 74.7 | 92.1 | 88.5 | 77.8 | 73.2 | 88.3 |
| O2 ($b_1 = 4, b_2 = 2$) | 88.5 | 76.4 | 91.7 | 88.2 | 79.2 | 74.8 | 88.5 |

Table 5: Result comparison of using different search breadths in MCTS. For O2, we have a broader spectrum for the initial step and narrow the search space for the subsequent steps of each path.

**Prompt Example.** See an example of the evaluation prompt we use for self-evalution in Table 6. For more details, please refer to our implementation code.

**QUESTION:** Which of the following is an example of the formation of a mixture? Answer Choices: (A) rust forming on an iron nail (B) sugar crystals dissolving in water (C) sodium and chlorine forming table salt (D) hydrogen and oxygen reacting to produce water

**EXAMPLE ANSWER:** The answer is (B) sugar crystals dissolving in water

**PROPOSED SOLUTION:** The formation of a mixture occurs when two or more substances are combined together without changing their individual properties. In the given options, rust forming on an iron nail is an example of the formation of a mixture. The iron nail and the oxygen in the air combine to form iron oxide, which is a mixture. The answer is A.

**QUESTION**: Evaluate if the proposed solution is logically heading in the correct direction.
Provide an answer of (A) correct or (B) incorrect.

**ANSWER**: The answer is

Table 6: Evaluation Prompt Template. The text underlined will be replaced with content from different examples.

## D Further Analysis

**Reward Criteria in MCTS.** We probe the effect of different reward guidance of MCTS in terms of both searching and training. Table 7 shows how different reward signals impact the pass rate of searching. The guidance of outcome correctness is substantially dominant in eliciting correct outcomes. We see that MCTS can produce significant improvement across various tasks with the reward signals integrated of outcome correctness and self-evaluation, increasing the baseline performance from $60.6\%$ to $83.0\%$ on ARC-C, $70.9\%$ to $90.5\%$ on AI2Sci-M, and $75.9\%$ to $85.8\%$ on GSM8K. We observe a significant performance gain from learning when using greedy decoding on commonsense reasoning. For example, learning increases the accuracy to $76.4\%$ (+$16.4\%$) on ARC-C, compared to the increase of $9.1\%$ on MCTS performance. This suggests a substantial improvement in the model's policy when applying our

MCTS-enhanced iterative learning to tasks that the initial policy is not good at. Furthermore, the ablation study on the reward components shows consistent improvement brought by self-evaluation to increase the MCTS performance in both before- and after- learning cases, suggesting the effectiveness of the integration of self-evaluation in our approach.

| Decoding Strategy | After Learning | ARC-C | AI2Sci-M | GSM8K |
|---|---|---|---|---|
| Greedy Decoding | ✗ | 60.6 | 70.9 | 75.9 |
| | ✓ | 76.4$\uparrow_{16.4}$ | 88.2$\uparrow_{17.3}$ | 80.7$\uparrow_{5.2}$ |
| MCTS w/o SE | ✗ | 82.5 | 87.3 | 84.4 |
| | ✓ | 91.0$\uparrow_{8.5}$ | 96.1$\uparrow_{9.8}$ | 89.0$\uparrow_{5.6}$ |
| MCTS | ✗ | 83.0 | 90.5 | 85.8 |
| | ✓ | 92.1$\uparrow_{9.1}$ | 97.3$\uparrow_{6.8}$ | 90.2$\uparrow_{4.4}$ |

Table 7: Pass Rates when Ablating MCTS Settings. SE represents the guidance from self-evaluation.

**Qualitative Analysis on Collected Preferences.** We show examples of the result search trees elicited via MCTS on different tasks in Figures 5–9.

Figures 5 and 6 show the result search trees to answer the same science question using MCTS employed with different search breadths. We see that MCTS not only figures out the correct answer (*i.e.*, the option "D") via broad searching but also serves as a policy improvement optimizer to collect steps along this path as positive samples for preference learning. For example, the $Q$ values of the preference pair at the last step (at the bottom right of Figure 5) are $0.70838$ and $-0.45433$, compared to the original probability in the policy generation as $0.37989$ and $0.38789$. Compared to searching with breadth $b_1 = 4, b_2 = 2$ in Figure 5, Figure 6 shows that a higher breadth for the subsequent steps can produce an even larger search tree. However, as we only collect preference pairs alongside the paths leading to correct prediction, these two search heuristics can result in preference data of similar size.

Figure 7 shows the search tree using the trained policy on commonsense reasoning. Compared to the one generated by the initial policy in Figure 5, the policy has a higher chance to elicit correct reasoning chains, as we see more successful predictions of the ground-truth option "D". We also observe that the policy tends to generate longer reasoning chains after being motivated to conduct chain-of-thought reasoning with fine-grained process supervision.

On arithmetic reasoning, we also probe the impact of diversity in model generations using policies trained for different numbers of epochs in SFT. Figures 8 and 9 show the elicited search trees with data sampled by policies corresponding to different levels of diversity, where the policy used in Figure 8 has generations with higher diversity. With higher diversity, MCTS can explore more alternatives of the correct solutions, as there are more paths of correct predictions, as shown in Figure 8 than Figure 9. Furthermore, higher diversity with reasonable quality also provide more fine-grained supervision signals as there are more branches alongside the reasoning path of correct predictions.

## E Extended Experiments

**Loss Function.** DPO is one of the reward-model-free loss functions we can use for preference learning. We now illustrate the generalizability of our approach using another loss function, Identity Preference Optimization (IPO) (Azar et al., 2023), which addresses the overfitting problem of DPO. Table 8 shows that IPO achieves similar performance as DPO. In practice, we find that IPO boosts the reasoning on validation tasks while maintaining a more stable performance on the held-out dataset, as indicated by the higher accuracy $89.8\%$ obtained on SciQ.

**Base Model.** We extensively validate the generalizability of our approach on Llama2-13B (Touvron et al., 2023) on arithmetic reasoning. We employ the same process of SFT on Arithmo and preference learning with DPO on GSM8K and MATH. This experiment is done on a maximum of $2 \times 80$GB GPUs (NVIDIA A100).

| APPROACH | ARC-E | ARC-C | AI2SCI-E | AI2SCI-M | OBQA | CSQA | SCIQ |
|---|---|---|---|---|---|---|---|
| SFT BASELINE | 69.2 | 60.6 | 74.9 | 70.9 | 59.8 | 54.1 | 80.8 |
| O1 (IPO) | 88.1 | 75.1 | 92.1 | 89.6 | 76.8 | 74.3 | 89.8 |
| O2 (DPO) | 88.5 | 76.4 | 91.7 | 88.2 | 79.2 | 74.8 | 88.5 |

Table 8: Result comparison of employing our approach with different loss functions.

| APPROACH | BASE MODEL | GSM8K-COT | GSM8K-POT | MATH-COT |
|---|---|---|---|---|
| SFT (ARITHMO) | LLAMA2-13B | 74.5 | 62.3 | 23.8 |
| OURS | | 78.9$\uparrow_{4.4}$ | 67.0$\uparrow_{4.7}$ | 26.1$\uparrow_{2.3}$ |

Table 9: Result comparison (accuracy %) for Llama2-13B on arithmetic tasks.

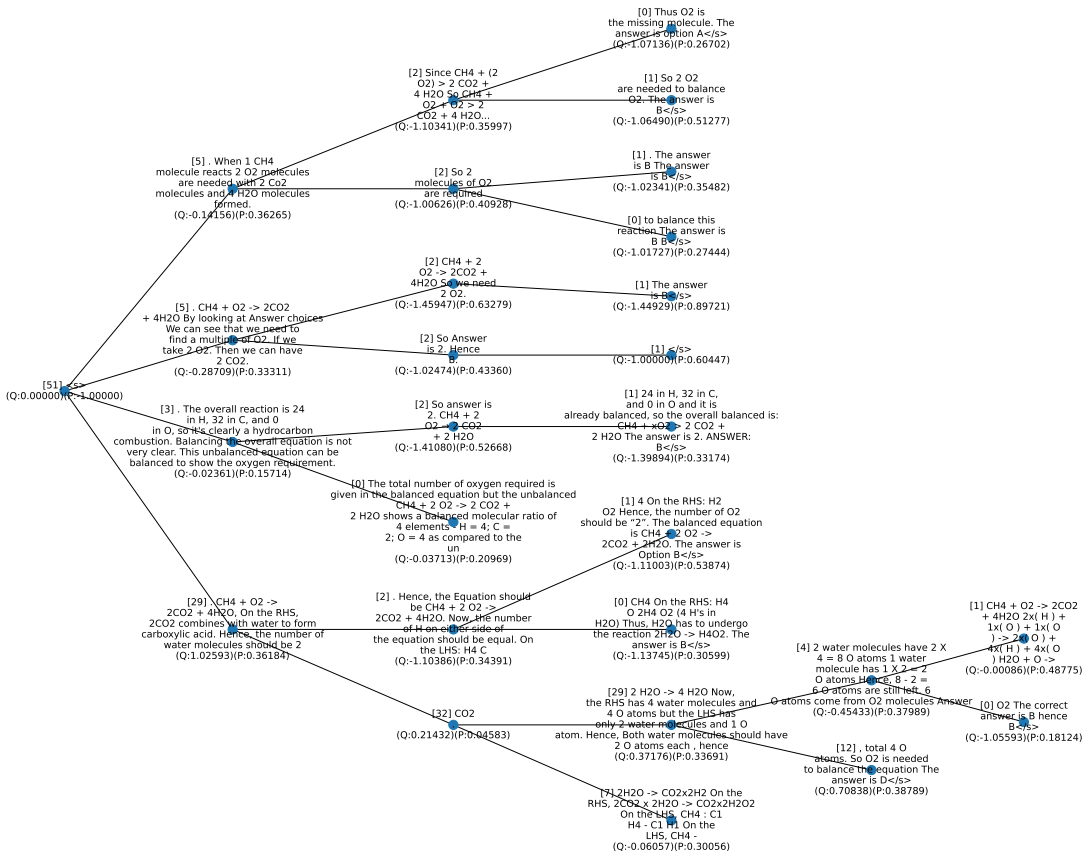


Figure 5: Example of the result search tree of a science question "An unbalanced equation for the reaction of methane gas ($CH_4$) with oxygen is shown below. $CH_4 + \square O_2 \rightarrow 2CO_2 + 4H_2O$ How many molecules of oxygen gas ($O_2$) are needed to properly balance this equation? Answer Choices: (A) 1 (B) 2 (C) 3 (D) 4". The ground truth answer is "(D) 4". Here, we set the search breadth as $b_1 = 4, b_2 = 2$. The numbers at the beginning of each sequence indicate the visit count $N$ of the corresponding node, while the $Q$ and $P$ values at the end of the sequence represent the $Q$ values and the sentence probability, respectively.

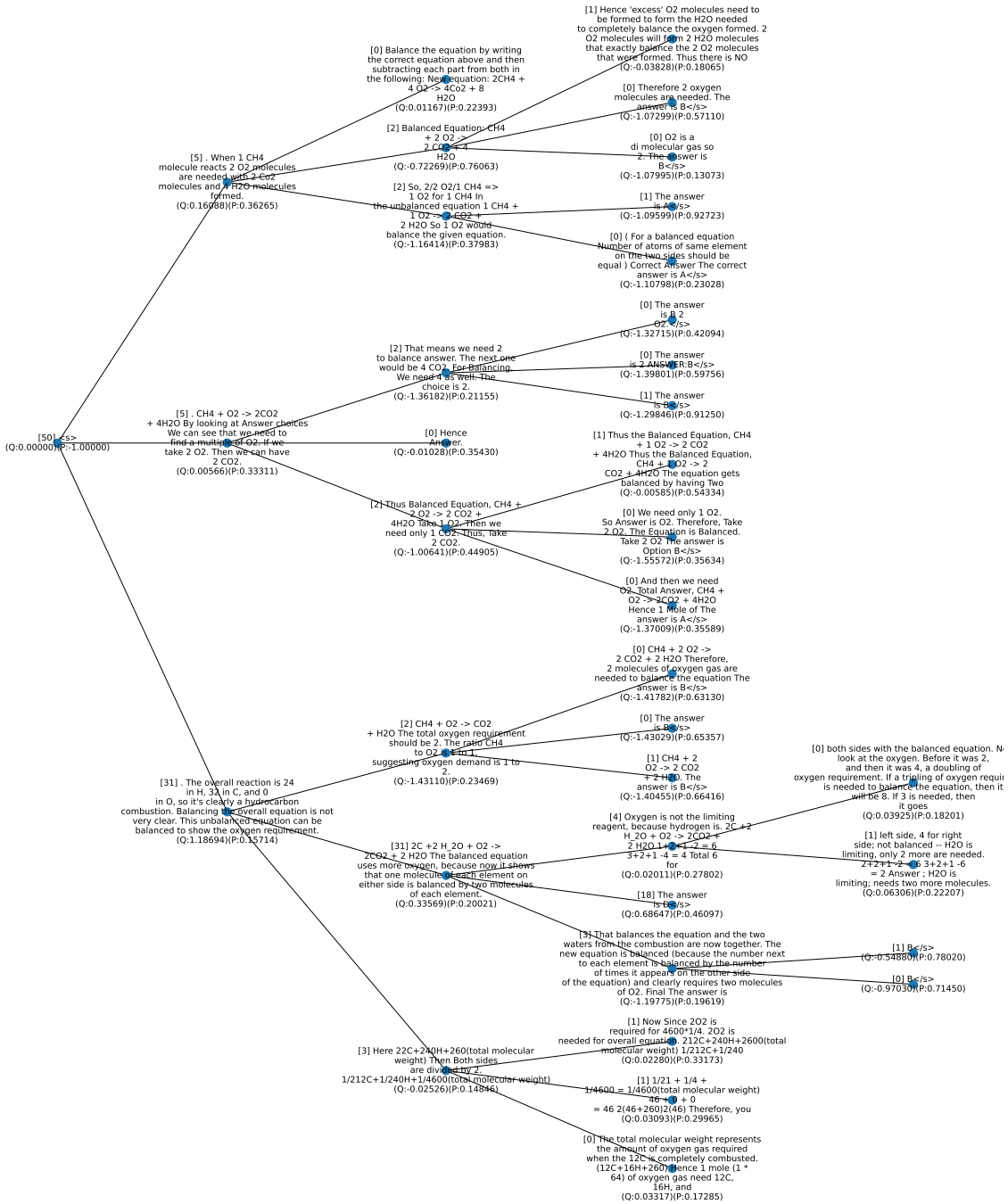


Figure 6: Example of the result search tree of the same science question as in Figure 5. Here, we set the search breadth as $b_1 = 3, b_2 = 3$.

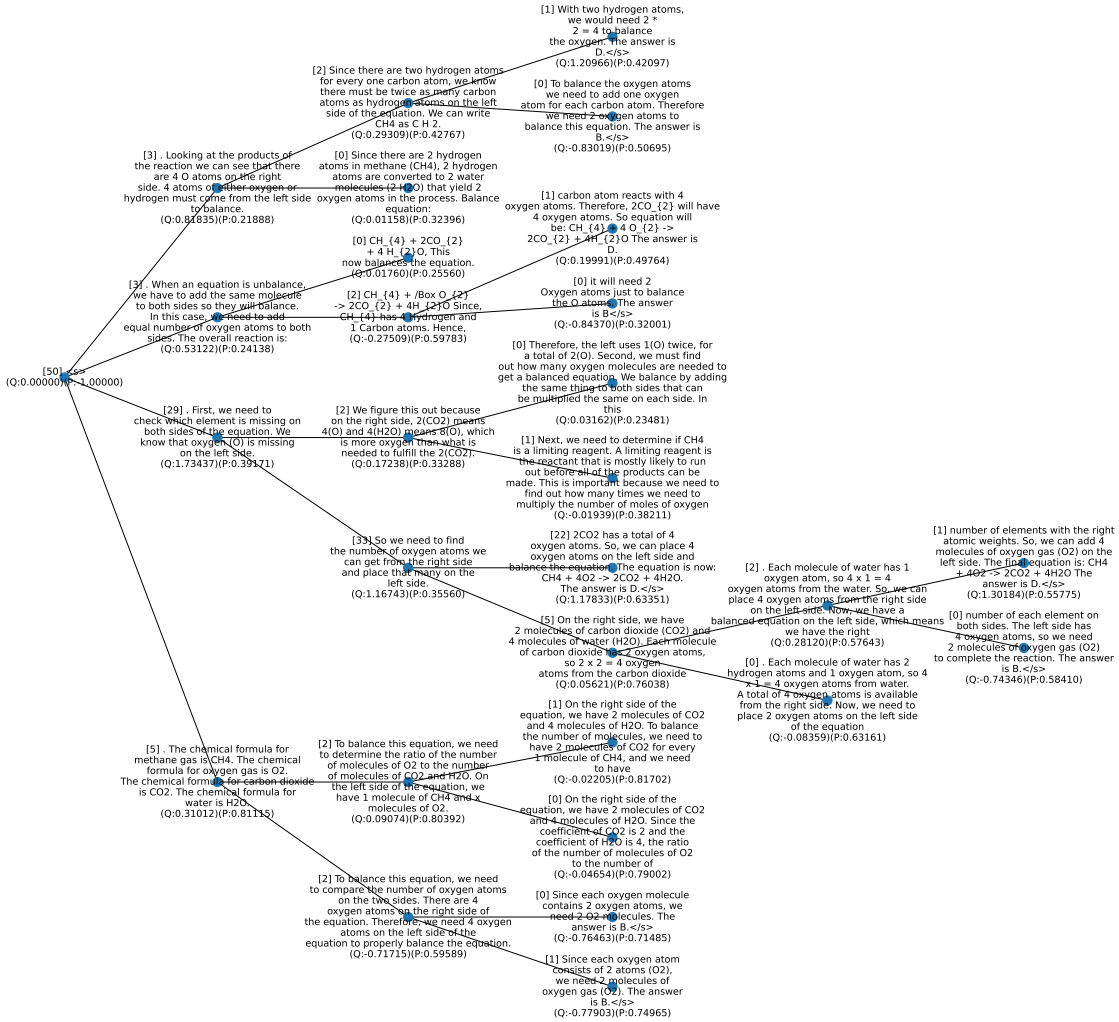


Figure 7: Example of the result search tree of the same science question as in Figure 5. Here, we use the policy after preference learning and set the search breadth as $b_1 = 4, b_2 = 2$.

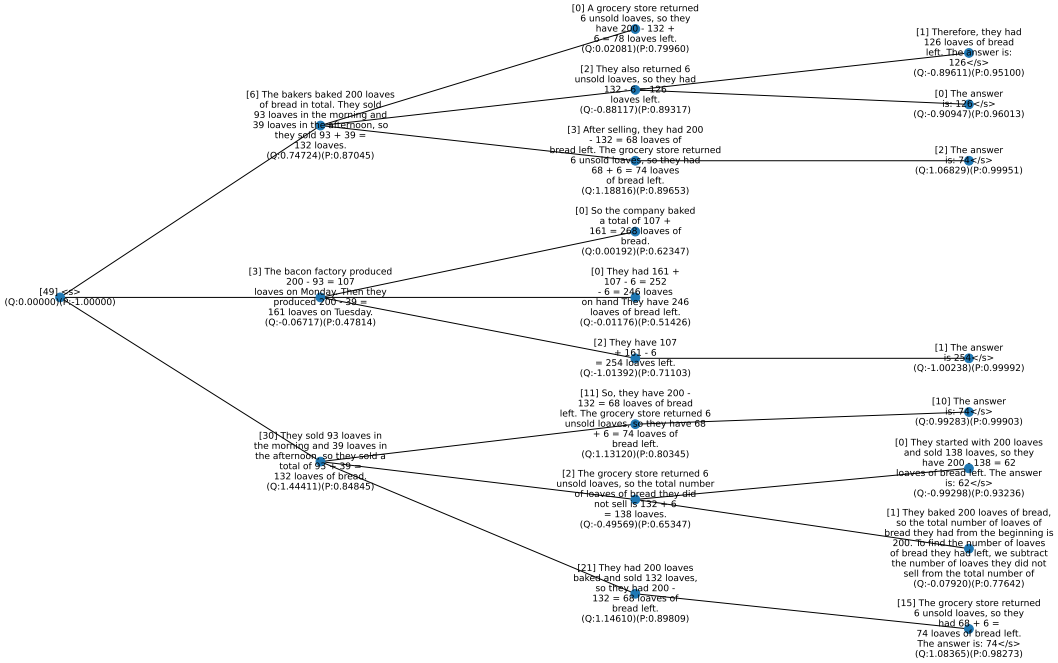


Figure 8: Example of the result search tree of a GSM8K question “The bakers at the Beverly Hills Bakery baked 200 loaves of bread on Monday morning. They sold 93 loaves in the morning and 39 loaves in the afternoon. A grocery store returned 6 unsold loaves. How many loaves of bread did they have left?”. The example solution is “The Bakery sold 93 + 39 = 132 loaves. The Bakery made 200 loaves and sold 132, leaving 200 - 132 = 68 loaves remaining. The grocery store returned 6 loaves, so there were 6 + 68 = 74 loaves left.”. The policy we use here is the one only tuned for 1 epoch on SFT training data. We conduct MCTS with breadth $b_1 = 5, b_2 = 3$. Duplicate generations are merged into one node.

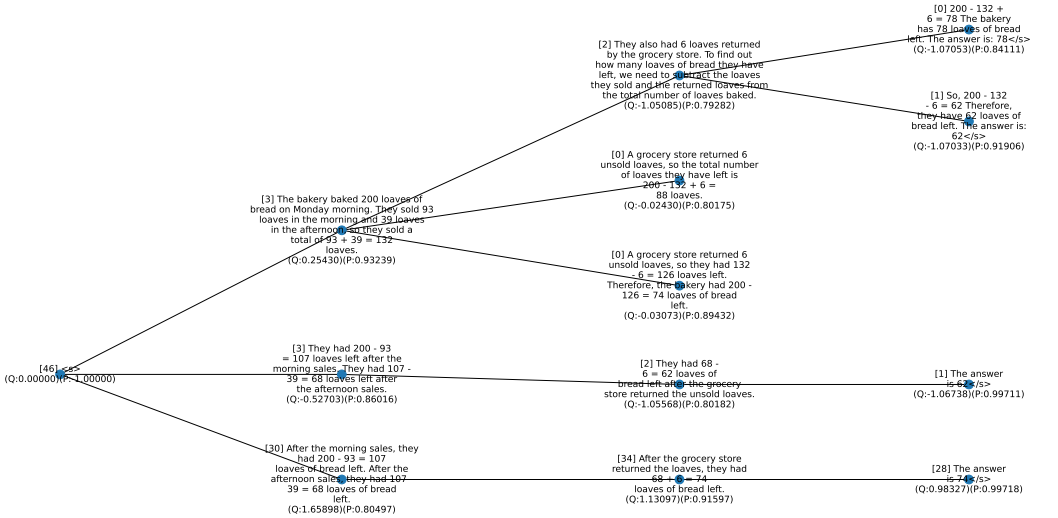


Figure 9: Example of the result search tree of the same GSM8K question as in Figure 8 with the same search breadth. We use the policy tuned after 3 epochs to sample the generations.