Teaching LLMs to Plan, Not Just Solve: Plan Learning Boosts LLMs Generalization in Reasoning Tasks

Tianlong Wang[1,2*†] , **Junzhe Chen**[1,3*†] , **Weibin Liao**[2*] , **Xueting Han**[1‡] , **Jing Bai**[1]
[1] Microsoft Research Asia [2] Peking University [3] Tsinghua University

Abstract

Reinforcement learning (RL) on self-generated data has emerged as a promising paradigm for improving reasoning in large language models (LLMs). However, RL relies on accurate reward signals, which are scarce in many domains, making it critical to train models that can generalize to unseen problems. Existing methods often focus on task-specific or domain-specific reasoning, lacking consideration for generalization and may degrade performance on other tasks. To address this, we distinguish between abstract plans, representing high-level problem-solving strategies, and concrete solutions, proposing that learning plans develops transferable general reasoning capabilities and promotes better generalization. Building on this insight, we propose `PlanLearn`, a framework that combines plan-based search with Step-level Advantage Preference Optimization (Step-APO) to optimize plan learning. Experimental results show that `PlanLearn`, trained exclusively on GSM8K and MATH, not only significantly improves in-domain performance but also enhances out-of-domain benchmarks, such as HumanEval (+12.2%), GPQA (+8.6%), ARC-C (+4.0%), MMLU-STEM (+2.2%), and BBH (+1.8%). The code is available at https://github.com/tianlwang/PlanLearn.

1 Introduction

Large language models (LLMs) have achieved significant success through scaling in pre-training (OpenAI et al., 2024; Dubey et al., 2024). To further unblock the potential of LLMs, recent efforts extend this scaling paradigm to post-training, where reinforcement learning (RL) on self-generated data has emerged as a new learning paradigm (e.g., STaR (Zelikman et al., 2022), ReST (Gulcehre et al., 2023)). Notably, systems

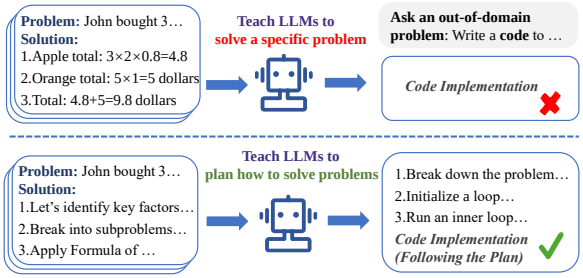

Figure 1: Teaching LLMs to Plan, Not Just Solve. Existing methods primarily focus on teaching LLMs to solve task-specific or domain-specific problems, which limits their generalization to unseen tasks and can degrade performance on other tasks. We propose teaching LLMs to learn **plans**—high-level problem-solving strategies—to foster transferable reasoning skills and improve generalization across reasoning tasks.

like OpenAI's o1 (OpenAI, 2024) demonstrate that large-scale RL can teach models to think more systematically, significantly enhancing their reasoning capabilities. Additionally, recent research works (Xie et al., 2024; Feng et al., 2023; Chen et al., 2024) leverage Monte Carlo Tree Search (MCTS) (Kocsis and Szepesvári, 2006) to collect high-quality reasoning paths, which RL then uses for iterative self-improvement.

However, RL requires accurate reward signals to guide the learning process; for example, ground truth answers are available to evaluate correctness in mathematical tasks, but obtaining such high-quality supervision remains challenging in many domains. To build a general reasoner, it is crucial to enhance the model's generalization ability. Most existing methods (Feng et al., 2023; Chen et al., 2024) focus on enhancing task-specific or domain-specific reasoning capabilities, such as in mathematics or coding, often relying on task-specific fine-tuning, which hampers generalization and may degrade performance on other tasks (Wang et al., 2024). Furthermore, unlike traditional RL, which operates in a limited action space, LLMs function

*These authors contributed equally to this work.
†Work done during internship at Microsoft Research Asia.
‡Corresponding author: chrihan@microsoft.com

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 8531–8545
November 4-9, 2025

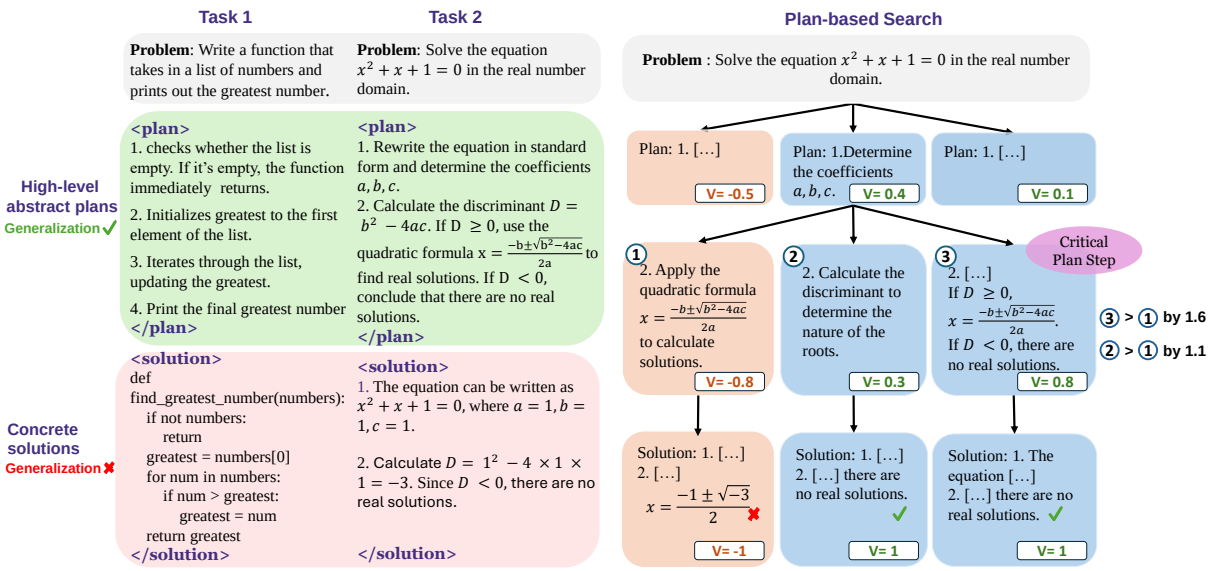

Figure 2: Illustration of PlanLearn. Left: Plans represent abstract thinking for problem-solving, which allows for better generalization, whereas concrete solutions often limit it. Right: PlanLearn searches within the action space on high-level abstract plans using MCTS and obtains advantage estimates for step-level preferences. PlanLearn can then identify and learn critical steps that provide a clear advantage over others.

within a vast search space. This expansive scope, combined with the high inference latency of LLMs, limits both the diversity and quality of explored reasoning paths.

To enhance generalization, we propose *teaching LLMs to plan, not just solve*. Specifically, we distinguish two components in reasoning traces (Wei et al., 2023; Wang et al., 2023): 1) **Abstract Plans**: High-level strategies expressed in natural language, focusing on general reasoning patterns like determining applicable knowledge, breaking down problems, and analyzing key information. 2) **Concrete Solutions**: Implementations derived from the plan—such as mathematical formulas, code, or symbolic solutions—are closely tied to task-specific skills. For example, outlining how to implement code is an abstract plan, while the actual code is a solution. Based on this distinction, we emphasize searching and learning on plans so that models develop a human-like thinking process and generalized problem-solving skills, rather than memorizing concrete solutions, thus improving generalization (illustrated in Figure 2 Left). Furthermore, with the challenge of a vast search space for reasoning paths, plan-based search enables more diverse exploration of high-level strategies, whereas solutions-based search may limit diversity, as different solutions may share the same underlying thought.

To enable effective plan learning, existing preference optimization methods face challenges. Direct Preference Optimization (DPO) (Rafailov et al., 2023) struggles with complex multi-step reasoning tasks due to its inability to capture fine-grained supervision. Recent works propose Step-level DPO (Setlur et al., 2024; Lai et al., 2024) to learn step-level preferences, but their reliance on heuristics to select preference pairs and treat all pairs equally limits model optimization. To address this, we propose a method to identify and learn critical plan steps (i.e., steps that provide greater advantages over others) for improving the model's reasoning ability, as illustrated in Figure 2 Right.

Thus, we introduce PlanLearn, which consists of two key components:

1. Searching on plan, specifically, we devise a step-by-step plan to solve the problem, with the final step providing the concrete solution based on the plan. Using MCTS to explore diverse plan steps in multi-step reasoning tasks, it creates a plan tree, where high-quality plan step preferences are derived from the final outcome. This process enables the exploration of high-level strategies, helping the model acquire task-agnostic skills and improve generalization across different tasks.

2. Learning critical plan steps through Step-level Advantage Preference Optimization (Step-APO), which builds upon DPO. Step-APO integrates advantage estimates for step-level preference pairs obtained via MCTS. This enables the model to

learn fine-grained preferences between steps, boost critical plan steps, and de-emphasize erroneous ones.

To conclude, our contributions are: 1) We distinguish the roles of abstract plans and concrete solutions in reasoning traces and their impact on generalization. We propose that searching and RL on high-level abstract plans can enhance model generalization. 2) We introduce a novel approach `PlanLearn`, which leverages MCTS to explore diverse plan steps, distinguishing it from existing methods that focus on exploring solutions, and uses our Step-APO to learn step-level plan preferences, thereby helping the model effectively identify and learn critical steps. 3) Extensive experiments show that `PlanLearn` enhances reasoning capabilities and generalization across tasks, achieving significant improvements in both in-domain and out-of-domain tasks.

2 Methods

In this section, we introduce our `PlanLearn`, which boosts model performance via an iterative process of plan-based search and step-level preference learning. We first introduce our plan-based MCTS, which enables the LLM to explore diverse plan strategies in the vast search space. Next, we present our Step-APO in detail to further explore the potential of step-level preference learning in multi-step reasoning tasks. Finally, we describe how we iteratively optimize the policy and value models.

2.1 Plan-based MCTS

MCTS builds a reasoning tree iteratively and autonomously explores step-level reasoning traces, which can be used to optimize LLMs. Existing methods (Chen et al., 2024; Xie et al., 2024) that leverage MCTS to collect data for training usually focus on exploring solution steps within the entire search space or on simultaneously exploring both plans and solutions. We propose searching on high-level abstract plans, expressed in natural language, as a universal interface across tasks. These plans represent abstract thinking for problem-solving, such as determining which knowledge to apply or how to decompose a problem, enabling models to develop broader, task-agnostic capabilities that enhance generalization.

Specifically, we design a two-shot prompt (See Appendix E) to guide the model in answering questions in two parts: (1) a step-by-step abstract plan to solve the problem, and (2) a concrete solution based on the plan. We obtain a plan tree and high-quality plan step preferences through iterative search with MCTS, in which each intermediate edge corresponds to a plan step and the final edge represents the concrete implementation.

Given the plan tree, each node represents a state $\mathbf{s}_t$, and each edge represents an action $\mathbf{a}_t$, which corresponds to a reasoning step that leads to the next state $\mathbf{s}_{t+1}$. Under the same parent node, different sibling nodes form a set of step-level preference pairs, with each node having its own value $V(\mathbf{s}_t)$ representing the expected future reward under state $\mathbf{s}_t$. These values can be obtained through the MCTS process, which involves four key operations: selection, expansion, evaluation, and backup. To enhance efficiency, we use a value model to assess the expected returns from the partial reasoning paths, with the final integration of both policy and value models guiding the search process. Next, we describe the four steps of MCTS.

Selection: We use the PUCT algorithm (Rosin, 2011) to select nodes in the search tree with the following formula, where N represents the visit count, $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ represents LLMs' generation probability, c_{puct} is a constant that balances exploitation and exploration:

$$\arg\max_{\mathbf{a}_t}\left[Q(\mathbf{s}_t,\mathbf{a}_t)+c_{\text{puct}}\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)\frac{\sqrt{N(\mathbf{s}_t)}}{1+N(\mathbf{s}_t,\mathbf{a}_t)}\right]. \quad (1)$$

Expansion: We expand the selected node by using the current states as the prompt, which contains the question and all previously generated steps, to guide the model in generating the next step candidates via temperature sampling.

Evaluation: During evaluation, the terminal node, which contains the concrete solution, is assessed by comparing the final answer with the ground truth, while the values of non-terminal nodes, which contain plan steps, are predicted by the value model for efficiency, instead of relying on resource-intensive rollout.

Backup: After evaluation, we perform a bottom-up backpropagate from the evaluated node back to the root. We update the visit count N, the state value V, and the transition value Q as follows:

$$Q(\mathbf{s}_t,\mathbf{a}_t)\leftarrow r(\mathbf{s}_t,\mathbf{a}_t)+V(\mathbf{s}_{t+1}), \quad (2)$$

$$V(\mathbf{s}_t)\leftarrow\sum_a N(\mathbf{s}_{t+1})Q(\mathbf{s}_t,\mathbf{a}_t)/\sum_a N(\mathbf{s}_{t+1}), \quad (3)$$

$$N(\mathbf{s}_t)\leftarrow N(\mathbf{s}_t)+1. \quad (4)$$

2.2 Step-APO to Learn Critical Plan Steps

Existing preference learning methods, such as Direct Preference Optimization (DPO) (Rafailov et al., 2023) face challenges in learning critical steps due to their inability to capture fine-grained supervision. Recent works propose Step-level DPO (Hwang et al., 2024; Lai et al., 2024) to learn step-level preferences, but its reliance on heuristics, such as marking the first error step as dis-preferred, limits the richness of paths learned and model optimization. Given the large variations in advantage differences across different data pairs, we propose Step-APO, which introduces advantage estimates for preference pairs into DPO. This enables the model to more effectively learn critical intermediate plan steps, thereby further improving its reasoning capabilities. Next, we will provide its derivation and analysis from the perspective of its gradient.

2.2.1 Preliminaries

The Classical RL Objective RLHF approaches (Ziegler et al., 2020; Bai et al., 2022; Ouyang et al., 2022) usually first learn a reward function from human feedback, then optimize it with a policy gradient-based method like PPO (Schulman et al., 2017) with an entropy-bonus using the following multi-step RL objective:

$$\max_{\pi_\theta} \mathbb{E}_{\mathbf{a}_t \sim \pi_\theta(\cdot|\mathbf{s}_t)} \left[\sum_{t=0}^{T} \left(r(\mathbf{s}_t, \mathbf{a}_t) + \mathrm{KL}_t \right) \middle| \mathbf{s}_0 \sim \rho(\mathbf{s}_0) \right], \quad (5)$$
$$\mathrm{KL}_t = \beta \log \pi_{\text{ref}}(\mathbf{a}_t|\mathbf{s}_t),$$

where $r(\mathbf{s}_t, \mathbf{a}_t)$ denotes the step-level reward function. KL_t is a KL penalty that aims to ensure the learned policy π_θ does not deviate significantly from the reference policy π_{ref}, which is typically produced via supervised fine-tuning.

Direct Preference Optimization DPO (Rafailov et al., 2023) uses the well-known closed-form optimal solution, which establishes a mapping between the reward model and the optimal policy under the KL divergence, obtaining the reward as:

$$r(\mathbf{x}, \mathbf{y}) = \beta \log \pi^*(\mathbf{y}|\mathbf{x}) - \beta \log \pi_{\text{ref}}(\mathbf{y}|\mathbf{x}) - Z(\mathbf{x}), \quad (6)$$

where $\mathbf{x}$ denotes the prompt and $\mathbf{y}$ denotes the response, π^* is the optimal policy, and $Z(\mathbf{x})$ is the partition function that normalizes it.

By substituting Equation 6 into the Bradley-Terry preference model and leveraging the maximum likelihood objective, DPO derives the loss:

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(\mathbf{x}, \mathbf{y}^w, \mathbf{y}^l) \sim \mathcal{D}} \left[\log \sigma \left(u(\mathbf{x}, \mathbf{y}^w, \mathbf{y}^l) \right) \right],$$
$$u(\mathbf{x}, \mathbf{y}^w, \mathbf{y}^l) = \beta \log \frac{\pi_\theta(\mathbf{y}^w \mid \mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}^w \mid \mathbf{x})} - \beta \log \frac{\pi_\theta(\mathbf{y}^l \mid \mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}^l \mid \mathbf{x})}, \quad (7)$$

where σ denotes the logistic function, and $\mathbf{y}^w$ and $\mathbf{y}^l$ denote the preferred and dis-preferred responses to the prompt $\mathbf{x}$.

2.2.2 Deriving the Step-APO Objective

In the general maximum entropy RL setting (Ziebart, 2010), the optimal policy $\pi^*(\mathbf{a}|\mathbf{s})$ of multi-step RL objective in Equation 5 is:

$$\pi^*(\mathbf{a}_t|\mathbf{s}_t) = e^{(Q^*(\mathbf{s}_t, \mathbf{a}_t) - V^*(\mathbf{s}_t))/\beta}, \quad (8)$$

where $Q^*(\mathbf{s}, \mathbf{a})$ is the optimal Q-function which models the expected future reward from $(\mathbf{s}_t, \mathbf{a}_t)$ under π^*. The optimal value function V^* estimates the expected future reward under state $\mathbf{s}_t$, and it's a function of Q^* (Rafailov et al., 2024).

Under the reward r with a KL divergence penalty, the relationship between Q-function and step-level reward function can be established with the Bellman equation as follows:

$$Q^*(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + \beta \log \pi_{\text{ref}}(\mathbf{a}_t|\mathbf{s}_t) + V^*(\mathbf{s}_{t+1}). \quad (9)$$

By log-linearizing the optimal policy in Equation 8 and substituting in the Bellman equation from Equation 9 (Nachum et al., 2017; Rafailov et al., 2024), we have below equation which is precisely the optimal advantage function $A^*(\mathbf{s}, \mathbf{a}) = Q^*(\mathbf{s}, \mathbf{a}) - V^*(\mathbf{s})$:

$$\beta \log \frac{\pi^*(\mathbf{a}_t|\mathbf{s}_t)}{\pi_{\text{ref}}(\mathbf{a}_t|\mathbf{s}_t)} = r(\mathbf{s}_t, \mathbf{a}_t) + V^*(\mathbf{s}_{t+1}) - V^*(\mathbf{s}_t). \quad (10)$$

Unlike DPO utilize response-level Bradley-Terry model, we introduce step-level Bradley-Terry preference model to learn fine-grained step-level preference:

$$p^*(\mathbf{a}^w \succeq \mathbf{a}^l|\mathbf{s}) = \frac{\exp\left(r(\mathbf{s}, \mathbf{a}^w)\right)}{\exp\left(r(\mathbf{s}, \mathbf{a}^w)\right) + \exp\left(r(\mathbf{s}, \mathbf{a}^l)\right)}. \quad (11)$$

By substituting Equation 10 into Equation 11 and leveraging the negative log-likelihood loss, we derive the objective for Step-APO:

$$\mathcal{L}_{\text{Step-APO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t^w, \mathbf{a}_t^l) \sim \mathcal{D}} \left[\log \sigma \left(u(\mathbf{s}_t, \mathbf{a}_t^w, \mathbf{a}_t^l) \right) \right],$$
$$u_\theta(\mathbf{s}_t, \mathbf{a}_t^w, \mathbf{a}_t^l) = \beta \log \frac{\pi_\theta(\mathbf{a}_t^w \mid \mathbf{s}_t)}{\pi_{\text{ref}}(\mathbf{a}_t^w \mid \mathbf{s}_t)} - \beta \log \frac{\pi_\theta(\mathbf{a}_t^l \mid \mathbf{s}_t)}{\pi_{\text{ref}}(\mathbf{a}_t^l \mid \mathbf{s}_t)}$$
$$- V(\mathbf{s}_{t+1}^w) + V(\mathbf{s}_{t+1}^l), \quad (12)$$

where $V(\mathbf{s}_{t+1}^w) - V(\mathbf{s}_{t+1}^l)$ denotes the advantage of $\mathbf{s}_{t+1}^w$ over $\mathbf{s}_{t+1}^l$ from the same start state.

To understand the difference between our Step-APO and other step-level DPO, we analyze the gradient of the $\mathcal{L}_{\text{Step-APO}}$:

$$\nabla_\theta \mathcal{L}_{\text{Step-APO}} = -\beta \mathbb{E}\big[\sigma\big(-u_\theta(\mathbf{s}_t, \mathbf{a}_t^w, \mathbf{a}_t^l)\big)\big] \cdot \Big[\nabla_\theta \log \pi(\mathbf{a}_t^w \mid \mathbf{s}_t) - \nabla_\theta \log \pi(\mathbf{a}_t^l \mid \mathbf{s}_t)\Big]. \quad (13)$$

Intuitively, the gradient increases the likelihood of preferred completions $\mathbf{a}_t^w$ and decreases that of dispreferred ones $\mathbf{a}_t^l$, with weights determined by the preference and advantage differences. Importantly, besides the examples are weighed by how much higher the $\hat{r}_\theta(\mathbf{s}_t, \mathbf{a}_t) = \beta \log \frac{\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}{\pi_{\text{ref}}(\mathbf{a}_t|\mathbf{s}_t)}$ incorrectly orders the completions, the examples are also weighted by how much higher the advantage of $\mathbf{a}_t^w$ is compared to $\mathbf{a}_t^l$. This allows for assigning different optimization weights and emphasizes critical steps. Our experiments prove the importance of this weighting.

2.3 Iterative Training of Policy and Value Model

Our approach employs iterative training for policy and value models. The policy model, denoted as π_θ, and the value model, denoted as v_ϕ, are two separate models adapted from the same base model. We add a value head for the value model, which is randomly initialized in the first round. However, as the MCTS simulations proceed in the first round, rewards from terminal nodes are back-propagated to the intermediate nodes, reducing the negative impact of the random value initialization.

For policy model training, we first perform supervised fine-tuning (SFT) using the correct paths collected from MCTS. We then apply our Step-APO (12) using step-level preference data also collected from MCTS. Notably, $V(\mathbf{s}_{t+1}^w)$ and $V(\mathbf{s}_{t+1}^l)$ in Equation 12, obtained from MCTS, represent the values of the corresponding states. The difference between these values reflects the advantage difference of the two actions under the same previous state $\mathbf{s}_t$:

$$\begin{aligned} A(\mathbf{s}_t, \mathbf{a}_t^w) - A(\mathbf{s}_t, \mathbf{a}_t^l) = \\ Q(\mathbf{s}_t, \mathbf{a}_t^w) - V(\mathbf{s}_t) - \Big(Q(\mathbf{s}_t, \mathbf{a}_t^l) - V(\mathbf{s}_t)\Big) \\ = V(\mathbf{s}_{t+1}^w) - V(\mathbf{s}_{t+1}^l). \end{aligned} \quad (14)$$

For value model optimization, we use a mean squared error (MSE) loss between the value model's prediction and values from MCTS. With the updated policy and value models, we advance to the next round of MCTS and iterate the training process to further enhance the models.

3 Experiments

In this section, we conduct a comprehensive evaluation of `PlanLearn` across two in-domain datasets and five out-of-domain (OOD) datasets to address the following research questions:

- **RQ1** (subsection 3.2): When trained exclusively on mathematical reasoning data, does `PlanLearn` achieve superior performance compared to existing methods on both in-domain and out-of-domain reasoning tasks?
- **RQ2** (subsection 3.3): Does plan-based learning effectively enhance the generalization capabilities of models across diverse reasoning tasks?
- **RQ3** (subsection 3.4): Does Step-APO provide a more effective optimization strategy for plan learning compared to existing approaches?

Furthermore, we explore the construction of step-level preference data to enhance generalization capabilities (subsection 3.5) and demonstrate the generalizability of `PlanLearn` across different models (subsection 3.6).

3.1 Implementation Details

We iteratively generate data using MCTS and train our policy and value models over two rounds. In each round, the policy model generates plan steps and concrete solutions via MCTS, while the value model assists in evaluating the intermediate steps. At the end of each round, the generated data is used to train both the policy and value models.

Model Architecture We employ the DeepSeekMathBase-7B (Shao et al., 2024) as our initial policy model and add a randomly initialized value head to this model, serving as the initial value model. We then optimize these two distinct models independently and use the updated models for the next round of data generation.

Datasets We construct our training data using the training sets from GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021b) datasets. GSM8K comprises 7,473 training and 1,319 test problems, while MATH includes 7,500 training and 5,000 test problems. From these datasets, we exclusively extracted question-answer pairs from the training sets, omitting the human-annotated solutions. This resulted in a total of 15k question-answer pairs for

Model	In-Domain		Out-of-Domain				
	MATH	GSM8K	HumanEval	ARC-C	GPQA	BBH	MMLU-stem
DeepseekMath-Base	35.18	63.23	40.90	52.05	25.75	58.79	52.74
STaR	37.33	69.98	43.29	53.45	28.78	60.04	54.33
Self-Explore-MATH	37.86	**78.39***	41.46	54.01	33.83	60.04	54.04
AlphaMath	-	-	49.39	53.41	33.33	56.63	55.31
`PlanLearn` (Round 1 SFT)	36.30	63.79	42.68	54.44	28.78	59.68	54.58
`PlanLearn` (Round 1 Step-APO)	40.56	71.06	46.34	55.55	31.31	60.18	55.15
`PlanLearn` (Round 2 SFT)	39.16	69.75	48.78	54.95	29.79	59.93	**55.44**
`PlanLearn` (Round 2 Step-APO)	**41.64**	73.77	**53.05**	**56.06**	**34.34**	**60.54**	54.93

Table 1: Main results on in-domain and out-of-domain reasoning tasks. Baseline results on MATH and GSM8K are reproduced. * denotes results from Self-Explore-GSM8K. - indicates that the model uses Python code interpreter and is not comparable with our method. Best results are bolded.

our training data.

Training Data Generation via MCTS For each problem, we employ MCTS to generate multiple step-level plans and the concrete solution. During the MCTS expansion phase, we expand 5 child nodes for the root node and 3 child nodes for other nodes. The search is conducted with a maximum depth of 6. We apply a temperature of 0.7 to encourage diverse generation.

In the first round, we generate data from a subset of 5k question-answer pairs, consisting of 4k from the MATH and 1k from GSM8K, for efficiency. We carefully design prompts and 2-shot demonstrations to guide the model's output (see Appendix E for details). We perform a large number of MCTS simulations, specifically 200, in this phase to mitigate the impact of the random initialization of the value model. Starting from the second round, with the fine-tuned models from the first round, we utilize the full set of 15k question-answer pairs for data generation. A 2-shot prompt formatted in XML is used, and we perform 100 MCTS simulations.

Training Data Construction We utilize the state value V of each node in MCTS to construct preference data. For plan step preference data, we categorize sibling nodes as "preferred" if their value is greater than 0 and "dispreferred" if their value is less than 0, forming preference pairs from any combination. For concrete solution step data, we randomly select one preference pair for each parent node to construct the dataset. This is based on our experimental findings that an excess of solution data can negatively impact performance on OOD reasoning tasks, whereas increasing the emphasis on plan data improves performance in both mathematical and other reasoning tasks (see subsection 3.5). We provide the statistics for the generated data in two rounds in Appendix B.

Training Details For the policy model, we first randomly select up to four correct responses per problem for supervised fine-tuning (SFT). Next, we employ step-level preference data from MCTS to train the model with our Step-APO algorithm.

For the value model, we use state value V from MCTS for partial responses as labels to update the model. This allows the value model to score both partial plans and complete responses.

Notably, because the value difference for final solution step preference pairs is 2, while the average value difference for other plan steps ranges between 0.6 and 0.8, we apply a scaling factor of 0.3 to the values of solution steps in Step-APO. In the second round of training, we utilize the data from the second round to train the base model, rather than the Round 1 model. The training hyperparameters are provided in Appendix D.

3.2 Main Results

We evaluate our method on both mathematical tasks and OOD reasoning tasks, as shown in Table 1.

Baseline Our baseline includes DeepseekMath-Base-7B (Shao et al., 2024), along with three additional baselines, STaR (Zelikman et al., 2022), Self-Explore (Hwang et al., 2024), AlphaMath (Chen et al., 2024). We reran STaR by repeated sampling Chains of Thought (CoTs) using the same model and data as ours. The latter two baselines are developed based on DeepSeekMath-Base and utilize solution-centric search methods, employing the GSM8K and MATH datasets, distinguishing them from our proposed plan-based search methodology.

In-domain tasks We evaluate `PlanLearn` in-

domain capabilities on MATH (Hendrycks et al., 2021b) and GSM8K (Cobbe et al., 2021). Details of the datasets and evaluation can be found in Appendix C.

On in-domain tasks, PlanLearn demonstrated significant performance improvements over DeepseekMath-Base on both the MATH and GSM8K datasets. Compared to Self-Explore, which leverages human-annotated rational data and extensive self-generated data for fine-tuning, PlanLearn does not use any human-annotated rational data and relies solely on the model's self-exploration with much less SFT data. While Self-Explore performs better on simpler tasks like GSM8K, possibly due to the use of golden rationales and more SFT data, our method significantly outperforms it on more challenging tasks like MATH, which requires more complex self-exploration of reasoning paths. These results underscore the efficacy of plan-based learning in enhancing the model's in-domain reasoning capabilities.

In both rounds, Step-APO consistently improves results over SFT. Additionally, Round 2 outperforms Round 1 in both SFT and Step-APO, demonstrating that the updated policy and value models generate better data through MCTS, further improving performance.

Out-of-domain reasoning tasks We select five benchmarks for evaluating OOD reasoning: HumanEval (Chen et al., 2021), ARC-C (Clark et al., 2018), GPQA (Rein et al., 2023), BBH (Suzgun et al., 2022), and MMLU-STEM (Hendrycks et al., 2021a). We employ few-shot prompting to evaluate these benchmarks. Details of the datasets and evaluation can be found in Appendix C.

As shown in Table 1, PlanLearn also achieves significant improvements on OOD tasks, with average improvements of 5.7%, 4.1%, 3.1%, and 2.2% compared to the base model, STaR, Self-Explore-MATH, and AlphaMath, respectively. This demonstrates that PlanLearn enhances the model's generalization ability across diverse reasoning tasks. Compared to AlphaMath, which was trained on the same 15k dataset for 3 rounds, our performance on these OOD reasoning tasks is much better. Notably, AlphaMath even shows a decrease in performance on certain tasks, such as a 2.2% drop in BBH.

Unlike baseline methods that focus on concrete solutions within the vast reasoning action space of LLMs, PlanLearn concentrates on exploring the action space on high-level abstract plans. These plans embody abstract problem-solving strategies, enabling models to develop broader, task-agnostic abilities that enhance generalization.

3.3 Effectiveness of Plan-based Learning

To validate whether plan-based learning enhances generalization, we compare plan-based MCTS with non-plan-based MCTS within the Round 1 SFT setting, evaluating on HumanEval, ARC-C, and BBH benchmarks. The results are shown in Table 2. The results demonstrate that MCTS-based searching for high-level abstract plans consistently outperforms searching for concrete solutions, showing stronger generalization capabilities.

Model	HumanEval	ARC-C	BBH
DeepseekMath-Base	40.90	52.05	58.79
w/o plan learning	38.41	53.00	59.03
w/ plan learning	**42.68**	**54.44**	**59.68**

Table 2: Effectiveness of plan-based learning.

We observed that SFT model with non-plan-based data resulted in a performance decline on HumanEval, consistent with the 2.2% drop in BBH performance shown by AlphaMath in Table 1. These results suggest that existing approaches focusing on task-specific learning significantly limit OOD generalization and may hurt performance on other tasks. In contrast, plan-based learning consistently improves performance across OOD tasks, highlighting its effectiveness.

3.4 Effectiveness of Step-APO

To investigate the effectiveness of Step-APO, we compared the performance of Instance-DPO, TDPO (Zeng et al., 2024), Step-DPO, and Step-APO using data obtained from the first round of MCTS. Instance-DPO involves preference learning on the model's complete response based on the correctness of the final answer. Step-DPO and Step-APO, on the other hand, performs finer-grained learning on each step within the model's response. The difference is that Step-APO incorporates advantage estimates for preference pairs obtained through MCTS, while Step-DPO does not. TDPO improves DPO by extending it to the token level and incorporates forward KL divergence constraints for each token, improving alignment. All four preference learning strategies were applied to the model after supervised fine-tuning (SFT). The experimental results are summarized in Table 3.

Model	In-Domain		Out-of-Domain				
	MATH	GSM8K	HumanEval	ARC-C	GPQA	BBH	MMLU-stem
SFT	36.30	63.79	42.68	54.44	28.78	59.68	54.58
Instance-DPO	37.72	69.29	43.90	54.61	24.24	60.13	54.42
TDPO	39.12	69.90	48.78	54.61	28.29	59.94	54.08
Step-DPO	37.89	69.83	42.68	54.44	25.25	59.44	54.68
Step-APO	**40.56**	**71.06**	**48.78**	**55.55**	**31.31**	**60.18**	**55.15**

Table 3: Effectiveness of Step-APO.

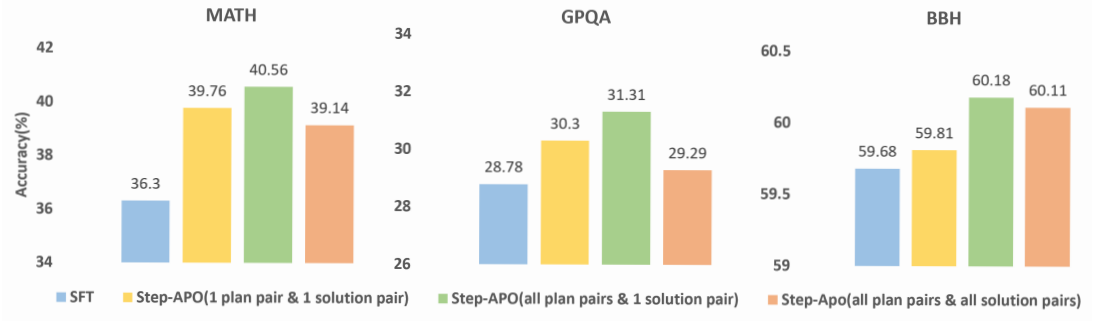

Figure 3: Impact of data construction. We outline different strategies for constructing Step-APO preference data based on the value V of each node in MCTS: 1 plan pair (selecting the plan step with the maximum positive V and the plan with the minimum negative V), 1 solution pair (randomly selecting one correct and one incorrect solution step), all plan pairs (selecting all combinations of plan steps with positive and negative V), and all solution pairs (selecting all combinations of correct and incorrect solution steps).

The results show that TDPO and Step-APO outperform Instance-DPO and Step-DPO across both in-domain and OOD tasks, primarily due to their distinct mechanisms: Instance-DPO fails to capture fine-grained preferences in reasoning processes, while Step-DPO, despite introducing step-wise supervision, treats all step pairs equally without accounting for advantage differences between steps, thereby restricting its optimization capability. Although TDPO performs better than Instance-DPO, it underperforms Step-APO, which we attribute to TDPO's focus on token-level decision quality, whereas Step-APO explicitly models advantage disparities between steps and prioritizes high-value reasoning steps, ultimately validating its superiority in multi-step reasoning scenarios.

3.5 Data Construction

To investigate how to construct step-level preference data, we use MATH as a representative in-domain task and BBH and GPQA as representative OOD tasks to compare three methods. The experimental results are shown in Figure 3.

Initially, we construct preference data by creating at most one pair for all sibling nodes: for plan steps, we select the plan with the highest positive value and the plan with the lowest negative value; for solution steps, we randomly select one correct and one incorrect solution. Using Step-APO on this data, we observe performance improvements over the SFT model in both in-domain and OOD reasoning tasks. Next, we enhance the plan step data by selecting all combinations of plans with positive and negative values while keeping the solution step data unchanged, which leads to further performance gains across both task types. However, continuously expanding the solution step data results in decreased model performance. Ultimately, we adopt the strategies of using all plan pairs and one solution pair for our experiments.

3.6 Generalizability Across Different Models

To further demonstrate the effectiveness of `PlanLearn` across different models, we conducted experiments on Qwen2.5-3B (Hui et al., 2024) and Llama-3-8B (Meta, 2024). We adhered to the Round 1 setup, utilizing MCTS to generate data from a 5,000-sample subset comprising 4,000 pairs from the MATH dataset and 1,000 from GSM8K. Both models were then optimized using SFT and Step-APO.

As shown in Table 4, the results demonstrate that `PlanLearn` yields significant performance improvements after a single round of optimization. For Qwen-2.5-3B, we observe average gains of 4.2% on in-domain tasks and 3.6% on out-of-domain (OOD) tasks. Similarly, Llama3-8B exhibits an average improvement of 3.4% on in-

Model	In-Domain		Out-of-Domain				
	MATH	GSM8K	HumanEval	ARC-C	GPQA	BBH	MMLU-stem
Qwen-2.5-3B	40.82	76.33	40.24	56.94	25.25	56.82	61.50
+ PlanLearn(Round 1)	**44.55**	**81.05**	**48.78**	**59.85**	**27.78**	**58.25**	**63.88**
Llama-3-8B	18.16	49.17	37.80	57.94	27.27	62.92	55.82
+ PlanLearn(Round 1)	**20.24**	**53.90**	**40.24**	**60.24**	**34.34**	**64.08**	**56.83**

Table 4: Additional results on Qwen2.5-3B and Llama-3-8B.

domain benchmarks and 2.8% on OOD benchmarks. This demonstrates the robustness and strong generalization capabilities of PlanLearn across different models.

4 Conclusion

Developing a general reasoner through RL remains an open and important research question. In this work, we propose searching and learning on high-level abstract plans to enhance model generalization, rather than focusing on concrete solutions that often limit generalization. PlanLearn enhances transfer performance on diverse out-of-domain tasks, underscoring its value for future research in generalizable reasoning.

Limitations

While PlanLearn demonstrates promising generalization when trained on mathematical reasoning tasks and evaluated across a variety of reasoning benchmarks, its effectiveness has so far been validated primarily in a math-centric training setting. Future work may explore whether PlanLearn's benefits extend to scenarios where the training data comes from domains beyond mathematics, such as code-related or commonsense reasoning tasks. This would help further assess whether plan-based learning provides consistent advantages regardless of the training domain.

Ethical Considerations

Our research focuses on improving the reasoning capabilities of LLMs through plan-based learning, which may contribute to more robust and generalizable AI systems. However, we acknowledge that such advancements could also be misused or lead to unintended consequences if applied without appropriate oversight. We encourage future work and deployment of our method to adhere to established ethical guidelines and prioritize the broader societal implications.

All experiments are conducted using publicly available and open-source datasets as well as open-source models, which supports transparency, reproducibility, and fairness in AI research, and aligns with our commitment to responsible scientific practice.

References

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, and 1 others. 2022. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*.

Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. 2024. Alphamath almost zero: process supervision without process. *CoRR*, abs/2405.03553.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, and 1 others. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

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. *Preprint*, arXiv: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.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *CoRR*, abs/2307.08691.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, and 1 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

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

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. A framework for few-shot language model evaluation.

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. *Preprint*, arXiv:2308.08998.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding. *Preprint*, arXiv:2009.03300.

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

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, and 1 others. 2024. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.

Hyeonbin Hwang, Doyoung Kim, Seungone Kim, Seonghyeon Ye, and Minjoon Seo. 2024. Self-explore to avoid the pit: Improving the reasoning capabilities of language models with fine-grained rewards. *Preprint*, arXiv:2404.10346.

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

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. *Preprint*, arXiv:2309.06180.

Xin Lai, Zhuotao Tian, Yukang Chen, Senqiao Yang, Xiangru Peng, and Jiaya Jia. 2024. Step-dpo: Step-wise preference optimization for long-chain reasoning of llms. *arXiv:2406.18629*.

AI Meta. 2024. Introducing meta llama 3: The most capable openly available llm to date. *Meta AI*.

Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. 2017. Bridging the gap between value and policy based reinforcement learning.

OpenAI. 2024. Introducing openai o1-preview.

OpenAI and 1 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

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*.

Rafael Rafailov, Joey Hejna, Ryan Park, and Chelsea Finn. 2024. From r to q^*: Your language model is secretly a q-function.

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

Samyam Rajbhandari, Olatunji Ruwase, Jeff Rasley, Shaden Smith, and Yuxiong He. 2021. Zero-infinity: breaking the GPU memory wall for extreme scale deep learning. In *International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2023. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*.

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.

Amrith Setlur, Saurabh Garg, Xinyang Geng, Naman Garg, Virginia Smith, and Aviral Kumar. 2024. Rl on incorrect synthetic data scales the efficiency of llm math reasoning by eight-fold. *Preprint*, arXiv:2406.14532.

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

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *Preprint*, arXiv:2210.09261.

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. *Preprint*, arXiv:2404.12253.

Chaojie Wang, Yanchen Deng, Zhiyi Lyu, Liang Zeng, Jujie He, Shuicheng Yan, and Bo An. 2024. Q*: Improving multi-step reasoning for llms with deliberative planning. *Preprint*, arXiv:2406.14283.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. *Preprint*, arXiv:2305.04091.

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. *Preprint*, arXiv:2201.11903.

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

Xiang Yue, Tuney Zheng, Ge Zhang, and Wenhu Chen. 2024. Mammoth2: Scaling instructions from the web. *arXiv preprint arXiv:2405.03548*.

Eric Zelikman, Georges Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah D Goodman. 2024. Quiet-star: Language models can teach themselves to think before speaking. *arXiv preprint arXiv:2403.09629*.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. 2022. Star: Bootstrapping reasoning with reasoning. *Preprint*, arXiv:2203.14465.

Yongcheng Zeng, Guoqing Liu, Weiyu Ma, Ning Yang, Haifeng Zhang, and Jun Wang. 2024. Token-level direct preference optimization. *Preprint*, arXiv:2404.11999.

Boning Zhang, Chengxi Li, and Kai Fan. 2024. Mario eval: Evaluate your math llm with your math llm–a mathematical dataset evaluation toolkit. *Preprint*, arXiv:2404.13925.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. *CoRR*, abs/2403.13372.

Brian D Ziebart. 2010. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2020. Fine-tuning language models from human preferences. *Preprint*, arXiv:1909.08593.

A Discussion on Related Work

Search-Guided Reasoning in LLMs Recent advancements (Feng et al., 2023; Chen et al., 2024; Xie et al., 2024) in enhancing LLM reasoning capabilities have focused on integrating Monte Carlo Tree Search (MCTS) to collect trajectories and train models, resulting in notable improvements in reasoning tasks. MCTS strikes a balance between exploration and exploitation, utilizing its look-ahead ability to obtain high-quality step-level supervision. For example, AlphaMath (Chen et al., 2024) employs MCTS to automatically generate process supervision, leading to significant improvements in mathematical reasoning. However, these MCTS-based training methods face challenges such as vast search spaces, limited solution diversity for LLMs. Furthermore, there is limited research on how these methods generalize to other reasoning tasks and enhance overall reasoning capabilities. To address these issues, we propose a method for searching over plan steps and learning critical plan steps for problem-solving, which aims to enhance generalization in reasoning tasks.

Post-Training on Self-Generated Data for Reasoning Improvement Table 5 summarizes the key distinctions between our approach and existing self-improvement methods in reasoning. These methods all improve reasoning ability through post-training on self-generated data, but differ in terms of search methods, supervision granularity, search space, and generalization capabilities across tasks.

Direct Preference Optimization (DPO) Algorithms DPO (Rafailov et al., 2023) uses instance-level preference data for model optimization but has notable limitations. It struggles with multi-step reasoning tasks because it cannot effectively correct specific errors within the reasoning process (Hwang et al., 2024). Moreover, training on model-generated positive data can amplify spurious correlations from incorrect intermediate steps, leading to poor generalization (Setlur et al., 2024). Recent work proposes step-level DPO (Setlur et al., 2024; Lai et al., 2024) to address these issues by providing the fine-grained error identification needed for improving reasoning capabilities. For example, Self-Explore (Hwang et al., 2024) identifies the first incorrect step in a solution and constructs step-level preference data to guide model improvement. Unlike these heuristic methods, we propose Step-APO to fully leverage step-level reasoning data and maximize optimization potential.

Discussions around "abstract plans" and "thoughts" We conceptualize "thoughts" in works like Quiet-STaR (Zelikman et al., 2024) as incorporating various types of information, such as high-level plans, concrete solution attempts, self-correction, and reflections. Our core contribution is to systematically isolate which components within

Method	Search Method	Supervision	Search Space	Generalization
STaR (Zelikman et al., 2022)	Repeated sampling	Response-level	Concrete solutions	✗
Self-Explore (Hwang et al., 2024)	Repeated sampling	Step-level	Concrete solutions	✗
TS-LLM (Feng et al., 2023)	MCTS	Response-level	Concrete solutions	✗
AlphaMath (Chen et al., 2024)	MCTS	Response-level	Concrete solutions	✗
ALPHALLM (Tian et al., 2024)	MCTS	Response-level	Concrete solutions	✗
MCTS-DPO (Xie et al., 2024)	MCTS	Step-level	Concrete solutions	✗
PlanLearn	MCTS	Step-level	Abstract plans	✓

Table 5: Key differences between existing self-improvement methods and our approach.

Round Num	Avg Depths	Pos:Neg	Plan Pairs Count	Solution Pairs Count
Round 1	4.18	1:3.16	18742	16506
Round 2	3.80	1:1.23	24707	24633

Table 6: Statistic for the generated data in two rounds.

this stream of thought are most critical for generalization. Our findings reveal that learning from explicit, high-level problem-solving strategies ("abstract plans") is substantially more effective for transferring reasoning skills than learning from the concrete solution steps.

B Statistic for the generated data

We list the statistic for the generated data in two rounds in Table 6. Round 2 generates more correct responses, indicating a stronger policy and value model.

C Evaluation Details

In-domain Tasks For our PlanLearn, we evaluated in-domain reasoning capabilities using a zero-shot setting on the MATH and GSM8K datasets.

- **MATH**: Comprising 5,000 intricate competition-level problems, aimed at evaluating the models' capability to perform complex mathematical reasoning.
- **GSM8K**: Containing 1,320 diverse grade school math problems, designed to assess basic arithmetic and reasoning skills in an educational context.

We utilized vLLM (Kwon et al., 2023) for inference during evaluation and employed the math evaluation toolkit (Zhang et al., 2024) to assess model-generated answers. For other baseline models, results were reproduced on our machine using the configurations and codes from the original papers.

Out-of-domain Reasoning Tasks For all models, we employed few-shot prompting through the lm-evaluation-harness (Gao et al., 2024) to evaluate performance on ARC-C (25-shot), BBH (3-shot), and MMLU-stem (5-shot). Following Yue et al. (2024), we utilized 5-shot prompting to evaluate the GPQA diamond subset. Following Chen et al. (2021), we utilized zero-shot setting to evaluate performance on HumanEval.

- **HumanEval**: HumanEval is a widely used benchmark for code generation tasks. It provides descriptive prompts for each problem, prompting LLMs to generate corresponding code. It contains 164 problems.
- **ARC-C**: ARC includes questions derived from various grade-level science exams, designed to test models' ability to handle both straightforward and complex scientific queries. The challenge subset contains 1,172 test questions.
- **GPQA**: Providing "Google-proof" questions in the fields of biology, physics, and chemistry, GPQA is designed to test deep domain expertise and reasoning under challenging conditions. We use the diamond subset, which contains 198 difficult problems.
- **BIG-Bench Hard (BBH)**: Comprising 23 tasks previously identified as challenging for language models in the BIG-Bench benchmark, BBH contains a total of 6,511 challenging problems, aimed at evaluating the capa-

bilities of large language models (LLMs) in solving these tasks.

- **MMLU-STEM**: Spanning 57 subjects across multiple disciplines, MMLU evaluates the breadth and depth of a model's knowledge in a manner similar to academic and professional testing environments. We select the STEM subset, which contains 3,130 problems.

D Implementation Details

All models in our experiments were trained on 8 * NVIDIA H100 GPUs. We implement our Step-APO in Llama Factory (Zheng et al., 2024) and use Llama Factory as the training framwork. We use vLLM (Kwon et al., 2023) as the inference framework. We train all models with DeepSpeed ZeRO Stage2 (Rajbhandari et al., 2021), Flash Attention 2 (Dao, 2023). The key hyperparameter of `PlanLearn` is listed in Table 7.

Hyperparameter	Value
c_{puct}	1.5
Simulations N	200 (for round 1) or 100
Expand child nodes	5 (for root) or 3
Temperature	0.7
Max depth	6
SFT batch size	512
SFT learning rate	1e-5
SFT epochs	5 (for round 1) or 3
Step-APO batch size	64
Step-APO β	0.3
Step-APO learning rate	1e-6
Step-APO epochs	2
Solution step scaling factor	0.3
Lr scheduler type	cosine
Warmup ratio	0.1

Table 7: Key hyperparameters of `PlanLearn`.

E Prompt used in MCTS

Prompts for Round 1 and Round 2 are listed below.

Round 1 2-shot prompt

```
You are a powerful agent with
advanced reasoning and planning
capabilities. Answer the questions
as best you can.

!!!Remember:
1. Your answer should have two
sections: "Plans" and "Detailed
Implementation".
2. In the "Plans" section, you
should outline step-by-step plans
for solving the problem. These
plans might include extracting key
information, forming
sub-questions, analyzing aspects,
etc. Each step should introduce
new insights, avoid overly
abstract or generic actions. End
each step with "<endstep>".
3. In the "Detailed
Implementation" section, provide
detailed steps that correspond to
each plan, and conclude with "The
final answer is
\boxed{answer}.<endsolution>"

The following is a template for
your answer:
Question: The input question
Plans:
Plan 1: Describe the first plan
step.<endstep>
Plan 2: Describe the second plan
step<endstep>
...
Plan N: Describe the final plan
step<endstep>

Detailed Implementation:
1. Execute the first plan step
2. Execute the second plan step
...
N. Execute the final plan step
The final answer is
\boxed{answer}.<endsolution>

The following are 2 demonstration
examples.

Question: Natalia sold clips to 48
of her friends in April, and then
she sold half as many clips in May.
How many clips did Natalia sell
altogether in April and May?

Plans:
Plan 1: Analyze the total number
of clips sold in April.<endstep>
```

```
Plan 2: Calculate the number of
clips sold in May by applying the
"half as many" condition to the
number sold in April.<endstep>
Plan 3: Sum the results from April
and May to determine the overall
total of clips sold over the two
months.<endstep>

Detailed Implementation:
1. Natalia sold 48 clips in April.
2. The number of clips sold in May
is $\frac{48}{2}=24$.
3. The total number of clips sold
in April and May combined is
$48+24=72$.
The final answer is
\boxed{72}.<endsolution>

Question: If $x^2+y^2=1$, what is
the largest possible value of
$|x|+|y|$?

Plans:
Plan 1: Understand that the
equation $x^2+y^2=1$ defines a
circle centered at the origin with
a radius of 1. To maximize
$|x|+|y|$, we need to consider
points on this circle that
maximize the sum of the absolute
values of $x$ and $y$.<endstep>
Plan 2: Recognize that $|x|+|y|$
is maximized when both $|x|$ and
$|y|$ are large. The maximum sum
occurs along lines where $x$ and
$y$ contribute equally,
specifically along the lines $y=x$
and $y=-x$.<endstep>
Plan 3: Identify the points of
intersection between the lines
$y=x$ and $y=-x$ with the circle
$x^2+y^2=1$. These points are
expected to yield the maximum
value of $|x|+|y|$.<endstep>
Plan 4: Evaluate $|x|+|y|$ for the
intersection points to determine
the maximum possible
value.<endstep>
```

```
Detailed Implementation:
1. The circle $x^2+y^2=1$ is
centered at the origin with a
radius of 1. We need to find the
points on this circle that
maximize the sum $|x|+|y|$.
2. To maximize $|x|+|y|$, the sum
is largest when both $|x|$ and
$|y|$ are large. This occurs along
the lines $y=x$ and $y=-x$, where
$x$ and $y$ contribute equally to
the sum.
3. The intersection points are
$\left(\frac{1}{\sqrt{2}},\frac{1
}{\sqrt{2}}\right)$,
$\left(\frac{1}{\sqrt{2}},-\frac{
1}{\sqrt{2}}\right)$,
$\left(-\frac{1}{\sqrt{2}},\frac{
1}{\sqrt{2}}\right)$, and
$\left(-\frac{1}{\sqrt{2}},-\frac
{1}{\sqrt{2}}\right)$.
4. For these points, calculate
$|x|+|y|$. For
$\left(\frac{1}{\sqrt{2}},\frac{1
}{\sqrt{2}}\right)$,
$|x|+|y|=\sqrt{2}$. The same value
applies to the other points.
Therefore, the maximum value is
$\sqrt{2}$.
The final answer is
$\boxed{\sqrt{2}}$.<endsolution>
Now! It's your turn.
```

Round 2 XML 2-shot prompt

```
<question>
Question: Natalia sold clips to 48
of her friends in April, and then
she sold half as many clips in May.
How many clips did Natalia sell
altogether in April and May?
</question>
<plan>
<step>
Plan 1: Analyze the total number
of clips sold in April.
</step>
<step>
```

```
Plan 2: Calculate the number of
clips sold in May by applying the
"half as many" condition to the
number sold in April.
</step>
<step>
Plan 3: Sum the results from April
and May to determine the overall
total of clips sold over the two
months.
</step>
</plan>
<solution>
1. Natalia sold 48 clips in April.
2. The number of clips sold in May
is $\frac{48}{2}=24$.
3. The total number of clips sold
in April and May combined is
$48+24=72$.
The final answer is \boxed{72}.
</solution>

<question>
If $x^2+y^2=1$, what is the
largest possible value of
$|x|+|y|$?
</question>
<plan>
<step>
Plan 1: Understand that the
equation $x^2+y^2=1$ defines a
circle centered at the origin with
a radius of 1.
To maximize $|x|+|y|$, we need to
consider points on this circle
that maximize the sum of the
absolute values of $x$ and $y.
</step>
<step>
Plan 2: Recognize that $|x|+|y|$
is maximized when both $|x|$ and
$|y|$ are large. The maximum sum
occurs along lines where $x$ and
$y$ contribute equally,
specifically along the lines $y=x$
and $y=-x.
</step>
<step>
```

```
Plan 3: Identify the points of
intersection between the lines
$y=x$ and $y=-x$ with the circle
$x^2+y^2=1$. These points are
expected to yield the maximum
value of $|x|+|y|.
</step>
<step>
Plan 4: Evaluate $|x|+|y|$ for the
intersection points to determine
the maximum possible value.
</step>
</plan>
<solution>
1. The circle $x^2+y^2=1$ is
centered at the origin with a
radius of 1. We need to find the
points on this circle that
maximize the sum $|x|+|y|$.
2. To maximize $|x|+|y|$, the sum
is largest when both $|x|$ and
$|y|$ are large. This occurs along
the lines $y=x$ and $y=-x$, where
$x$ and $y$ contribute equally to
the sum.
3. The intersection points are
$\left(\frac{1}{\sqrt{2}},\frac{1
}{\sqrt{2}}\right)$,
$\left(\frac{1}{\sqrt{2}},-\frac{
1}{\sqrt{2}}\right)$,
$\left(-\frac{1}{\sqrt{2}},\frac{
1}{\sqrt{2}}\right)$, and
$\left(-\frac{1}{\sqrt{2}},-\frac
{1}{\sqrt{2}}\right)$.
4. For these points, calculate
$|x|+|y|$. For
$\left(\frac{1}{\sqrt{2}},\frac{1
}{\sqrt{2}}\right)$,
$|x|+|y|=\sqrt{2}$. The same value
applies to the other points.
Therefore, the maximum value is
$\sqrt{2}$.
The final answer is
$\boxed{\sqrt{2}}.
</solution>
```