SPPD: Self-training with Process Preference Learning Using Dynamic Value Margin

Anonymous ACL submission

Abstract

Enhancing the numerical and logical reasoning capabilities of Large Language Models (LLMs) has become a prominent research focus. Existing approaches exhibit notable limitations: inference-phase techniques, such as Chain of Thought, depend on prompt engineering and pretrained knowledge; sentence-level Supervised Fine-Tuning (SFT) and Direct Preference Optimization (DPO) struggle to ensure stepwise mathematical correctness and often rely on model distillation or human annotations; Reinforcement Learning (RL) methods entail high GPU memory consumption and training instability. To overcome these challenges, we propose **S**elf-training with **P**rocess **P**reference learning using **D**ynamic value margin (**SPPD**). SPPD formulates reasoning as a process-based Markov Decision Process (MDP), leveraging the Bellman optimality equation to derive a **dynamic value margin** for step-level preference optimization. It further incorporates tree-based self-sampling of model responses, **eliminating the need for distillation.** We theoretically establish that SPPD is **equivalent to on-policy policy gradient methods** under constrained reward functions. Experimental results on 7B-scale models show consistent superiority across both in-domain and out-of-domain mathematical benchmarks. Our code is publicly available at https://anonymous.4open.science/r/SPPD-DCDD.

1 Introduction

Recent advancements in O-series models (OpenAI, 2024) have significantly enhanced the mathematical reasoning capabilities of large language models (LLMs), positioning numerical and logical reasoning as a focal point in current research (Chen et al., 2023; Yu et al., 2023; Jimenez et al., 2023; Shao et al.; Liao et al., 2024b; Lai et al., 2024; Guo et al., 2025).

To enhance model reasoning during the inference phase, techniques such as Chain of Thoughts (CoT) prompts (Wei et al., 2022), Tree of Thoughts (ToT) (Yao et al., 2024), Best of N (BoN) (Zheng et al., 2024a; Yuan et al., 2024), and Monte Carlo Tree Search (MCTS) (Feng et al., 2023; Zhang et al., 2024a) are employed. These methods rely on prompt selection and pretrained knowledge but do not involve policy model training. Supervised Fine-Tuning (SFT) (Zhang et al., 2024a; Feng et al., 2023) and Direct Preference Optimization (DPO) (Rafailov et al., 2024b,a), which leverage human annotations or AI feedback, improve sentence-level reasoning but struggle to ensure stepwise correctness in mathematical contexts. These approaches often require manual selection or support from stronger models like STILL-2 (Min et al., 2024) and Skywork-o1-open (Skywork, 2024b). Achieving further performance improvements without distillation using the strongest available models remains challenging. Although RL-based methods, such as Proximal Policy Optimization (PPO) (Schulman et al., 2017), Group Relative Policy Optimization (GRPO) (Shao et al.; Guo et al., 2025), and Reinforcement Fine-Tuning (RFT) (Luong et al., 2024), address some limitations, they are resource-intensive.

To address these challenges, we introduce Self-training with Process Preference learning using Dynamic value margin (SPPD). Unlike SFT and DPO, SPPD optimizes preferences at the step level via dynamic value margins, eliminating the need for distillation. Specifically, SPPD utilizes a process-based Markov Decision Process (MDP) and the Bradley-Terry (BT) Model (Bradley and Terry, 1952), employing the Bellman optimality equation (Barron and Ishii, 1989) and an online RL objective modeled with MDP (Rafailov et al., 2024a). This approach enables step-wise DPO without relying on stronger models for data distillation. Instead, it employs tree search methods that sample trajectories solely from the model's responses and logits scores. Additionally, a strategy combining SFT and

DPO based on Process Reward Model(PRM) rejection sampling enhances training efficiency, progressively refining reasoning capabilities. Theoretically, our method is equivalent to an on-policy policy gradient method under specific reward constraints.

Experimental results demonstrate that SPPD achieves great improvements across various architectures and datasets, outperforming many open-source and some closed-source models. Our contributions include:

- We derive SPPD utilizing the Bellman optimality equation and an online RL objective modeled with MDP;
- We design a step-level self-sampling scheme without stronger model distillation;
- We prove that SPPD is theoretically equivalence to on-policy policy gradient optimization .

2 Related Work

Enhance Reasoning Capability of LLMs. Recent research has concentrated on enhancing the reasoning capabilities of large language models (LLMs), categorized into inference and post-training phases. During inference, methods such as Chain of Thoughts (CoT) prompts (Wei et al., 2022; Yao et al., 2024) stimulate inherent model reasoning, whereas strategies like self-consistency (Yuan et al., 2024; Wang et al., 2022) and tree search (Feng et al., 2023; Zhang et al., 2024a) guide more accurate decoding without additional training. In the post-training phase, SFT (Feng et al., 2023) and DPO (Rafailov et al., 2024b,a) leverage human-curated trajectories or distillation from stronger models (Min et al., 2024), improving weaker models' reasoning performance. RL paradigms, including PPO (Schulman et al., 2017), GRPO (Guo et al., 2025; Shao et al.), and ReFT (Luong et al., 2024), address limitations but introduce GPU memory and stability challenges.

Step-Level Direct Preference Optimization. To optimize reasoning at the step level, CPO (Zhang et al., 2024b) aligns CoT reasoning paths with Tree of Thoughts (ToT) preferences, using prompt-based control which may affect generation quality. Step-DPO (Lai et al., 2024) optimizes individual reasoning steps, relying on GPT-4 for correctness evaluation, an expensive approach prone to bias. TPO (Liao et al., 2024b) proposes learning from ranked preference lists with adaptive step rewards, yet this can exacerbate catastrophic forgetting and reward value imbalance in the preference tree.

3 Preliminaries

In this section, we first define the step-level MDP in natural language processing. Subsequently, based on this step-level MDP, we reformulate the original Reinforcement Learning from Human feedback (RLHF) objective and derive its optimal (fixed-point) solution to the maximum causal entropy problem.

Step-Level MDP in LLMs. We define the step-level MDP in natural language processing as the quintuple: $\mathcal{M} = (\mathcal{A}, \mathcal{S}, f, r, \rho_0)$, where $\mathcal{A}$ denotes the action space, consisting of reasoning steps a_t; $\mathcal{S}$ denotes the state space, representing the sequence of the problem and preceding actions: $s_t = s_0 \,|\, a_1 \,|\, a_2 \,|\, \ldots \,|\, a_t$, where $|$ represents string concatenation and s_0 is the initial problem. The selection of each action a_t is conditioned on the current state s_{t-1}; $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the state transition function, defined as $f(s, a) = s \,|\, a$; $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, indicating the immediate reward obtained by performing action a in state s; ρ_0 denotes the distribution over initial problems.

RLHF Objective under the Step-Level MDP. The original RLHF objective models trajectory-based rewards as a bandit problem (Ouyang et al., 2022; Zhao et al., 2024). However, such sparse reward formulations are generally inadequate for policy learning, particularly in complex domains such as mathematical reasoning, where reward signals are often insufficient or ambiguous (Riedmiller et al., 2018; Wilcox et al., 2022). To address this limitation, we adopt a step-level MDP formulation and adapt the RLHF objective accordingly (Rafailov et al., 2024a):

$$\max_{\pi_\theta} \mathbb{E}_{a_t \sim \pi_\theta(\cdot|\mathbf{s}_t)} \Bigg[\sum_{t=0}^{T} r(\mathbf{s}_t, \mathbf{a}_t) + \underbrace{\beta \log \pi_{\text{ref}}(\mathbf{a}_t \mid \mathbf{s}_t)}_{\text{KL penalty}} + \beta \mathcal{H}(\pi_\theta) \mid \mathbf{s}_0 \sim \rho(\mathbf{s}_0) \Bigg], \quad (1)$$

where π_θ denotes the parameterized large language policy model, π_{ref} is the reference model, β controls the degree of deviation from π_{ref}, and $\mathcal{H}(\pi_\theta)$ represents the entropy of π_θ. This optimization framework corresponds to the principle of **Maximum Causal Entropy**. Ziebart (2010) have shown

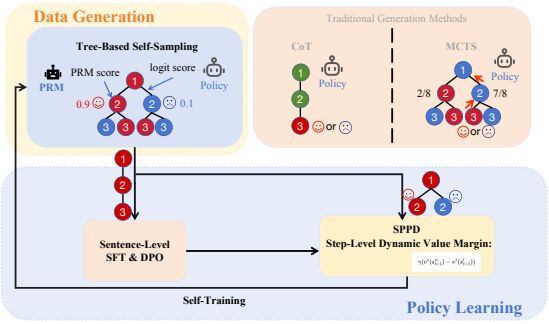

Figure 1: The framework of SPPD: unlike CoT and MCTS, Tree-Based Self-Sampling generates step trajectories with common prefixes and preserves the output distribution of the policy. The former provides step preference signals for SPPD, while the latter theoretically ensures consistency with on-policy gradient methods, thereby enabling self-enhancement of the model's reasoning capabilities.

that Equation (1) admits a fixed-point solution π^*, given by:

$$\pi^*(a_t \mid s_t) = \pi_{\text{ref}}(a_t \mid s_t) \exp\left(\frac{Q^*(s_t, a_t) - V^*(s_t)}{\beta}\right), \quad (2)$$

where $V^*(s_t)$ serves as the normalization term (i.e., partition function) for the distribution π^*, and $Q^*(s_t, a_t)$ denotes the expected cumulative reward starting from state-action pair (s_t, a_t) under policy π^*.

4 Method

In this section, we first introduce a tree-based self-sampling method to generate step trajectories sharing common prefixes without distillation. We further incorporate sentence-level SFT and DPO using the PRM, aiming to enhance the smoothness and effectiveness of model training. Finally, step preference learning with dynamic value margins is proposed and further refined based on reward equivalence.

4.1 Tree-Based Self-Sampling on LLMs

Traditional reasoning algorithms based on token-level decoding are generally unable to generate reasoning trajectories with shared prefixes. To address this limitation, we employ a tree-structured reasoning approach, as illustrated in Figure 1. The process consists of four stages: *Selection*, *Expansion*, *Collection*, and *Scoring*.

During the selection phase, at each state s_t, we compute the average log probability score for each child node a_t as follows:

$$s(a_t \mid s_t) = \frac{1}{|a_t|} \sum_{i=0}^{|a_t|} \log \pi_{\text{infer}}(a_{t,i} \mid s_t, a_{t,<i}),$$

where $|a_t|$ denotes the token length of the current step, $a_{t,<i}$ represents the first $i-1$ tokens of a_t, and π_{infer} refers to the inference model's output distribution (i.e., the policy in RL). In practice, we set $\pi_{\text{infer}} = \pi_{\text{ref}}$. Subsequently, we normalize the scores across all child nodes and perform sampling to select a path from the root node to a leaf node.

If a selected node is non-terminal and has no children, we expand it by generating C possible reasoning steps. After repeating this process K times, we traverse the constructed prefix tree and collect all complete reasoning paths along with their final answers. Finally, we apply the Policy Reward Model (PRM) to evaluate each step in the trajectory, yielding the final step-level dataset:

$$\mathcal{D}_{\text{step}} = \{(s_0^{(i)}, s_t^{(i,j)}, v_t^{(i,j)}) \mid i \in [N], j \in [K], t \in |\tau^{(i,j)}|\},$$

where N is the number of problems, and $v_t^{(i,j)}$ denotes the PRM score assigned to the state $s_t^{(i,j)}$, which is the t-th step in the j-th prefix sequence of problem $s_0^{(i)}$.

4.2 PRM-Enhanced SFT & DPO

We incorporate curriculum learning to streamline the model's training, initially focusing on sentence-level strategies. This approach utilizes PRM feedback on sampled trajectories for rejection sampling, enhancing the model through supervised and preference learning. Specifically, positive and negative sample trajectories are defined as:

$$\tau_+^{(i)} = \max_{j \in [K]} \min_{v_t^{(i,j)}} \mathcal{D}_{\text{step}}^+,$$

$$\tau_-^{(i)} = \min_{j \in [K]} \min_{v_t^{(i,j)}} \mathcal{D}_{\text{step}}^-.$$

Here, $\mathcal{D}_{\text{step}}^+$ and $\mathcal{D}_{\text{step}}^-$ denote trajectories ending in correct and incorrect answers, respectively. In the SFT phase, we minimize next-token prediction loss on $\tau_+^{(i)}$. For DPO, positive samples from $\{\tau_+^{(i)}\}_{i=1}^N$ and negative samples from $\{\tau_-^{(i)}\}_{i=1}^N$ are selected to construct preference samples for sentence-level DPO. Notably, both SFT and DPO aim to preliminarily enhance the model's reasoning capabilities, preparing it for subsequent fine-grained preference learning at the step level.

4.3 Process Preference Learning with Dynamic Value Margin

We derive process preference learning with a dynamic value margin based on the optimal Bellman equation and reinterpret traditional step DPO (Lai et al., 2024) from a novel theoretical perspective.

Lemma 4.1 (Optimal Step Reward Function)**.** *Under the step MDP formulation defined in Section 3 and given the fixed solution to the maximum causal entropy problem (Equation (2)), the optimal step reward function is expressed as:*

$$r(s_t, a_t) = \underbrace{\beta \log \frac{\pi^*(a_t|s_t)}{\pi_{ref}(a_t|s_t)}}_{\textit{Implicit Reward}} - \underbrace{(V^*(s_{t+1}) - V^*(s_t))}_{\textit{Value Gain}}. \tag{3}$$

The proof of Lemma 4.1 is provided in Appendix E.1. Equation (3) decomposes the immediate reward into two components: the model's **implicit reward**, derived from the policy ratio relative to the reference policy, and the **value gain**, representing the change in the optimal value function between consecutive states. Given step-level preference pairs (s_t, a_t^w, a_t^l), we define the optimal preference distribution as:

$$p^*(a_t^w \succ a_t^l) = \sigma\left(r(s_t, a_t^w) - r(s_t, a_t^l)\right), \tag{4}$$

where $\sigma(x)$ denotes the sigmoid function. This formulation implies that action preferences depend on the difference in their corresponding rewards.

Next, we derive the step DPO loss incorporating a dynamic value margin.

Theorem 4.2 (Step DPO Loss with Dynamic Value Margin)**.** *If the objective is to minimize the Kullback–Leibler divergence between the empirical preference distribution p_{data} (as defined in Section 4.1) and the model's current preference distribution p_θ (See Equation (12)), under sampling from π_{ref}, the resulting loss function is:*

$$\begin{aligned}\mathcal{L}_{\textit{step-dpo}} = -\mathbb{E}_{a_t^w, a_t^l \sim \pi_{ref}(\cdot|s_t)}\Big[\\ \log \sigma\big(\beta h_\theta(a_t^w, a_t^l) \\ - (V^*(s_{t+1}^w) - V^*(s_{t+1}^l))\big)\Big],\end{aligned} \tag{5}$$

where $h_\theta(a_t^w, a_t^l) = \log \frac{\pi_\theta(a_t^w|s_t)}{\pi_{ref}(a_t^w|s_t)} - \log \frac{\pi_\theta(a_t^l|s_t)}{\pi_{ref}(a_t^l|s_t)}$. Here, p_{data} represents the dataset-induced preference distribution, which follows a conditional point-mass distribution. For any instance (s_t, a_t^w, a_t^l), it holds that: $p_{data}(a_t^w \succ a_t^l|s_t) = 1, \quad p_{data}(a_t^l \succ a_t^w|s_t) = 0$.

The proof is detailed in Appendix E.2.

In conventional step DPO (Lai et al., 2024), the value function prediction is assumed to be zero at each time step. In contrast, our formulation incorporates the **value gain** from Equation (3), specifically the term $V^*(s_{t+1}^w) - V^*(s_{t+1}^l)$, which reflects the difference in optimal value function estimates for the preferred and dispreferred next states. This introduces a dynamic value margin into the step DPO loss, varying across state transitions rather than being constant. In practice, we approximate the optimal value function using a PRM score. A more rigorous theoretical analysis will be presented in Section 5.

Reward Equivalence. To enhance the controllability of the optimization process, we incorporate the principle of reward equivalence into the derivation.

Lemma 4.3 (Reward Equivalence; Rafailov et al. (2024a))**.** *Two reward functions r and r' are equivalent if and only if there exists a potential function $\Phi : \mathcal{S} \to \mathbb{R}$ such that:*

$$r(s_t, a_t) = r'(s_t, a_t) + \Phi(f(s_t, a_t)) - \Phi(s_t).$$

In Equation (3), the potential function corresponds to the optimal value function, i.e., $\Phi(s) = V^*(s)$. Scaling this function via $\Phi'(s) = \gamma\Phi(s)$ preserves its validity as a potential function. Consequently, we derive an equivalent reward expression:

$$r^\gamma(s_t, a_t) = r(s_t, a_t) + \gamma\Phi(f(s_t, a_t)) - \gamma\Phi(s_t).$$

Revisiting the derivation in Section 4.3, the modified loss becomes:

$$\begin{aligned}\mathcal{L}^\gamma_{\text{step-dpo}} = -\mathbb{E}_{a_t^w, a_t^l \sim \pi_{\text{ref}}(\cdot|s_t)}\Big[\\ \log \sigma\big(\beta h_\theta(a_t^w, a_t^l) \\ - \gamma(V^*(s_{t+1}^w) - V^*(s_{t+1}^l))\big)\Big].\end{aligned} \tag{6}$$

Remark. *While Rafailov et al. (2024a) establishes that all reward-equivalent models belong to the same class—including the original step DPO when $\gamma = 0$, introducing γ allows greater control over the optimization trajectory. Empirical validation of this effect is provided in Section 6.3.*

5 Theoretical Analysis

In this section, we establish the equivalence between offline step DPO and online policy gradient under a specific reward formulation.

Lemma 5.1 (Online Policy Gradient on π_θ^p (see Definition C.1); Lin and Zhou (2019))**.** *For any MDP, the expected long-term reward under π_θ^p is given by $J(\theta) = \sum_\tau \pi_\theta^p(\tau) r(\tau)$, where $r(\tau)$ denotes the cumulative reward along trajectory τ. The corresponding policy gradient is:*

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta^p} \left[r(\tau) \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta^p(a_t^w | s_t) \right]. \tag{7}$$

Theorem 5.2 (Equivalence Between Offline Step DPO and Online Policy Gradient)**.** *Define the trajectory reward in Equation (7) as:*

$$r(\tau) = \prod_{i=0}^{t+1} \frac{\pi_{ref}(a_t|s_t)}{\pi_\theta^p(a_t|s_t)}.$$

*Let the **offline every-step preference loss** be defined as:*

$$\mathcal{L}_{\textit{every-step}} = \mathbb{E}_{\tau \sim \pi_{ref}^p} \left[-\sum_{t=0}^{T-1} \log \pi_\theta^p(a_t^w | s_t) \right]. \tag{8}$$

Then, the following identity holds:

$$\nabla_\theta J(\theta) = -\nabla_\theta \mathcal{L}_{\textit{every-step}}.$$

The proof is provided in Appendix E.3.

Remark. *It is evident that $\mathcal{L}_{\textit{every-step}}$ (Equation (8)) corresponds to $\mathcal{L}_{\textit{step-dpo}}$ (Equation (5)) when the sampling tree has two branches ($C = 2$) and preference comparisons are made at every step.*

Theorem 5.2 establishes that, under the specified reward definition, optimizing the gradient of the offline preference loss is mathematically equivalent to performing policy gradient updates on the preference decoding model in an online setting.

Furthermore, under the defined reward $r(\tau)$, large values indicate low trajectory probabilities under π_θ^p. Consequently, during optimization, greater emphasis will be placed on minimizing the loss associated with such trajectories, effectively guiding the model toward higher-confidence preference predictions.

6 Experiments

6.1 Setup

Datasets. We sample a total of 10,000 training prompts from the GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021) datasets, with proportions of 40% and 60%, respectively. The base models used are Qwen2.5-7B-Base (Yang et al., 2024) and Llama3.1-8B-Instruct (Meta@AI, 2024). Step-level preference data $\mathcal{D}_{\text{step}}$ is generated using the method described in Section 4.1, with Skywork-o1-Open-PRM-Qwen-2.5-7B (Skywork, 2024a) as the preference reward model (PRM). Further details on data format and PRM scoring distribution are provided in Appendix A and Appendix B.

Evaluation. The maximum generation length during inference is set to 2048. The test set includes both in-domain and out-of-domain subsets: GSM8K, MATH500 (in-domain), and Gaokao2023 (Liao et al., 2024a), OCW Course (OCW) (Lewkowycz et al., 2022), and OE-TO-MATH-COMP from OlympiadBench (He et al., 2024) (out-of-domain). The evaluation methods are as follows:

1) **Greedy-CoT**: Inference using greedy decoding with CoT prompting; performance measured by pass@1.

2) **MAJ@N**: Perform N independent inferences using CoT prompting; final answer selected via majority voting.

3) **ORM_VOTE@N**: Repeat inference N times with CoT prompting; use Skywork-o1-Open-PRM-Qwen-2.5-7B as an output reward model (ORM) to score each response. Aggregate scores across identical answers and select the highest-scoring one.

4) **ORM_MAX@N**: Similar to ORM_VOTE@N, but without aggregating scores for duplicate answers; the highest-scoring individual response is selected directly. Additional evaluation protocols are detailed in Appendix D.

Implementation. During data generation, we perform tree sampling with $K = 64$ rollouts per question, where each node branches into $C = 2$ candidate steps. For step-level preference pair selection, only pairs with a PRM score difference exceeding 0.5 are retained to reduce noise (PRM scores range from 0 to 1). In the SFT phase, optimization is conducted using the Adam optimizer with a learning rate of 5×10^{-6}, whereas in both DPO and dynamic margin step-DPO phases, the SGD optimizer with a learning rate of 1×10^{-5} is employed. Learning rate decay follows a cosine schedule in all cases. The temperature parameter β is fixed at 0.1 for both DPO and step-DPO. The scaling factor γ in step-DPO is selected from the set $\{0.1, 0.5, 1.0, 2.0, 5.0\}$. All experiments are conducted on 8 Nvidia 80GB H800 GPUs.

Model	Size	Open	General	MATH500	GSM8k
Claude-3-Opus*	-	✗	✔	60.1	95.0
GPT4-1106 (Achiam et al., 2023)*	-	✗	✔	64.3	91.4
GPT4o-0513*	-	✗	✔	76.6	95.8
o1 (OpenAI, 2024)*	-	✗	✔	94.8	-
Qwen2-7B-Instruct-Step-DPO (Lai et al., 2024)	7B	✔	✗	55.0	85.4
DeepSeek-MATH-7B-Instruct (Shao et al.)	7B	✔	✗	44.4	80.9
OpenMath2-Llama3.1-8B (Toshniwal et al., 2024)	8B	✔	✗	65.4	90.1
Llama3.1-8B-Instruct (Meta@AI, 2024)	8B	✔	✔	47.0	82.6
Qwen2.5-7B-Instruct (Yang et al., 2024)	7B	✔	✔	72.8	89.3
Qwen2.5-7B-Base	7B	✔	✔	60.0	82.3
+SFT-PRM	7B	✔	✗	64.4	88.1
+SFT-PRM & DPO-PRM	7B	✔	✗	68.2	89.3
+SPPD	7B	✔	✗	**71.0** +2.8%	**89.8** +0.5%
+SPPD+MAJ@64	7B	✔	✗	76.4	93.2
+SPPD+ORM_MAX@64	7B	✔	✗	74.0	**94.9**
+SPPD+ORM_VOTE@64	7B	✔	✗	**79.0**	94.7
+SPPD-Stage2	7B	✔	✗	**72.2** +4.0%	**90.3** +1.0%
+SPPD-Stage2+MAJ@64	7B	✔	✗	78.6	93.6
+SPPD-Stage2+ORM_MAX@64	7B	✔	✗	78.0	**95.0**
+SPPD-Stage2+ORM_VOTE@64	7B	✔	✗	**80.4** +12.2%	94.6 +5.3%

Table 1: Main Results. * denotes we use officially reported results. **SFT-PRM** refers to using the PRM to select the correct sequence among 64 sampled responses, and then performing SFT. **DPO-PRM** refers to using the PRM to select the positive sample and the negative sample, and then performing DPO (see Section 4.2). **SPPD-Stage2** indicates that we iterate the SPPD method twice, noting that the second stage ignores SFT and DPO training.

6.2 Main Result

Compared to the Base Model. Our method achieves substantial improvements without relying on responses from stronger models for distillation, as shown in Table 1. Using SFT-PRM, we obtain performance gains of 4.4% and 5.8% on the in-domain datasets MATH and GSM8K, respectively. With DPO-PRM, the improvements are 3.8% and 1.2%, respectively. Building upon this, SPPD further enhances reasoning capabilities through step-level dynamic optimization guided by PRM signals, yielding additional gains of 2.8% and 0.5% on the same datasets. During inference, increasing computational resources and applying the **ORM_VOTE** aggregation strategy reveal the model's full potential, achieving accuracies of 79% and 94.7% on MATH and GSM8K, respectively, outperforming existing models of comparable size.

Compared to the other methods. We compared the performance of SPPD with RL (PPO, GRPO), preference learning (SimPO), and distillation algorithms. All experiments were trained on the same dataset (see Section 6.1). The results are shown in Table 2. The experimental results demonstrate that:

- Compared to online RL algorithms (GRPO & PPO), SPPD achieves comparable performance on ID (MATH500 & GSM8k) tasks while exhibiting better generalization on OOD (GaoKao2023) tasks than GRPO and PPO, highlighting its effectiveness and robustness.
- Compared to offline preference learning (SimPO (Meng et al., 2024)) and distillation methods, our approach proves more effective and robust in both ID and ODD evaluations.

Continued Gains in the Second Stage. After exhausting the training data generated by the base

model in the first stage, we follow the principles of offline RL and update the policy model's sampling trajectories. Using the best-performing model from the first stage as the new policy model, we reiterate the training process to obtain SPPD-Stage2. Compared to SPPD, SPPD-Stage2 achieves further improvements of 1.2% and 0.5% on MATH and GSM8K, respectively. These results confirm the effectiveness of policy model updating and underscore the robustness of the SPPD framework.

Methods	MATH500	GSM8k	GaoKao2023	Avg
Qwen2.5-7B-Base	60.0	82.3	48.0	63.4
+SPPD	**72.2**	**90.4**	**56.8**	**73.1**
+GRPO(Guo et al., 2025)	71.1	90.2	54.3	71.9
+PPO (Schulman et al., 2017)	71.9	89.8	55.2	72.3
+SimPO (Meng et al., 2024)	70.3	88.7	53.3	70.8
+Distillation*	69.5	88.0	53.1	70.2
Llama3.1-8B-Instruct	46.2	81.2	35.1	54.1
+SPPD	58.2	**88.5**	**42.1**	**69.2**
+GRPO(Guo et al., 2025)	**58.4**	88.1	41.5	62.7
+PPO (Schulman et al., 2017)	57.9	88.7	42.2	62.6
+SimPO (Meng et al., 2024)	56.9	87.5	41.1	61.8
+Distillation*	55.4	87.1	40.2	60.9

Table 2: Compare to other methods. *In the distillation experiments, we performed supervised fine-tuning on the model using data generated by Qwen2.5-7B-Instruct.

6.3 Ablation Study

Different Base Model. We assess the effectiveness of the SPPD method across two distinct base models: Llama3.1-8B-Instruct and Qwen2.5-7B-Instruct. As Instruct models are already optimized at the sentence level, we omit PRM-SFT and PRM-DPO training and instead directly employ model-generated trajectories for step-level DPO training with dynamic value margin. Results are summarized in Table 3, showing that SPPD improves performance by 4.6% and 3.6% on MATH and GSM8K, respectively, for Llama3.1-8B-Instruct, and by 2.2% and 0.8%, respectively, for Qwen2.5-7B-Instruct. These results demonstrate the robustness of SPPD across different base models.

Effectiveness of Dynamic Value Margin. In Section 4.3, we formulate the dynamic value margin within a Markov Decision Process (MDP) framework, deriving a step-level DPO method with mathematically grounded, dynamically adjusted margins. To validate the efficacy of this approach, we conduct ablation studies using Qwen2.5-7B-Base and Llama3.1-8B-Instruct as base models, followed by PRM-SFT and PRM-DPO training.

Model	MATH500	GSM8K
Llama3.1-8B-Instruct	46.6	81.2
+SPPD	51.2 +4.6%	84.8 +3.6%
+SPPD+MAJ@64	58.2	88.5
+SPPD+ORM_MAX@64	**67.0**	**92.0**
+SPPD+ORM_VOTE@64	66.4 +19.8%	90.7 +9.5%
Qwen2.5-7B-Instruct	72.8	89.3
+SPPD	75.0 +2.2%	91.1 +0.8%
+SPPD+MAJ@64	80.6	93.4
+SPPD+ORM_MAX@64	77.0	**95.2**
+SPPD+ORM_VOTE@64	**82.2** +9.4%	94.6 +5.3%

Table 3: Result on Llama3.1-8B-Instruct and Qwen2.5-7B-Instruct.

We compare SPPD against two variants: no-margin step DPO ($\gamma = 0$) and fixed-margin step DPO. Results are reported in Table 7.

The results show that fixed-margin step DPO outperforms no-margin step DPO, demonstrating that margin adjustment enhances step-level preference learning. Moreover, on MATH and GSM8K, SPPD surpasses fixed-margin step DPO, achieving improvements of 0.9% and 0.31% for Qwen2.5-7B-Base, respectively, 2.0% and 1.3% for Llama3.1-8B-Instruct, respectively. Moreover, when measured using the **MAJ_VOTE@64**, SPPD also demonstrates better reasoning capabilities. These gains arise from our modeling of value model score differences between preference pairs, enabling dynamic margin adaptation during preference learning. This mechanism enhances the reliability of step-level training and mitigates overfitting risks.

Different PRMs. To examine the influence of different Preference Reward Models (PRMs) on the performance of **SPPD**, we selected several PRMs: *Skywork-o1-Open-PRM-Qwen-2.5-7B* (Skywork, 2024a), *Skywork-o1-Open-PRM-Qwen-2.5-1.5B* (Skywork, 2024a), and *Qwen2.5-Math-7B-PRM800K* (Zheng et al., 2024b). According to Table 6 in Zheng et al. (2024b), *Qwen2.5-Math-7B-PRM800K* outperforms *Skywork-o1-Open-PRM-Qwen-2.5-7B* in process evaluation on MATH and GSM8K, whereas *Skywork-o1-Open-PRM-Qwen-2.5-1.5B* underperforms compared to its 7B counterpart on the same benchmarks. The

PRM	Methods	MATH500	GSM8k
Skywork-PRM-7B	PRM-SFT	64.4	88.1
	PRM-SFT&PRM-DPO	68.2	89.3
	SPPD	**71.0**	**89.8**
Qwen-PRM-7B	PRM-SFT	63.9	88.4
	PRM-SFT&PRM-DPO	68.6	89.6
	SPPD	**71.8**	**90.3**
Skywork-PRM-1.5B	PRM-SFT	62.1	87.5
	PRM-SFT&PRM-DPO	68.1	88.9
	SPPD	**70.5**	**89.3**

Table 4: SPPD on different PRMs scoring. Skywork-PRM-7B means *Skywrok-o1-Open-PRM-Qwen-2.5-7B*(Skywork, 2024a), Skywork-PRM-1.5B means *Skywrok-o1-Open-PRM-Qwen-2.5-1.5B*(Skywork, 2024a) and *Qwen-PRM means Qwen2.5-Math-7B-PRM-800k*(Zheng et al., 2024a) .

performance of SPPD using different PRMs is summarized in Table 4.

We observe that with *Qwen2.5-Math-7B-PRM800K*, SPPD generates higher quality process supervision signals, leading to enhanced generalization and robustness across reasoning steps. Notably, even when using the less capable *Skywork-o1-Open-PRM-Qwen-2.5-1.5B*, SPPD achieves comparable reasoning performance, demonstrating its robustness to variations in PRM quality.

SPPD-Stage2 w/ or w/o PRM-SFT and PRM-DPO. In our experiments, we omit the SFT-PRM

Methods	MATH500	GSM8k
Qwen2.5-7B-Base		
SPPD-Stage1	**71.0**	**89.8**
SFT-PRM-Stage2	69.1	88.4
DPO-PRM-Stage2	68.3	87.6
Llama-8B-Instruct		
SPPD-Stage1	**51.2**	**84.8**
SFT-PRM-Stage2	49.8	84.1
DPO-PRM-Stage2	49.5	83.9

Table 5: SPPD-Stage2 with PRM-SFT and PRM-DPO. SFT-PRM-Stage2 means that it only runs SFT-PRM after the first stage and SFT-PRM-Stage2 means that it runs SFT-PRM & DPO-PRM after the first stage.

and DPO-PRM steps during the second stage. We emphasize that SFT-PRM and DPO-PRM operate at a coarse-grained level, primarily serving to initialize and strengthen the model's reasoning abilities for subsequent step-level preference learning. Continuing these operations in the second stage would risk overfitting, as confirmed by experimental results shown in Table 5. During this phase, SFT-PRM and DPO-PRM training exhibited signs of overfitting, whereas SPPD maintained strong generalization performance, demonstrating its superior robustness.

More Supplementary experiments is shown in Appendix F.

Model	GaoKao2023	OCW	OlympaidBench*
Qwen2.5-7B-Base	48.0	6.3	20.5
+SFT-PRM	52.2	19.1	22.8
+SFT-PRM & DPO-PRM	55.0	16.1	23.7
+SPPD	**56.8** +1.8%	**20.0** +4.8%	**26.1** +2.4%
+SPPD+MAJ@64	62.6	29.4	43.3
+SPPD+ORM_MAX@64	63.4	28.3	41.4
+SPPD+ORM_VOTE@64	**64.4** +9.4%	**30.9** +14.8%	**45.4** +21.7%

Table 6: Result on out-of-domain test datasets. OlympaidBench* denotes we only use OlympaidBench-OE-TO-Math-COMP test dataset.

Methods	Margin	MATH500	GSM8k
Qwen2.5-7B-Base			
SPPD (ours)	Dynamic	**71.0**	**89.8**
Step-DPO	0	69.6	89.40
Step-DPO-fix-margin	γ^*	70.1	89.4
SPPD+**MAJ@64** (ours)	Dynamic	**76.4**	**93.2**
Step-DPO+**MAJ@64**	0	64.2	82.3
Step-DPO-fix-margin+**MAJ@64**	γ^*	75.0	93.1
LLama3.1+8B-Instruct			
SPPD (ours)	Dynamic	**51.2**	**84.8**
Step-DPO	0	48.8	83.2
Step-DPO-fix-margin	γ^*	49.2	83.5
SPPD+**MAJ@64** (ours)	Dynamic	**58.2**	**88.5**
Step-DPO+**MAJ@64**	0	55.3	86.4
Step-DPO-fix-margin+**MAJ@64**	γ^*	56.9	87.2

Table 7: SPPD vs fixed margin step DPO on Qwen2.5-7B-Base and Llama3.1-8B-Instruct. γ^* represents $\gamma(V^*(s^w_{t+1}) - V^*(s^l_{t+1})) = \gamma^*$ in Formula 6.

7 Conclusion

In this work, we propose SPPD, a self-training with process preference learning using dynamic value margin. SPPD utilizes the Bellman optimality equation and the online RL objective modeled with MDP and designs a step-level tree self-sampling scheme without any distillation. Moreover, we propose a SFT and DPO scheme using PRM for rejection sampling, making the training of SPPD smothor and more effective. Finally, we theoretically demonstrate that under specific reward constraints, our method is equivalent to on-policy policy gradient optimization.

Limitations

Several limitations remain in our current work. Firstly, our work relies on the effectiveness of PRM, and studies have shown that PRM's performance varies across different policy models and task scenarios; some PRMs may fail under specific tasks (Zheng et al., 2024a). This work neglects the updates of PRM. As policy is continuously iterated, PRM faces the risk of becoming ineffective. Additionally, both PPO and GRPO are modeled based on bandit, and how to integrate MDP modeling with on-policy methods remains an important subject for future research.

References

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

EN Barron and H Ishii. 1989. The bellman equation for minimizing the maximum cost. *NONLINEAR ANAL. THEORY METHODS APPLIC.*, 13(9):1067–1090.

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.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2023. Teaching large language models to self-debug. *arXiv preprint arXiv:2304.05128*.

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

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

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

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

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

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2023. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*.

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 preprint arXiv:2406.18629*.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. 2022. Solving quantitative reasoning problems with language models, 2022. *URL https://arxiv. org/abs/2206.14858*.

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

Weibin Liao, Xu Chu, and Yasha Wang. 2024b. Tpo: Aligning large language models with multi-branch & multi-step preference trees. *arXiv preprint arXiv:2410.12854*.

Kaixiang Lin and Jiayu Zhou. 2019. Ranking policy gradient. *arXiv preprint arXiv:1906.09674*.

Trung Quoc Luong, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. 2024. Reft: Reasoning with reinforced fine-tuning. *arXiv preprint arXiv:2401.08967*.

Yu Meng, Mengzhou Xia, and Danqi Chen. 2024. Simpo: Simple preference optimization with a reference-free reward. *Advances in Neural Information Processing Systems*, 37:124198–124235.

Meta@AI. 2024. Introducing llama 3.1: Our most capable models to date.

Yingqian Min, Zhipeng Chen, Jinhao Jiang, Jie Chen, Jia Deng, Yiwen Hu, Yiru Tang, Jiapeng Wang, Xiaoxue Cheng, Huatong Song, et al. 2024. Imitate, explore, and self-improve: A reproduction report on slow-thinking reasoning systems. *arXiv preprint arXiv:2412.09413*.

OpenAI. 2024. Openai o1-mini.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Rafael Rafailov, Joey Hejna, Ryan Park, and Chelsea Finn. 2024a. From r to q^*: Your language model is secretly a q-function. *arXiv e-prints*, pages arXiv–2404.

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

Martin Riedmiller, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degrave, Tom Wiele, Vlad Mnih, Nicolas Heess, and Jost Tobias Springenberg. 2018. Learning by playing solving sparse reward tasks from scratch. In *International conference on machine learning*, pages 4344–4353. PMLR.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. *URL https://arxiv.org/abs/2402.03300*.

Skywork. 2024a. Skywork/skywork-o1-open-prm-qwen-2.5-7b.

Skywork. 2024b. Sskywork-o1-open.

Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisacanin, Alexan Ayrapetyan, and Igor Gitman. 2024. Openmathinstruct-2: Accelerating ai for math with massive open-source instruction data. *arXiv preprint arXiv:2410.01560*.

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

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

Albert Wilcox, Ashwin Balakrishna, Jules Dedieu, Wyame Benslimane, Daniel Brown, and Ken Goldberg. 2022. Monte carlo augmented actor-critic for sparse reward deep reinforcement learning from suboptimal demonstrations. *Advances in neural information processing systems*, 35:2254–2267.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

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

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. 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*.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024a. Rest-mcts*: Llm self-training via process reward guided tree search. *arXiv preprint arXiv:2406.03816*.

Xuan Zhang, Chao Du, Tianyu Pang, Qian Liu, Wei Gao, and Min Lin. 2024b. Chain of preference optimization: Improving chain-of-thought reasoning in llms. *arXiv preprint arXiv:2406.09136*.

Heyang Zhao, Chenlu Ye, Quanquan Gu, and Tong Zhang. 2024. Sharp analysis for kl-regularized contextual bandits and rlhf. *arXiv preprint arXiv:2411.04625*.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2024a. Processbench: Identifying process errors in mathematical reasoning. *arXiv preprint arXiv:2412.06559*.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2024b. Processbench: Identifying process errors in mathematical reasoning. *arXiv preprint arXiv:2412.06559*.

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

A Data Example

We demonstrate the preference trajectories sampled from the tree using SPPD as shown below. As previously mentioned, the chosen step and the rejected step are selected based on their PRM score difference exceeding 0.5.

Training data format.

[**Question**]
Evaluate: $64^2 - 36^2$.
[**Common Prefix**]
Step 1: Recognize that the given expression is of the form $a^2 - b^2$, which can be

factorized using the difference of squares method.
Step 2: Apply the difference of squares formula, which is $a^2 - b^2 = (a+b)(a-b)$.
Step 3: Identify a and b in the given expression. Here, $a = 64$ and $b = 36$.
Step 4: Evaluate the sum and difference of a and b.
a + b = 64 + 36
a - b = 64 - 36
Step 5: Multiply the result from step 4 back together.
[**Chosen Step**]
$(64 + 36)(64 - 36)$
Step 6: Perform the addition and subtraction operations.
The sum of 64 and 36 is 100. The difference of 64 and 36 is 28.
[**Chosen PRM Score**]
0.934
[**Rejected Step**]
Step 6: $(64 + 36) * (64 - 36) = (100) * (28) = 1400$
[**Rejected PRM Score**]
0.258

B PRM score distribution

We use the Skywork-o1-Open-PRM-Qwen-2.5-7B (Skywork, 2024a)model to score 545,990 reasoning trajectories obtained from Qwen2.5-7B-Base (Yang et al., 2024) through Tree-Based Self-Sampling. The score for the t-th step of the i-th trajectory is denoted as $v_t^{(i)}$.

First, we calculate three metrics (ORM score, Mean PRM score, and Minium PRM score) on trajectories that produce correct answers and those that result in incorrect answers. If a metric exceeds 0.5, the PRM considers the sample to be a correct trajectory; otherwise, it is deemed an incorrect trajectory. We then compute the PRM accuracy rates under these three metrics, see Table 8. The experimental results demonstrate that Skywork-o1-Open-PRM-Qwen-2.5-7B exhibits strong discriminative ability for both correct and incorrect trajectories under sampled trajectories. Specifically, the ORM metric shows superior performance in identifying correct trajectories, achieving over 90% accuracy. In contrast, the minimum PRM score excels in distinguishing incorrect trajectories, reaching an accuracy of 92.5%. However, using the mean PRM score, the discriminative ability for correct trajectories is notably higher than for incorrect trajectories. This is because Skywork-o1-Open-PRM-Qwen-2.5-7B can effectively identify erroneous steps, resulting in high scores (close to 1) before these steps occur, which renders the mean PRM score ineffective for judging incorrect trajectories. Conversely, the minimum PRM score identifies the lower bound of trajectory scoring, making it the most suitable metric for evaluating incorrect trajectories.

Metric	#	ORM	Mean PRM	Minium PRM
Correct	281,983	0.908	0.920	0.705
Incorrect	264,007	0.870	0.696	0.925

Table 8: Skywork-o1-Open-PRM-Qwen-2.5-7B accuracy.

Meanwhile, we divide each trajectory into five equal segments, calculate the average score for each segment, and plot the score distribution in box plots categorized by correct and wrong trajectories, as shown in the Figure 2. The figure indicates that for correct trajectories, PRM assigns relatively high scores to all steps with smaller variance; for wrong trajectories, the segment scores given by PRM tend to decrease on average as they get closer to the answer, with the variance also decreasing, suggesting that PRM's confidence in the wrong trajectory leading to an incorrect answer increases.

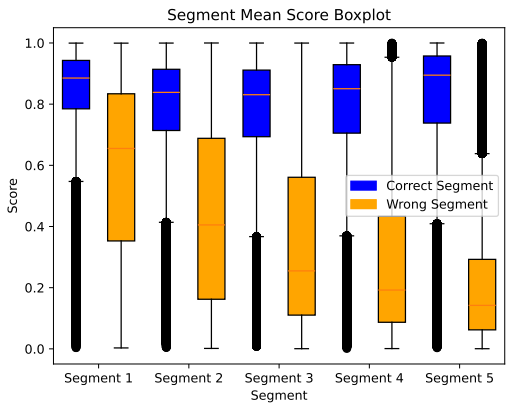

Figure 2: Skywork-o1-Open-PRM-Qwen-2.5-7B distribution.

C Definition

C.1 Preference Decoding Model

Definition C.1 (Preference Decoding Model π_θ^p Induced by π_θ)**.** *Assume that at state $s = s_t$, the available action space is $\mathcal{A}_t = \{a_t^w, a_t^l\}$. The preference decoding model π_θ^p is defined as the following parameterized distribution:*

$$\pi_\theta^p(a_t^w|s_t) = \sigma(r_{\theta,t}^w - r_{\theta,t}^l),$$

where

$$r_{\theta,t}^w = \beta \log \frac{\pi_\theta(a_t^w|s_t)}{\pi_{ref}(a_t^w|s_t)} - V^*(s_{t+1}^w) + V^*(s_t),$$

$$r_{\theta,t}^l = \beta \log \frac{\pi_\theta(a_t^l|s_t)}{\pi_{ref}(a_t^l|s_t)} - V^*(s_{t+1}^l) + V^*(s_t).$$

Remark. *The preference decoding model π_θ^p can be interpreted as performing preference-based sampling over a binary prefix tree. It is derived from the probability outputs of the base language model π_θ.*

D Evaluation

D.1 Evaluation Prompts

For a fair evaluation, the same prompt and format is applied to our trained models as well as other open-source models:

Prompt used for evaluation.

[SYSTEM]
Please reason step by step and put your answer in \\boxed{}.
[Question]
{question}.

E Proofs

E.1 Proof of Lemma (4.1)

Lemma E.1 (Optimal Step Reward Function)**.** *Under the step MDP definition3 and fix solution for the maximum casual entropy problem (Equation (2)), the optimal step reward function can be calculate as follow:*

$$r(s_t, a_t) = \underbrace{\beta \log \frac{\pi^*(a_t|s_t)}{\pi_{ref}(a_t|s_t)}}_{\textit{Implicit Reward}} - \underbrace{(V^*(s_{t+1}) - V^*(s_t))}_{\textit{Value Gain}}. \quad (9)$$

Proof. According to the Bellman optimality equation (Barron and Ishii, 1989) in step MDP, we have:

$$Q^*(s_t, a_t) = r(s_t, a_t) + V^*(f(s_t, a_t)). \quad (10)$$

Here, if $s_{t+1} = f(s_t, a_t)$ is a terminal state, then $V^*(f(s_t, a_t)) = 0$. Meanwhile, if we log-linearize the Equation (2), we have:

$$Q^*(s_t, a_t) = \beta \log \frac{\pi^*(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)} + V^*(s_t). \quad (11)$$

Therefore, combine the Equation (10) & (11), we have:

$$r(s_t, a_t) = \underbrace{\beta \log \frac{\pi^*(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)}}_{\text{Implicit Reward}} - \underbrace{(V^*(s_{t+1}) - V^*(s_t))}_{\text{Value Gain}}.$$

□

E.2 Proof of Theorem E.2

Theorem E.2 (Step DPO Loss Using Dynamic Value Margin.)**.** *If we aim to minimize the Kullback-Leibler(KL) divergence between the step-level preference distribution p_{data} in $\mathcal{D}_{step}$ and the model's current preference distribution p_θ under the sampling of π_{ref}, we can obtain the following loss function:*

$$\begin{aligned}\mathcal{L}_{\textit{step-dpo}} = -\mathbb{E}_{a_t^w, a_t^l \sim \pi_{ref}(\cdot|s_t)}[\\ \log \sigma(\beta h_\theta(a_t^w, a_t^l) \\ - (V^*(s_{t+1}^w) - V^*(s_{t+1}^l)))],\end{aligned}$$

where $h_\theta(a_t^w, a_t^l) = \log \frac{\pi_\theta(a_t^w|s_t)}{\pi_{ref}(a_t^w|s_t)} - \log \frac{\pi_\theta(a_t^l|s_t)}{\pi_{ref}(a_t^l|s_t)}$. Regarding p_{data}, it represents the preference distribution in the dataset $\mathcal{D}_{step}$, which takes the form of a conditional point-mass distribution. For instance, given a data instance (s_t, a_t^w, a_t^l), we have: $p(a_t^w \succ a_t^l|s_t) = 1$ and $p(a_t^l \succ a_t^w|s_t) = 0$.

Proof. According to the Equation (3) and Equation (4), we have:

$$\begin{aligned}&p_\theta(a_t^w \succ a_t^l|s_t) \\ &= \sigma(\beta h_\theta(a_t^w, a_t^l) - (V^*(s_{t+1}^w) - V^*(s_{t+1}^l)))\end{aligned} \quad (12)$$

So the KL divergence between p_θ and p_{data} under

the sampling of π_{ref} is:

$$\mathbb{E}_{a_t^w, a_t^l \sim \pi_{ref}(\cdot|s_t)}[D_{KL}(p_{data}||p_\theta)]$$
$$= \mathbb{E}_{a_t^w, a_t^l \sim \pi_{ref}(\cdot|s_t)}[$$
$$p_{data}(a_t^w \succ a_t^l|s_t) \log \frac{p_{data}(a_t^w \succ a_t^l|s_t)}{p_\theta(a_t^w \succ a_t^l|s_t)}$$
$$+ p_{data}(a_t^l \succ a_t^w|s_t) \log \frac{p_{data}(a_t^l \succ a_t^w|s_t)}{p_\theta(a_t^l \succ a_t^w|s_t)}]$$
$$= -\mathbb{E}_{a_t^w, a_t^l \sim \pi_{\text{ref}}(\cdot|s_t)}[\log p_\theta(a_t^w \succ a_t^l|s_t)],$$

which is the same as Equation (5).

□

E.3 Proof of Theorem 5.2

Theorem E.3 (Equivalence Between Offline Step DPO and Online Policy Gradient)**.** *If we define the reward in Equation (7) as* $r(\tau) = \prod_{i=0}^{T-1} \frac{\pi_{ref}(a_t|s_t)}{\pi_\theta^p(a_t|s_t)}$, *and define the* ***Offline every-step preference loss*** *as:*

$$\mathcal{L}_{\textit{every-step}} = \mathbb{E}_{\tau \sim \pi_{ref}^p}\left[-\sum_{t=0}^{T-1} \log \pi_\theta^p(a_t^w|s_t)\right],$$

then the following equivalence holds:

$$\nabla_\theta J(\theta) = -\nabla_\theta \mathcal{L}_{\textit{every-step}}.$$

Proof.

$$\nabla_\theta \mathcal{L}_{every-step}$$
$$= \mathbb{E}_{\tau \sim \pi_{ref}^p}[-\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta^p(a_t^w|s_t))]$$
$$= \mathbb{E}_{\tau \sim \pi_\theta^p}[-\frac{\pi_{ref}^p(\tau)}{\pi_\theta^p(\tau)} \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta^p(a_t^w|s_t))]$$
$$= \mathbb{E}_{\tau \sim \pi_\theta^p}[$$
$$-\prod_{i=0}^{T-1} \frac{\pi_{ref}^p(a_t|s_t)}{\pi_\theta^p(a_t|s_t)} \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta^p(a_t^w|s_t))]$$
$$= \mathbb{E}_{\tau \sim \pi_\theta^p}[-r(\tau) \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta^p(a_t^w|s_t))]$$
$$= -\nabla_\theta J(\theta).$$

□

F Supplementary experiments

F.1 Computational cost and efficiency of the SPPD

Compared to online reinforcement learning algorithms such as PPO and GRPO, our method requires self-generated paths and PRM scoring. Although these steps are computationally intensive, they only need to be performed once as training and data generation are decoupled. This leads to higher training efficiency for the model. In contrast, PPO and GRPO require online trajectory sampling during training, resulting in lower training efficiency. For example, under the setup of 8 H800 GPUs and 10k data samples, with 8 sampling iterations per question, the time consumption of each method is shown in the Table 9.

Methods	Time
SPPD-Generation & PRM Scoring	6.6h
SPPD-Train	1.6h
GRPO	7.3h
PPO	10.5h

Table 9: Computational cost and efficiency of the SPPD and other methods. SPPD-Generation indicates the model generating trajectories by itself. SPPD-Scoring refers to PRM scoring. SPPD-Train represents the training of PRM-SFT, PRM-DPO, and step-DPO with the dynamic margin.

F.2 Impact of γ.

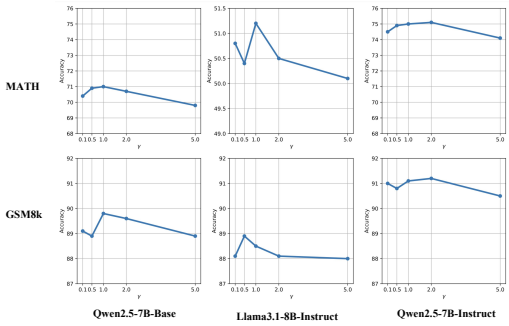

Figure 3: Impact of γ in dynamic value margin.

To evaluate the impact of the hyperparameter γ on the SPPD method (Equation 6), we selected three base models: Qwen2.5-7B-Base, Llama3.1-8B-Instruct, and Qwen2.5-7B-Instruct. We varied γ within the set $\{0.1, 0.5, 1.0, 2.0, 5.0\}$ and assessed model performance on the MATH and GSM8K datasets. The results are illustrated in Figure 3. Our findings indicate that an optimal γ selection benefits SPPD training. Specifically, both excessively high and low values of γ impair the training of dynamic value margins, affecting generalization. However, overall performance remains stable, particularly on the GSM8K dataset.

This highlights the importance of a balanced γ for optimizing SPPD effectiveness across different models. While extreme values of γ may lead to subpar performance compared to fixed-margin step-DPO, Figure 3 demonstrates that $\gamma \approx 1$ consistently outperforms fixed-margin methods across all models. Consequently, setting $\gamma = 1$ represents a robust choice in practice.