Coarse-to-Fine Process Reward Modeling for Mathematical Reasoning

Anonymous ACL submission

Abstract

The Process Reward Model (PRM) plays a crucial role in mathematical reasoning tasks, requiring high-quality supervised process data. However, we observe that reasoning steps generated by Large Language Models (LLMs) often fail to exhibit strictly incremental information, leading to redundancy that can hinder effective reasoning. To address this issue, we propose CFPRM, a simple yet effective coarse-to-fine strategy. Instead of focusing on the detection of redundant steps, our approach first establishes a coarse-grained window to merge adjacent reasoning steps into unified, holistic steps. The window size is then progressively reduced to extract fine-grained reasoning steps, enabling data collection at multiple granularities for training. By leveraging this hierarchical refinement process, CFPRM mitigates redundancy while preserving essential fine-grained knowledge. Extensive experiments on two reasoning datasets across three loss criteria validate the CFPRM's effectiveness and versatility. Our code is available https://anonymous.4open.science/r/CFPRM-0FF2.

1 Introduction

Large language models (LLMs) have demonstrated promising capabilities across a wide range of domains (Kaddour et al., 2023; Achiam et al., 2023; Dubey et al., 2024; Yang et al., 2024), including complex mathematical reasoning tasks (Lightman et al., 2023; Huang et al., 2023). An accurate process reward model (PRM) is vital for reasoning tasks, as it provides intermediate supervision signals for each individual step (Uesato et al., 2022).

Training PRM requires the collection of stepwise annotated corpora (Lightman et al., 2023; Uesato et al., 2022). For instance, Lightman et al. (Lightman et al., 2023) propose manually annotating the intermediate MATH data, where each step is assigned a ternary label. However, such human-intensive labeling is costly, hindering broader practical applications. An alternative approach involves constructing automatic labeling methods, either by defining the probability of each intermediate step as the potential to deduce the final correct answer (Wang et al., 2024), or by using a tree-based structure to iteratively refine the logits of each intermediate trajectory (Zhang et al., 2024). Despite the preliminary success of these methods, they primarily focus on accurately assigning labels to each step, while overlooking the potential redundancy of steps that may offer no incremental information gain (Li and Li, 2024). Given that mathematical reasoning is a progressive process, where each current step depends on previous ones (Li and Li, 2024), later steps should ideally provide more informative contributions toward approximating the final answer. To illustrate this, we present a data collection example from the MATH dataset (Hendrycks et al., 2021) via Shepherd (Wang et al., 2024) in Figure 1. However, we observe that steps s_1, s_2, and s_3 are logically correct, but the repetitive reasoning procedures fail to yield any new information, which contradicts the learning objective.

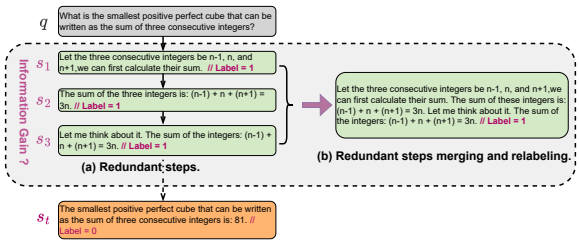


Figure 1: Redundant steps merging.

To tackle the limitation, we propose CFPRM, a coarse-to-fine strategy for process data collection and training, which is simple yet effective. We do not explicitly detect redundant steps; as the name suggests, we collect process training data in a coarse-to-fine manner and proceed with the learning process in the same way. Specifically, we define

a step window size C to represent the initial step granularity, i.e., every C steps are collected and merged into a holistic step, with the corresponding label of the merged step determined by the label of the last individual step. Subsequently, C is gradually reduced until it reaches 1, and training data are collected in the same way following the above procedure. This strategy gathers training data of diverse granularity, directly integrating consecutive steps to form coarse steps without designing methods to detect redundant steps. Meanwhile, the initial individual steps are preserved to offer necessary fine-grained signals. We validate the proposed strategy on two cutting-edge LLMs across three learning criteria, yielding consistently enhanced performance, demonstrating the effectiveness and versatility of CFPRM.

2 Methodology

2.1 Preliminaries

We denote an LLM policy as π, and r_θ as the PRM fine-tuned upon π, parameterized by θ. For reasoning tasks, π generates responses step by step given an input query x in an autoregressive manner: $s_t \sim \pi_\theta(\cdot \mid x, s_{1:t-1}), t \leq T$, T is the total reasoning steps. The PRM policy r_θ then outputs a reward given the partial solutions and the input query as: $r_{s_t} = r_\theta(s_{1:t}, x)$. We regard y_{s_t} as the label for step t. In addition, the existing PRM training objectives can be summarized into three types, including mean square error (MSE) (Zhang et al., 2024), binary cross-entropy (BCE) (Wang et al., 2024), and Q-value rankings (Q-ranking) (Li and Li, 2024), as shown in Table 1. It is worth noting that CFPRM can be applied to arbitrary loss criteria and achieves consistent improvements, as demonstrated in Section 3.

Table 1: The typical loss objectives for PRM training.

Losses	Formulation
BCE	$\sum_{t=1}^{T} y_{s_t} \log r_{s_t} + (1 - y_{s_t}) \log (1 - r_{s_t})$
MSE	$\sum_{t=1}^{T} (r_\theta(s_{1:t}, x) - y_{s_t})^2$
Q-Ranking	$-\frac{1}{\lvert T \rvert} \sum_{t=0}^{\lvert T \rvert} \log \frac{\exp(r_{c_t})}{\sum_{q=0}^{t} \exp r_{c_q} + \sum_{w \in W} \exp(Q_w + \zeta)}$

W indicates negative steps, Q is the ranking value, and ζ is the margin hyperparameter.

2.2 Coarse-to-fine Process Data Collection

CFPRM can be applied to reasoning data collected by any kind of structure, such as Chain of Thought (Wang et al., 2024) or Monte Carlo Tree Search (Zhang et al., 2024) methods. We extend the solution process in Figure 1 as an example and present the coarse-to-fine process data collection in Figure 2. The process is intuitive, involving first merging the consecutive steps and then relabeling each merged step.

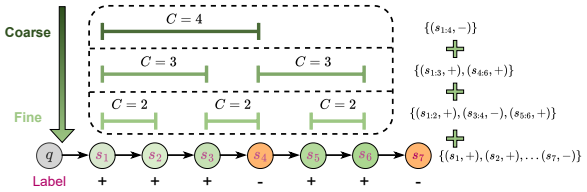


Figure 2: Coarse-to-fine process data collection.

Steps merging. Consider the problem in Figure 1 containing a trajectory of 7 reasoning steps, where steps s_1, s_2, s_3, s_5, and s_6 are correct steps, and s_4 and s_7 are wrong steps. It is worth noting that step s_4 fails to make incremental reasoning from s_3, but the LLM policy manages to adjust the wrong step (Setlur et al., 2024). The final step fails to reach the correct answer.

As a coarse-to-fine method, CFPRM gradually consolidates multiple reasoning steps based on a predefined sliding window size, denoted as C. Here, C represents the size of the merging window, with C_{max} as the maximum window size, while the minimum size is usually set to 1. Ideally, C can initially be set to the total number of steps in the entire reasoning trajectory, and the merged trajectory is labeled with the label of the last step, acting as ORM. In practice, the continuous partial steps are merged instead of the entire trajectory to avoid excessive concentration of knowledge. In Figure 2, C is initially set to 4, yielding the merged partial trajectory s_1, s_2, s_3, s_4, treated as a single step $s_{1:4}$. Then, the window slides to the next starting point (s_5) to collect a new partial trajectory. Subsequently, C is sequentially decreased to 3 and 2, yielding additional partial trajectories of different sizes. After finishing collecting the consecutive steps using the sliding window, we combine the merged coarse training samples with the original fine-grained data.

Merged steps labeling. In addition, each collected partial trajectory may contain positive or negative steps, which makes it difficult to determine its label. Drawing inspiration from ShepHerd (Wang et al., 2024), which considers a trajectory's label to

Table 2: Main results measured by BoN accuracy. C is set to 2.

Version	Models	GSM-Plus					MATH500				
		@8	@16	@32	@64	Avg.	@8	@16	@32	@64	Avg.
Instruct	ORM	67.0	66.0	68.8	66.8	67.2	71.4	69.2	70.2	68.1	69.7
	SHerd	67.8	67.2	68.4	67.0	67.6	74.4	75.5	75.8	76.0	75.4
	SHerd$_{+\text{CFPRM}}$	68.2	67.4	69.0	70.0	**68.7**$_{\uparrow 1.1}$	75.6	76.2	76.6	77.0	**76.4**$_{\uparrow 1.0}$
	RMCTS*	69.4	69.2	66.8	68.0	68.4	71.0	71.2	71.9	72.6	71.7
	RMCTS*$_{+\text{CFPRM}}$	68.2	70.4	70.2	70.0	**69.7**$_{\uparrow 1.3}$	75.2	75.6	74.9	74.6	**75.1**$_{\uparrow 3.4}$
	PQM	67.6	68.8	66.4	67.0	67.5	74.6	75.4	75.8	75.3	75.3
	PQM$_{+\text{CFPRM}}$	68.0	68.0	69.4	71.0	**69.1**$_{\uparrow 1.6}$	75.4	76.2	76.7	77.1	**76.4**$_{\uparrow 1.1}$
MATH	ORM	63.0	61.8	62.8	62.8	62.6	78.0	77.7	77.6	77.6	77.7
	SHerd	69.2	69.2	69.2	70.0	69.4	82.1	81.8	81.9	82.0	82.0
	SHerd$_{+\text{CFPRM}}$	69.2	70.8	71.2	73.0	**71.2**$_{\uparrow 1.8}$	82.4	82.3	82.8	82.8	**82.6**$_{\uparrow 0.6}$
	RMCTS*	68.6	69.2	68.6	70.0	69.1	81.6	81.7	82.0	81.8	81.8
	RMCTS*$_{+\text{CFPRM}}$	69.2	70.0	70.4	72.0	**70.4**$_{\uparrow 1.3}$	82.2	82.7	82.8	82.8	**82.6**$_{\uparrow 0.8}$
	PQM	70.2	69.8	72.2	73.0	71.3	84.0	84.1	84.2	84.2	84.1
	PQM$_{+\text{CFPRM}}$	69.4	71.4	72.2	74.0	**71.8**$_{\uparrow 0.5}$	84.5	84.9	85.2	85.2	**85.0**$_{\uparrow 0.9}$

depend on its potential to deduce the answer, we label each merged step by the label of the last step in the window. For example, the label of $s_{1:4}$ is the same as the label of step s_4, indicating a negative sample, and we add $(s_{1:4}, +)$ into the set $\mathcal{D}_{C=4}$. Similarly, the trajectory $s_{1:2}$ is treated as a positive sample, sharing the same label as step s_2. Following this, each merged step is relabeled and used as supervisory training samples, with the trajectory added to the set $\mathcal{D}_{C=2}$. In this way, we obtain a renewed training corpus containing samples of diverse granularity.

Training and inference. We proceed with training after obtaining the training samples of different granularity. We choose to recombine the training trajectories according to their granularity. Specifically, we traverse the corpus sequentially from C_{max} to 1. Through this process, the process supervision knowledge is gradually distilled in a coarse-to-fine manner. The training process is identical for different loss criteria. During inference, we use the trained PRM to predict scores for each single step. We summarize the overall process in Appendix 1.

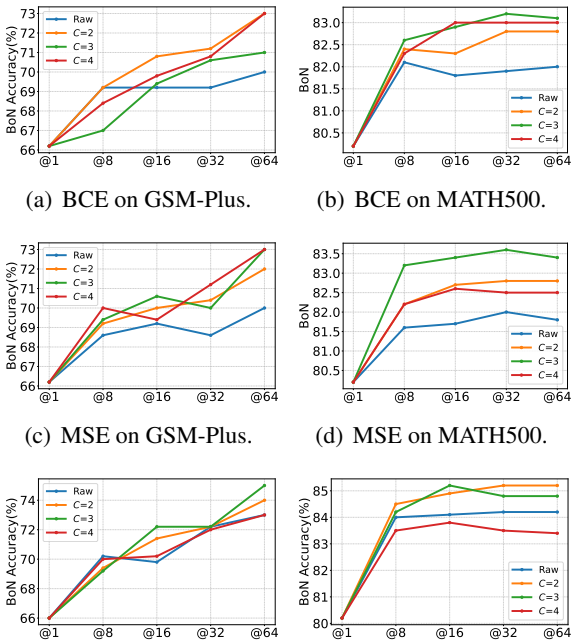
3 Experiment

3.1 Experimental Setup

Datasets and models. We adopt two widely used mathematical reasoning test sets, GSM-Plus (Li et al., 2024) and MATH500 (Hendrycks et al., 2021), for evaluation. GSM-Plus is built upon GSM8K (Cobbe et al., 2021) with various mathematical perturbations. The original GSM8K is a benchmark of grade-school level problems. The MATH dataset consists of high school math competition problems, which are more challenging. For each candidate, the PRM evaluates the score of each step. Instead of collecting raw process data from scratch, we utilize the off-the-shelf PRM800K (Lightman et al., 2023) dataset to train the PRM. We employ two cutting-edge LLMs, Qwen2.5-7B-Instruct and Qwen2.5-7B-MATH (Yang et al., 2024), as the backbone models. Following previous studies (Wang et al., 2024; Li and Li, 2024), we evaluate the PRM using the best-of-n (BoN) sampling strategy, with n set to 8, 16, 32, and 64, respectively. We also use the same backbone model to generate 64 candidates for each given question, ensuring consistency in the backbone model and the BoN sampling policy.

Baselines and details. As a method to refine the data collection mechanism, CFPRM can be seamlessly applied to any existing methods. Specifically, we select the most recent methods covering the

three loss criteria in Table 1 for comparison, including ShepHerd (Wang et al., 2024) using the BCE objective, ReSTMCTS* (Zhang et al., 2024) using the MSE objective, and PQM (Li and Li, 2024) built upon the Q-ranking objective. We also include ORM for comparison. The best performance is marked in bold. We set the max length to 2048, the learning rate to 2e-6, and the batch size to 32. All experiments are conducted on an H800-80G GPU, with each experiment repeated five times to report the mean results. Accuracy is reported as the evaluation metric.



(a) BCE on GSM-Plus. (b) BCE on MATH500.

(c) MSE on GSM-Plus. (d) MSE on MATH500.

(e) Q-Ranking on GSM-Plus. (f) Q-Ranking on MATH500.

Figure 3: The BoN accuracy change under different values of C.

3.2 Main Results

For CFPRM, we set C to 2 to merge the adjacent two steps, showing the performance in Table 2. SHerd and RMCTS* are abbreviations for ShepHerd (Wang et al., 2024) and ReSTMCTS* (Zhang et al., 2024), respectively. The last column of each dataset indicates the average performance across four sampling conditions.

Our experimental results indicate that CFPRM consistently brings performance improvements across all configurations, irrespective of the backbone model or loss objectives, underscoring its generalizability. For example, when employing MSE as the loss function, CFPRM improves upon the baseline, ReSTMCTS* (Zhang et al., 2024), by 1.3% and 3.4% on GSM-Plus and MATH500, respectively. Among the three learning objectives, the Q-ranking criterion exhibits superior performance, likely due to its foundation in the Markov Decision Process (MDP), which emphasizes evaluating transitions between adjacent steps. However, the presence of redundant steps can impede this learning process. By integrating CFPRM with the Q-ranking-based method (Li and Li, 2024), we observe further performance enhancements. Overall, these findings robustly affirm the efficacy and adaptability of CFPRM. Considering the simplicity of CFPRM, CFPRM can be applied as a plug-and-play strategy to various scenarios.

3.3 Further Studies

We further explore the impact of varying C, ranging from 2 to 4. It is worth noting that only the newly synthesized data is added. For simplicity, we only take the Qwen2.5-7B-MATH as the backbone model, leveraging BCE, MSE, and Q-ranking as the loss objectives, studying the performance changes on GSM-Plus and MATH500.

We can observe from Figure 3 that CFPRM generally brings performance gains across different learning objectives. However, the impact of different C varies, indicating that the optimal merge window size is not fixed for different learning criteria. In addition, we also find that the Q-ranking-based method is more sensitive to C. When C is set to 4, the performance of the Q-ranking-based method on GSM-Plus does not exhibit a difference compared to the raw baseline (Figure 3(e)). Moreover, the performance on MATH500 lags behind the raw baseline (Figure 3(f)). We ascribe this to the fact that the Q-ranking-based objective is sensitive to the interdependence between steps, and a large merging window hinders the learning of necessary fine-grained dependencies. Generally, setting C to 2 or 3 can better boost the overall ranking performance.

4 Conclusion

In this paper, we briefly review the issue of redundant steps in process data collection for PRM training, which may hinder downstream performance. To tackle this problem, we propose CFPRM, a coarse-to-fine strategy that employs a sliding window to collect process data at diverse granularities. We validate CFPRM across multiple experimental settings, confirming its effectiveness and versatility.

5 Limitations

A more reasonable method should involve detecting and removing redundant steps, which we have not discussed. Another limitation is that the optimal C should be designed adaptively with respect to different loss criteria. Future work should focus on designing methods to accurately detect redundant steps and developing adaptive methods to choose a feasible value of C.

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*.

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*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

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*.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2023. Large language models cannot self-correct reasoning yet. *arXiv preprint arXiv:2310.01798*.

Jean Kaddour, Joshua Harris, Maximilian Mozes, Herbie Bradley, Roberta Raileanu, and Robert McHardy. 2023. Challenges and applications of large language models. *arXiv preprint arXiv:2307.10169*.

Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. 2024. Gsm-plus: A comprehensive benchmark for evaluating the robustness of llms as mathematical problem solvers. *arXiv preprint arXiv:2402.19255*.

Wendi Li and Yixuan Li. 2024. Process reward model with q-value rankings. *arXiv preprint arXiv:2410.11287*.

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

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. 2024. Rewarding progress: Scaling automated process verifiers for llm reasoning. *arXiv preprint arXiv:2410.08146*.

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

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439.

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*.

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

A Supplement Material

Algorithm 1: Coarse-to-Fine Step Merging and Relabeling

Require: Trajectory of reasoning steps $S = \{s_1, s_2, \ldots, s_N\}$, where each step s_i has a label $l_i \in \{+, -\}$.

Ensure: Mixed training corpus $\mathcal{D}$ with samples of diverse granularity.

Initialize C_{max} as the maximum window size.
Initialize $\mathcal{D} \leftarrow \emptyset$ {Empty set to store merged samples.}
for $C = C_{\text{max}}$ **to** 1 **do**
  for each window of size C in S **do**
    Merge steps in the window into a single step $s_{i:j}$, where $j = i + C - 1$.
    Assign the label of $s_{i:j}$ as the label of the last step s_j.
    Add $(s_{i:j}, l_j)$ to $\mathcal{D}_C$.
  end for
  Combine $\mathcal{D}_C$ with $\mathcal{D}$.
end for
return $\mathcal{D}$ {Mixed training corpus with diverse granularity.}