RREDCoT: Segment-Level Reward Redistribution for Reasoning Models

Anonymous authors
Paper under double-blind review

Abstract

Recent advancements in reasoning language models have been driven by Reinforcement Learning (RL) fine-tuning. Most often, these rely on the Group Relative Policy Optimization (GRPO) algorithm or modifications thereof to steer the models to produce Chain-of-Thought (CoT) traces. The final answer can only be verified, and the reward assigned, after the CoT trace is complete, making it a delayed reward problem. GRPO and its modifications correspond to Monte Carlo methods in standard RL, which are known to suffer from high variance. A possible solution to this problem is the redistribution of rewards through credit assignment, where segments of the CoT trace that are important for arriving at the desirable solution are emphasized by assigning a higher reward. While Monte Carlo sampling can be used to provide an unbiased estimate of intermediate state values, its computational overhead makes it unsuitable for train-time credit assignment in long contexts at high granularity. We introduce RREDCoT (Reward REDistribution for Chain of Thoughts), which utilizes the model itself to approximate the optimal reward redistribution without additional generation. We investigate the advantages of our method compared to MC sampling and several attribution methods. We further analyze several aspects relevant to the construction of the redistribution such as segmentation of CoT traces and state value estimation.

1 Introduction

As established modes of language model pretraining hit the boundaries of data availability, the focus shifts further towards maximizing the utility extracted from the pretrained models in subsequent "post-training" stages (Kimi Team et al., 2025). Techniques such as Reinforcement Learning from Human Feedback (RLHF) (Ouyang et al., 2022) have emerged, relying on a reward signal from possibly vague human preference data. These early attempts to improve the quality of language models were based on the Proximal Policy Optimization (PPO) algorithm (Schulman et al., 2017).

Today, Group Relative Policy Optimization (GRPO) (Shao et al., 2024) has become the most popular choice for RL-based fine-tuning, because it circumvents the necessity of simultaneously learning an advantage function. This is combined with RL with Verifiable Rewards (RLVR) paradigm, in which model generates a Chain-of-Thought (CoT) before producing the final result, with training often consisting of online tuning followed by distillation on the generated reasoning traces (DeepSeek-AI et al., 2025). However, CoT generation poses several challenges: models tend to produce excessively long traces, and answer extraction and reward estimation can yield disparate performance assessments and noisy training signals (Shao et al. (2025), Chandak et al. (2025)).

One prominent issue that RL fine-tuning of reasoning language models still faces is the lack of fine-grained reward signals. In RLVR, the reward for the entire episode is assigned only at the very end of a generated Chain-of-Thought (CoT) and distributed uniformly over the full trajectory, which provides no direct supervision for individual reasoning steps. In on-policy reasoning fine-tuning, this problem has been approached from several directions. Some works propose step-by-step analysis of CoT traces using judge models as a means

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

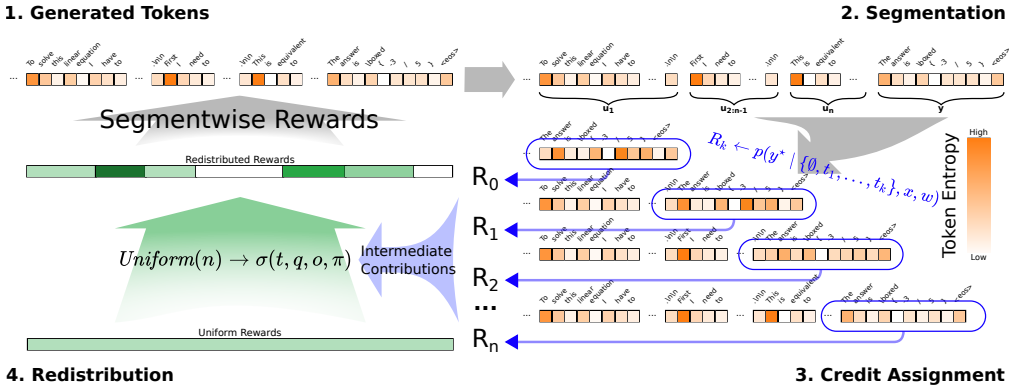


Figure 1: Overview of the RREDCoT algorithm for reward redistribution. To compute a reward redistribution σ: (1) generate a CoT trace, (2) segment the trace, (3) compute each segment's immediate reward, and (4) use these rewards to derive the final redistribution.

of credit assignment (Xie et al., 2025; Ou et al., 2025; Jayalath et al., 2025), while others attempt to extract intermediate utility directly from model statistics (Li et al., 2026). Monte Carlo sampling (MCS) based state value estimation methods have been shown to provide effective intermediate value estimates (Kazemnejad et al., 2025; Guo et al., 2025) at the cost of considerable additional token generation.

With RREDCoT, we adapt the core principles of RUDDER (Arjona-Medina et al., 2019) to the specific structure of the CoT generation MDP and derive a tractable approximation of the optimal reward redistribution that requires neither additional models nor extra generation steps. Unlike the original RUDDER formulation, which employs an LSTM (Hochreiter and Schmidhuber, 1997) for return decomposition, we utilize the generating language model itself. This exploits the properties of autoregressive sequence generation together with a novel fast value function estimator inspired by Bayesian methods in natural language generation (Malinin and Gales, 2020; Aichberger et al., 2024).

Our contributions are as follows:

- We introduce RREDCoT - a tractable credit assignment and reward redistribution algorithm for CoT traces that does not require additional models.
- We devise an entropy-based segmentation strategy for the CoT traces that is specifically tailored for RREDCoT.
- We analyze the relation between our estimator, MC sampling and several attribution methods.

2 CoT Generation as a Reinforcement Learning Problem

We consider a Markov Decision Process $\mathcal{P} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, p, \gamma)$ where $\mathcal{S}$ is a set of states, $\mathcal{A}$ the set of actions, and $\mathcal{R}$ the reward space. A state at step t is given by $s_t = (\boldsymbol{x}, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_{t-1})$ where $\boldsymbol{x}$ is the original query and each $\boldsymbol{u}_t$ is a generated CoT *segment*. The language model is then the policy with a discrete categorical action space, i.e., the space of possible next tokens a, also called vocabulary.

Segments are contiguous and non-overlapping sequences of generated tokens that together span the entire sequence. In the corner cases, the whole generated text would be split into segments consisting of individual tokens or kept together as a large single segment. This depends on *segmentation strategy* discussed further in Sec. 3. Importantly, as the state is defined by $\boldsymbol{s}_t = (\boldsymbol{x}, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_{t-1})$, the dynamics function is deterministic when conditioned on the next segment, i.e., $p(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, \boldsymbol{u}_t) = 1$. Thus, there is only stochasticity through the policy (the LLM). In the subsequent equations, whenever the transitions (e.g. $p(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t)$) are listed without specifying the parameters, $\boldsymbol{w}$ of the generating model are implied.

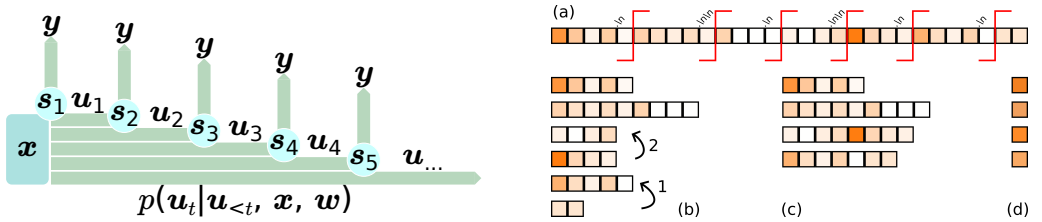


Figure 2: **(Right)** The MDP structure of autoregressive CoT generation. At each point, either another segment $\boldsymbol{u}_{t+1}$ or the answer $\boldsymbol{y}$ can be generated. The state $\boldsymbol{s}_t$ at each point consists of the entire sequence generated so far and the original input: $\{\boldsymbol{x}, \boldsymbol{u}_1 \dots \boldsymbol{u}_{t-1}\}$. **(Left)** Hybrid segmentation approach. *(a)* keyword segmentation; *(b)* iterative merging of the lowest total entropy pairs; *(c)* merging stops when criteria is met (number of segments); *(d)* the segment entropies are homogenized.

Reward Modeling for CoT Outputs. The reward model of text generation used for evaluation is as per Eq. (1). We call $\zeta : \boldsymbol{y} \mapsto [0, 1]$ a utility function. It maps from the answer space to a real value, which is the inverse of the cost of generating the final output sequence $\boldsymbol{y}$. In the simplest case, ζ is defined as the correctness of the answer. Often it is a combination of several different cost functions, such as adherence to output format or the overall length of $(\boldsymbol{u}, \boldsymbol{y})$, with improperly formatted or overly long sequences being associated with higher cost. The return is the expected inverse risk, conditioned on the CoT trace $\boldsymbol{u}$.

$$R_u = \mathrm{E}_{\boldsymbol{y} \sim p(\boldsymbol{y}|\boldsymbol{u},\boldsymbol{x},\boldsymbol{w})}[\zeta(\boldsymbol{y})] \tag{1}$$

The CoT trace $\boldsymbol{u}$ does not affect the reward assigned to the final action $\boldsymbol{y}$ directly. Instead, it affects the reward by changing the conditional probability distribution of $\boldsymbol{y}$. The VR estimator uses Monte Carlo integration to estimate $R_{\boldsymbol{y}}$ with most prior work only sampling a single output together with the original trace (Appx.Eq. (13)). Several recent works propose using Probability Reward (PR) instead (Zhou et al., 2025; Yu et al., 2025a). PR reward estimators use importance sampling (Appx.Eq. (14)) to reduce variance. This way, the ability of a model to serve as a dynamics function (i.e., world model) is leveraged. It is thus important that the model is reasonably well calibrated to assess the degree of causality between the CoT trace and the answer. PR estimator provides a denser signal but may suffer from increased noise as was pointed out in Yu et al. (2025b).

CoT Generation MDP and its Bellman Equation. A peculiar feature of the particular MDP at hand is that the action space $\mathcal{A}$ is partitioned into $\{\mathcal{Y}, \mathcal{U}\}$ subspaces. The $\mathcal{Y}$ subspace contains all the actions that result in the beginning of the final output, while the $\mathcal{U}$ subspace contains all of the actions that do not trigger the output and instead continue the generation of the CoT trace. In the commonly used models, the first subset of actions is initiated when end of thinking token (EOT) is produced. Upon selecting this token, the model can then produce the answer for which the reward can be evaluated as per Eq. (1). Therefore, we can rewrite the Bellman equation for such MDP in the following fashion:

$$\begin{aligned} v_{\boldsymbol{w}}(\boldsymbol{s}_t) &= \sum_{a \in \mathcal{Y}} p(a \mid \boldsymbol{s}_t)\ \zeta(\boldsymbol{s}_{t+1}) + \sum_{a \in \mathcal{U}} p(a \mid \boldsymbol{s}_t)\ \gamma\ v_{\boldsymbol{w}}(\boldsymbol{s}_{t+1}) \\ &= \mathrm{E}_{\boldsymbol{y} \sim p(\boldsymbol{y}|\boldsymbol{s}_t)}[\zeta(\boldsymbol{y})] + \mathrm{E}_{\boldsymbol{u}_t \sim p(\boldsymbol{u}_t|\boldsymbol{s}_t)}[v_{\boldsymbol{w}}(\boldsymbol{s}_{t+1})] \end{aligned} \tag{2}$$

In Eq. (2), we use the knowledge of the CoT MDP structure to split the value function on the action space $\mathcal{A}$ into two parts: one that corresponds to the model selecting an answer and the other that corresponds to the model selecting to continue CoT generation with fragment $\boldsymbol{u}_t$. With CoT generation, the model only gets rewarded for generating useful output in the end. The immediate reward for generating another thought is zero. CoT traces from commonly used models can reach thousands of tokens in length (Muennighoff et al., 2025; DeepSeek-AI et al., 2025), resulting in highly delayed rewards. We can further derive the corresponding action-value function $q^{\boldsymbol{w}}(\boldsymbol{s}_t, a)$:

$$q^{\boldsymbol{w}}(\boldsymbol{s}_t, a) = \begin{cases} \mathrm{E}_{a \sim p(a|\boldsymbol{s}_t,\boldsymbol{x},\boldsymbol{w})}[\zeta(\boldsymbol{s}_t)] & \text{if } a \in \mathcal{Y} \\ v_{\boldsymbol{w}}(\boldsymbol{s}_{t+1}) & \text{if } a \notin \mathcal{Y} \end{cases} \tag{3}$$

Algorithm 1 Approximately Optimal Reward Redistribution for CoT traces.

Supplies: Provides the optimal redistribution of Conditional Code Length
Requires: CoT trace of T segments $(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_T)$, reference answer $\boldsymbol{y}^\star$, reference solution $\boldsymbol{u}^\star$, original input sequence $\boldsymbol{x}$, predictive model with parameters $\boldsymbol{w}$ used for autoregression.

for $t = 0$ to T **do**
  $\hat{R}_t \leftarrow p(\boldsymbol{y}^\star \mid (\emptyset, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_t), \boldsymbol{x}, \boldsymbol{w})$ // the answer probs are evaluated in prefill
  $\hat{v}_t \leftarrow p(\boldsymbol{u}^\star \mid (\emptyset, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_t), \boldsymbol{x}, \boldsymbol{w})$ // the answer probs are evaluated in prefill
end for
for $t = T$ to 1 **do**
  $\sigma_t^{\text{unnorm}} \leftarrow R_{t+1} - R_t + \hat{v}_{t+1} - \hat{v}_t$ // computing the sigmas
end for
$\sigma \leftarrow \texttt{normalize}(\sigma^{\text{unnorm}})$ // normalizing sigmas

Alternatively, the value term of the CoT MDP can be interpreted as an expectation under the predictive distribution of the reasoning language model:

$$\begin{aligned} v_{\boldsymbol{w}}(\boldsymbol{s}_t) &= \mathrm{E}_{\boldsymbol{y},\boldsymbol{u} \sim p(\boldsymbol{y},\boldsymbol{u} \mid \boldsymbol{x},\boldsymbol{w})} \Big[\zeta(\boldsymbol{y}) \Big] \\ &= \sum_{\boldsymbol{y} \in \mathcal{Y}, \boldsymbol{u} \in \mathcal{U}} p(\boldsymbol{y}, \boldsymbol{u} \mid \boldsymbol{x}, \boldsymbol{w}) \cdot \zeta(\boldsymbol{y}) \\ &= \sum_{\boldsymbol{y} \in \mathcal{Y}, \boldsymbol{u} \in \mathcal{U}} p(\boldsymbol{y} \mid \boldsymbol{u}, \boldsymbol{x}, \boldsymbol{w}) p(\boldsymbol{u} \mid \boldsymbol{x}, \boldsymbol{w}) \cdot \zeta(\boldsymbol{y}) \end{aligned} \tag{4}$$

In this formulation, the reward function ζ is integrated over the space of $\boldsymbol{Y} \times \boldsymbol{U}$ that consists of the final outputs $\boldsymbol{y}$ and chains of thought $\boldsymbol{u}$. What enables such a transformation is that (a) the dynamics function is trivial; (b) the intermediate rewards for transitions within $\boldsymbol{u}$ are zero; (c) γ is normally ignored and set to one in the CoT setting. This view enables us to use some of the insights from Bayesian methods applied to LLMs to estimate the value term.

3 RREDCoT

RUDDER (Arjona-Medina et al., 2019) is an algorithm for decomposing and redistributing the rewards for delayed reward problems. In Deep Reinforcement Learning (DRL), it manages to mitigate the bias of the TD methods and the variance of the MCMC methods by learning an additional LSTM model (Hochreiter and Schmidhuber, 1997) that is then used for credit assignment to intermediate actions, speeding up the convergence of on-policy methods. Arjona-Medina et al. (2019) describe the optimal reward redistribution $\tilde{\mathcal{P}}$ as such, that any future reward $\kappa(T-t-1, t) = 0$ for $0 \leq t \leq T-1$. Under such redistribution, the following conditions are equivalent and sufficient for optimal reward redistribution:

$$\begin{gathered} \kappa(T-t-1, t) = 0 \\ \Leftrightarrow \\ \mathrm{E}[R_{t+1} \mid \boldsymbol{s}_{t-1}, a_{t-1}, \boldsymbol{s}_t, a_t] = \tilde{q}^\pi(\boldsymbol{s}_t, a_t) - \tilde{q}^\pi(\boldsymbol{s}_{t-1}, a_{t-1}) \end{gathered} \tag{5}$$

We can use the action values derived for CoT MDP in Eq. (3) together with Eq. (2) and the second definition of optimal reward redistribution in Eq. (5) to derive an optimal reward redistribution for the CoT trace. Let us consider the scenario where the model is in state $\boldsymbol{s}_t$, $a_t \notin \mathcal{Y}$, and $a_{t-1} \notin \mathcal{Y}$:

$$\mathrm{E}[R_{t+1} \mid \boldsymbol{s}_{t-1}, a_{t-1}, \boldsymbol{s}_t, a_t] = \tag{6}$$

$$= q^{\boldsymbol{w}}(\boldsymbol{s}_t, a_t) - q^{\boldsymbol{w}}(\boldsymbol{s}_{t-1}, a_{t-1}) \tag{7}$$

$$\begin{aligned} &= \mathrm{E}_{\boldsymbol{y} \sim p(\boldsymbol{y} \mid \boldsymbol{s}_{t+1})}[\zeta(\boldsymbol{y})] - \mathrm{E}_{\boldsymbol{y} \sim p(\boldsymbol{y} \mid \boldsymbol{s}_t)}[\zeta(\boldsymbol{y})] \\ &\quad + \mathrm{E}_{\boldsymbol{u}_{t+1}}[v^{\boldsymbol{w}}(\boldsymbol{s}_{t+2})] - \mathrm{E}_{u_t}[v^{\boldsymbol{w}}(\boldsymbol{s}_{t+1})] \end{aligned} \tag{8}$$

The part of the sum in Eq. (7) can be computed by direct estimation of expectations in Eq. (1) using the model with parameters $\boldsymbol{w}$. To estimate these expectations under the output

distribution of an autoregressive models one could use multinomial sampling as was done by Hammoud et al. (2025). At the same time, under $\{0, 1\}$ outcome reward ζ and a given small set of reference answers is known, we can estimate the Eq. (1) using a PR estimator.

The value part of the reward redistribution (Eq. (8)) requires estimating an expectation over the plausible subsequent steps. Estimating this quantity would require sampling for every segment, which would render it either too slow to be practically applied to online training or require a considerable reduction in segment number, yielding coarser credit assignment. However, this expectation would generally be low if the probabilities of the actions $p(\boldsymbol{u}_t \mid \boldsymbol{s}_t, \boldsymbol{w})$ are small. This gives rise to our segmentation strategy in the following section which seeks to decouple the fragments from existing CoT

Hybrid Segmentation Strategy While our algorithm can be used for token-level attribution, this would require thousands of evaluations for every trace, rendering it impractical. The segmentation strategy may influence the subsequent credit assignment and additional computational costs. Several prior works consider the segmentation of CoT traces. For instance, Hammoud et al. (2025) define subthoughts based on special sequences, e.g. "Wait", "But", etc. More generic substring segmentation approaches are also possible, such as splitting on double newlines. (Guo et al., 2025) and (Gong et al., 2026) propose segmentation strategies based on token probabilities. While keyword based approaches are inherently anthropomorphic and may lead to suboptimal segmentation from the algorithmic standpoint, the purely entropy or probability based approaches can fall into the trap of synonyms, where the high immediate entropy does not lead to actual plurality of the subsequent strings.

We propose using a hybrid keyword-entropy segmentation approach (Fig. 2 (Left)). The hybrid segmentation starts with splitting by generic keywords, such as a newline character, and later iteratively merging consecutive segments with the lowest combined entropy until the desired consolidated number of exit points is reached. High entropy tokens have high plurality and are therefore opportune 'exit points' for estimating the immediate contributions. Additionally, we can then homogenize segment likelihood and allow better estimation of the value term in Eq. (8).

Credit Assignment Once the segmentation is done, we must assign intermediate advantages or proportions thereof to every segment. The goal of this stage is to construct an efficient estimator for the Eq. (7) and (8). Here, it is important to delineate credit assignment from the attribution analysis. The question of attribution analysis is "what contributed to obtaining this trajectory" whereas the question of credit assignment is "what contributed to getting this reward". In other words, credit assignment introduces additional information about the reward, while the attribution analysis operates merely on the sampled trajectory and environment dynamics.

With the PR estimator we can estimate not only the immediate term in Eq. (7), but also get an estimate of the value term. We can achieve that by using a reference solution path, which is usually provided in the datasets. Immediate reward estimates in this case can be viewed as a distance to the goal state, where the generation of the desired high utility output is inevitable.

Our PR-style estimator for the value function is then as follows:

$$\begin{aligned}\hat{v}^{\text{our}}_{\boldsymbol{w}}(\boldsymbol{s}_t) &= \sum_{\boldsymbol{y},\boldsymbol{u} \sim q} \frac{p(\boldsymbol{y} \mid \boldsymbol{u}, \boldsymbol{x}, \boldsymbol{w}) p(\boldsymbol{u} \mid \boldsymbol{x}, \boldsymbol{w})}{q(\boldsymbol{y}, \boldsymbol{u} \mid \boldsymbol{x}, \boldsymbol{w})} \cdot \zeta(\boldsymbol{y}) \\ &= \frac{1}{N} \sum_{\substack{\boldsymbol{y} \in \{\mathcal{Y}^\star\} \\ \boldsymbol{u} \in \{\mathcal{U}^\star\}}} p(\boldsymbol{y} \mid \boldsymbol{u}, \boldsymbol{x}, \boldsymbol{w}) p(\boldsymbol{u} \mid \boldsymbol{x}, \boldsymbol{w}) \cdot \zeta(\boldsymbol{y}) \end{aligned} \tag{9}$$

Where q is a proposal distribution which we define as a uniform distribution over selected reference answer-solution pairs $\mathcal{Y}^\star$ and $\mathcal{U}^\star$. These sequences are a predetermined set of answers and corresponding solution paths accordingly. The crucial quality that the members of the set must possess is non-zero utility ζ. In the simplest case, this set consists of a single reference answer and solution. Alternatively, it can feature multiple combinations

of answers, solutions coming from different sources, including the model's own high utility solution answer pairs and solutions generated by teacher models.

$\hat{v}^{\text{our}}_{\boldsymbol{w}}$ is an importance sampling estimator that uses the biased proposal distribution of the available solution paths. Let us derive the extent of bias of this point estimator using the unbiased MC estimator (for detailed transformations refer to Appx. D.5):

$$\begin{aligned}\text{Bias}\left[v_{\boldsymbol{w}}(\boldsymbol{s}_t), \hat{v}^{\text{our}}_{\boldsymbol{w}}(\boldsymbol{s}_t)\right] &= \mathrm{E}[\hat{v}^{\text{our}}_{\boldsymbol{w}}(\boldsymbol{s}_t)] - v_{\boldsymbol{w}}(\boldsymbol{s}_t) \\ &= -\sum_{\boldsymbol{y}\in\mathcal{Y}\setminus\mathcal{Y}^\star, \boldsymbol{u}\in\mathcal{U}\setminus\mathcal{U}^\star} \frac{1}{N} \underbrace{p(\boldsymbol{y}, \boldsymbol{u} \mid \boldsymbol{x}, \boldsymbol{w})}_{\text{answer \& solution prob.}} \cdot \underbrace{\zeta(\boldsymbol{y})}_{\text{utility}} \end{aligned} \tag{10}$$

where Z is the cardinality of the set of all possible solutions and answers for the autoregressive model and N is the cardinality of the reference solution set. We note that Z, being the number of all possible sequences generated by the model, is finite since the maximum length must be specified for the autoregressive decoding algorithm (Malinin and Gales, 2020). Essentially, this means that for every pair $\boldsymbol{y}^\star$ and $\boldsymbol{u}^\star$ we sum the integrand over all continuations that are not this specific one. Under the practically sensible assumption that ζ is non-negative, this bias is less or equal to zero and will lead to underestimation of the value function. In the best case scenario, our proposal distribution q captures all sequences that have non-zero probability and non-zero utility leading to zero bias.

While explicitly estimating the bias in Eq. (10) is intractable, we can draw some hypotheses about it by leveraging the empirical insights from Bayesian methods for LLMs. It is known that the predictive distribution of Language Models is structured and will generally contain clusters that are correlated (Kuhn et al., 2023; Farquhar et al., 2024). With this in mind, if we take a difference between $\hat{v}^{\text{our}}_{\boldsymbol{w}}(\cdot)$ evaluated at $\boldsymbol{s}_t$ and $\boldsymbol{s}_{t+1}$ as is required to estimate the Eq. (8), we can expect the difference of the probabilities of the reference solution to be indicative of the dynamics for the whole cluster. One scenario where this bias could be substantial is when there are multiple diverse solutions that are plausible under the model $\boldsymbol{w}$ and have high utility for the given problem $\boldsymbol{x}$. If the solutions are diverse, they might be parts of different clusters that are disjoint in terms of probability. In this case, the mass in Eq. (10) would have a high magnitude and it would be possible that the estimates obtained through the difference of $\hat{v}^{\text{our}}_{\boldsymbol{w}}(\cdot)$ could even be nonsensical.

Subgoal structure of reference solution. We assume that the regions of the predictive distribution of the reasoning model that are disjoint from the region containing the reference solution, yet contain sequences with positive utility are rare. Given the body of work in Bayesian Language Modeling (i.e., Aichberger et al. (2024)) we feel such an assumption is safe. (DeepSeek-AI et al., 2025) and the majority of subsequent RLVR works consider mathematics datasets (e.g. Zhang and Math-AI (2024); Hendrycks et al. (2021b)) which have the advantage of having unique solutions (down to symbolic transformation). Non-mathematical problems, such as logic and programming (Stojanovski et al., 2025; Hendrycks et al., 2021a), also provide unique solutions down to invariances. In many problems with composite answers (i.e., answers that contain multiple parts that can be independently correct or wrong), such as Sudoku puzzles or Python programs, the subgoal structure is revealed even without an explicit solution path. For example, in a Sudoku puzzle, the placement of each number is a subtask and when we see the filled-out answer to it, we can assess the achievement of individual subgoals.

Often a reference solution path is not available. As we show in the later section, the reward redistribution requires more information than just the produced sequence itself. The manner in which such information is obtained (whether the reference solution or an extensive tree search) is beyond the scope of our work. In this case one can consider starting with applying a search algorithm of choice which attempts to find a solution for a given problem. This solution path can then be summarized and used for redistribution. We assume that some hint of the subgoal structure is available during training.

Table 1: RREDCoT performance improvements on Numina-CoT (Li et al., 2024) dataset with long generation length (25k tokens).

Model	AIME24	AIME25	AIME26	Minerva	MATH500
Qwen3-4B Instruct 2507	.692	.429	.267	.906	.781
GRPO	.850	**.600**	.442	.915	.804
RREDCoT	**.908**	.583	**.475**	**.935**	**.823**

3.1 Integrating Reward Redistribution into Commonly Used RL Objectives

The RREDCoT redistribution approximation derived so far depends only on the properties of the CoT generation MDP. This means that it can be integrated into any RL objective so long as it is applied to a problem of that MDP structure. One important condition of reward redistribution is return-equivalence, meaning that the original episode return must be preserved in the reward redistribution SDP. We further reformulate the generalized policy gradient objective from Shao et al. (2024) to include the redistribution of rewards:

$$\nabla_\theta \mathcal{J}_\mathcal{A}(\theta) = \mathrm{E}_{\underbrace{(q,o)\sim\mathcal{D}}_{\text{Data Source}}} \left[\sum_{t=0}^{|o|} \underbrace{\sigma(t,q,o,\pi_{\text{rf}})}_{\text{Redistribution}} \underbrace{GC_\mathcal{A}(q,o,\pi_{\text{rf}})}_{\text{Gradient Coefficient}} \underbrace{\nabla_\theta \log \pi_\theta(o_t \mid q, o_{<t})}_{\text{Token-Wise Gradient}} \right] \tag{11}$$

Where $\sigma(t,q,o,\pi_{\text{rf}})$ is the token-wise reward coefficient that depends on the reference model and question-answer pair. The policy gradient formulations attribute reward uniformly over the whole sequence Eq. (12):

$$\sigma(t,q,o,\pi_{\text{rf}}) = \frac{1}{|o|} \tag{12}$$

σ must sum up to 1 in order to satisfy the return equivalence from Arjona-Medina et al. (2019). We note, that this formulation does not bias the original objective according to Theorem 1 in Arjona-Medina et al. (2019) if the condition holds. This leads to the same optimal policy. In practice, the diversity of data and breadth of autoregressive predictive distribution mean that full convergence i.e. perfect knowledge of the general reasoning, is unattainable in practice. As a consequence, speeding up the convergence rate would mean de facto improvement of final performance.

Several GRPO improvements use the normalizer σ for reward shaping without adhering to these conditions, therefore changing the optimal policy. For example, BNPO (Xiao et al., 2025) uses $\sigma = \frac{1}{|\mathcal{B}|}$ where $\mathcal{B}$ is the number of tokens in a batch. DR-GRPO (Liu et al., 2025b) uses $\sigma = \frac{1}{|\mathcal{M}|}$ where $\mathcal{M}$ is the maximum number of tokens allowed in the batch.

4 Experiments

In this section, we evaluate the proposed RREDCoT. Additional implementation details of the experiments as well as hyperparameters are provided in Appx. E.

Table 2: RREDCoT models, small-scale application of the reward redistribution to online refinement of small reasoning models. Notably, the RREDCoT models were tuned with the context size of only 1024. The tuning was performed on the `open-rs` dataset (Dang and Ngo, 2025), while the `open-rs` models were taken as is from HuggingFace.

Model	AIME24	AMC23	MATH500	MINERVA	Olympiad Bench
Qwen2.5-1.5B R1-Distilled	.300	.675	.840	**.320**	.521
Open-RS	**.366**	<u>.725</u>	<u>.848</u>	.316	<u>.524</u>
RREDCoT-Nano	<u>.333</u>	**.800**	**.858**	.305	**.531**

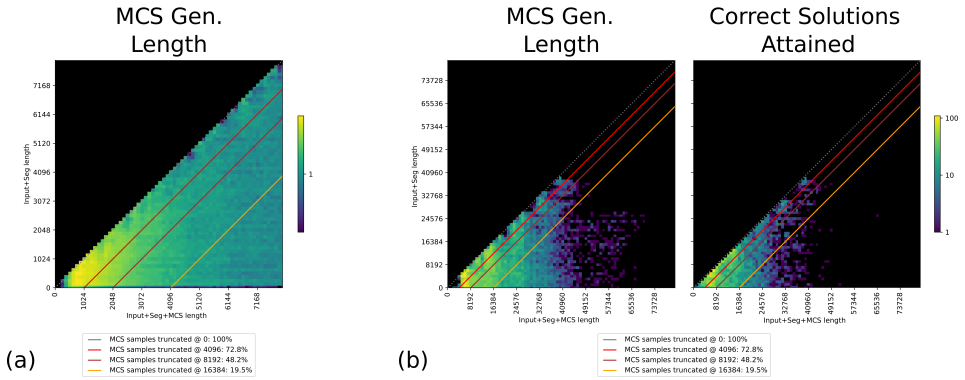


Figure 3: Proportion of rollouts lost by the truncated MC sampling with respect to truncation length. (a) MATH-500 dataset; (b) AIME-25 dataset. Computing MC sampling estimates in (b) took 100 GPU-hours for the 30 questions given and maximum number of segments of 40. In both cases we see that truncation of the MC generations leads to loss of substantial amount of the completions, including those that lead to correct answers.

Variance and Bias of Truncated MC Value Estimator. We conducted an analysis of the variability of the estimates of intermediate values by MC sampling. In Appx.Fig. 7, we show how the number of MC samples impacts the standard deviation of the estimates. We observe SD of the estimator falling below the 0.1 mark after approximately 5 samples. In this example, we have computed the MC values by generating sequences with a fixed total horizon, so the effects of increased variance at the early stages of CoT generation are not visible. Such computation would be impractical at these levels of granularity for on-policy optimization. In Fig. 3 we show that truncating the MC sampling completions can lead to loss of a large number of completions, including ones that lead to correct answers. This introduces a bias similar to that in Eq. (10).

Correlation analysis between attribution and credit assignment methods. A number of attribution techniques exist, such as Shapley values (Beechey et al., 2023), Leave One Out (LOO) (Liu et al., 2025a; Khandoga et al., 2026) and gradient-based techniques (Sundararajan et al., 2017). Attribution methods have their advantages and drawbacks, and there is no one-size-fits-all. Generally, attribution in neural networks is a computationally intensive procedure. We want to test the hypothesis that attribution unconditioned on additional information is disjoint from credit assignment/reward redistribution and would be ill-suited for explicitly weighing the tokens during training. The results of the correlation analysis are presented in Fig. 4. The gradient attribution is highly correlated with LOO attribution, meaning that using the LOO attribution values as an explicit weighing signal at training time would be redundant. This shows that using LOO style attribution analysis for the purposes of assigning intermediate rewards would be redundant at training time, since much of its allocation would be implicitly performed by the gradient descent. We speculate that this would also negate the effect of any other purely attribution explicit weight assignment, including, for example, attention weight-based attribution. At the same time, RREDCoT reward redistribution shows relatively high correlation, especially towards the later parts of the trajectories, where the MC estimates might be more precise.

On Policy LM Fine-Tuning. We have utilized the objective in Eq. (11) to assess the utility of our reward redistribution. We used 4B parameter Qwen3 Instruct model as a starting point with maximum generation length set to 25k tokens. The optimization was performed for 500 steps with hyperparameters detailed in Appx.E. The σ values for redistribution were normalized using L1 norm. This aligns conveniently with using PR approaches (Yu et al., 2025b; Zhou et al., 2025) to estimate the reward, as the sum of all intermediate rewards in log space equals the log PR. For GRPO, standard verifiers along were used along with several beneficial adjustments, such as sequence level importance (Zheng et al., 2025) and implementation level importance weights as is provided in the Transformer

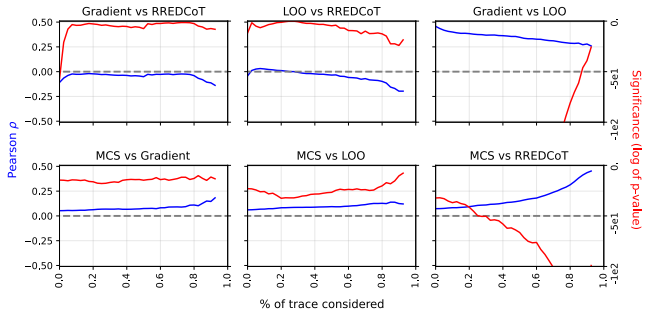


Figure 4: Correlation between LOO, Gradient attribution, MC sampling and RREDCoT credit assignment techniques. On the x-axis we have the percent of the trace used from the right, e.g., a value of 0.25 means that from each trace, the first 25% of segments were used to compute correlation. Blue lines are correlation coefficients (left scale) while the red lines are their corresponding log(p-values) (right scale). All values were computed using the model that produced the CoT. The gradient norm and LOO, being attribution methods, correlate highly with each other, while RREDCoT, being a credit assignment method, is most similar to MC sampling, especially at the later stages.

Reinforcement Learning library (von Werra et al., 2020). For each question in evaluation datasets 8 independent rollouts have been produced. Each output was evaluated using an ensemble of the LLM-as-a-Judge models with two prompts 8 samples each (Ielanskyi et al., 2026) for robustness. Tab. 1 shows a confident increase in efficiency when using our method for longer generation lengths.

Redistributing Reward Using Model's Own Answer. In order to check if our objective works with bootstrapping the model on its own answers, we have utilized the objective in Eq. (11) but used the models own answer for redistribution. The advantage terms were estimated as in GRPO (Eq. (13)). Since the advantage values were computed from the models own answers, we could normalize the σ values using the softmax, making the σ values positive and sum up to 1. Log-differences of transitions were used as attribution values to estimate the optimal redistribution (Eq. (7)). The results are presented in Tab. 2. Best of 2 checkpoints were taken for RREDCoT and best of 3 for open-rs over a short training run (300 steps). The resulting model compares favorably to those obtained by (Dang and Ngo, 2025) using GRPO and hyperparameter tuning.

5 Conclusion and Outlook

In this work, we introduce a new reward redistribution and credit assignment method RREDCoT which is intended to improve the sample efficiency of reasoning models fine-tuning. We have described the method and its versatile compatibility with the existing RLVR / RLPR pipelines and optimization objectives, conducted evaluations regarding its empirical benefits and discussed its relation to alternatives. The estimator agrees well with the values provided by our thorough MC sampling estimate while being considerably less computationally expensive. We additionally show that traditional attribution methods capture different aspects of CoT generation than reward redistribution and MC-based credit assignment. These insights show that attribution methods that are not conditioned on the correct solutions are bound to capture aspects not related to arriving at the desired state. We anticipate future work to further explore the application of Bayesian methods to credit assignment problems in this increasingly important setting. The RREDCoT can function as both a training signal amplifier and an analysis tool.

AI use statement

For the purposes of this paper, AI was used in a minimal fashion, largely to improve the prose of existing text.

References

Lukas Aichberger, Kajetan Schweighofer, and Sepp Hochreiter. Rethinking Uncertainty Estimation in Natural Language Generation, 2024. URL http://arxiv.org/abs/2412.15176.

Jose A. Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. RUDDER: Return decomposition for delayed rewards. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/16105fb9cc614fc29e1bda00dab60d41-Paper.pdf.

Daniel Beechey, Thomas M. S. Smith, and Ozgur Simsek. Explaining Reinforcement Learning with Shapley Values, 2023. URL http://arxiv.org/abs/2306.05810.

Nikhil Chandak, Shashwat Goel, and Prabhu Ameya. Incorrect Baseline Evaluations Call into Question Recent LLM-RL Claims | Notion, 2025. URL https://safe-lip-9a8.notion.site/Incorrect-Baseline-Evaluations-Call-into-Question-Recent-LLM-RL-Claims-2012f1fbf0ee8094ab8ded1953c15a37.

Quy-Anh Dang and Chris Ngo. Reinforcement Learning for Reasoning in Small LLMs: What Works and What Doesn't, 2025. URL http://arxiv.org/abs/2503.16219.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, and et al. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning, 2025. URL http://arxiv.org/abs/2501.12948.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630, 2024.

Xue Gong, Qi Yi, Ziyuan Nan, Guanhua Huang, Kejiao Li, Yuhao Jiang, Ruibin Xiong, Zenan Xu, Jiaming Guo, Shaohui Peng, and Bo Zhou. Segmental Advantage Estimation: Enhancing PPO for Long-Context LLM Training, 2026. URL http://arxiv.org/abs/2601.07320.

Yiran Guo, Lijie Xu, Jie Liu, Dan Ye, and Shuang Qiu. Segment Policy Optimization: Effective Segment-Level Credit Assignment in RL for Large Language Models, 2025. URL http://arxiv.org/abs/2505.23564.

Hasan Abed Al Kader Hammoud, Hani Itani, and Bernard Ghanem. Beyond the Last Answer: Your Reasoning Trace Uncovers More than You Think, 2025. URL http://arxiv.org/abs/2504.20708.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850, Bangkok, Thailand, 2024.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. Measuring Coding Challenge Competence With APPS. *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 1, 2021a.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring Mathematical Problem Solving With the MATH Dataset. *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 1, 2021b.

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9:1735–1780, 1997.

Mykyta Ielanskyi, Kajetan Schweighofer, Lukas Aichberger, and Sepp Hochreiter. Addressing pitfalls in the evaluation of uncertainty estimation methods for natural language generation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=OxWnOV5q8w.

Dulhan Jayalath, Shashwat Goel, Thomas Foster, Parag Jain, Suchin Gururangan, Cheng Zhang, Anirudh Goyal, and Alan Schelten. Compute as Teacher: Turning Inference Compute Into Reference-Free Supervision, 2025. URL http://arxiv.org/abs/2509.14234.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. VinePPO: Refining Credit Assignment in RL Training of LLMs, 2025. URL http://arxiv.org/abs/2410.01679.

Mykola Khandoga, Rui Yuan, and Vinay Kumar Sankarapu. Beyond Uniform Credit: Causal Credit Assignment for Policy Optimization, 2026. URL http://arxiv.org/abs/2602.09331.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, Chuning Tang, Congcong Wang, Dehao Zhang, Enming Yuan, and et al. Kimi k1.5: Scaling Reinforcement Learning with LLMs, 2025. URL https://arxiv.org/abs/2501.12599.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *The Eleventh International Conference on Learning Representations*, 2023.

Aitor Lewkowycz, Anders Johan Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Venkatesh Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving Quantitative Reasoning Problems with Language Models. In *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=IFXTZERXdM7.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. NuminaMath, 2024.

Ziheng Li, Liu Kang, Feng Xiao, Luxi Xing, Qingyi Si, Zhuoran Li, Weikang Gong, Deqing Yang, Yanghua Xiao, and Hongcheng Guo. Outcome-Grounded Advantage Reshaping for Fine-Grained Credit Assignment in Mathematical Reasoning, 2026. URL http://arxiv.org/abs/2601.07408.

Fengyuan Liu, Nikhil Kandpal, and Colin Raffel. AttriBoT: A Bag of Tricks for Efficiently Approximating Leave-One-Out Context Attribution, 2025a. URL https://arxiv.org/abs/2411.15102.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-Like Training: A Critical Perspective, 2025b. URL http://arxiv.org/abs/2503.20783.

Andrey Malinin and Mark Gales. Uncertainty Estimation in Autoregressive Structured Prediction. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=jN5y-zb5Q7m.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. S1: Simple test-time scaling, 2025. URL http://arxiv.org/abs/2501.19393.

Weixuan Ou, Yanzhao Zheng, Shuoshuo Sun, Wei Zhang, Baohua Dong, Hangcheng Zhu, Ruohui Huang, Gang Yu, Pengwei Yan, and Yifan Qiao. SERL: Self-Examining Reinforcement Learning on Open-Domain, 2025. URL http://arxiv.org/abs/2511.07922.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll 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. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, 2017. URL http://arxiv.org/abs/1707.06347.

Rulin Shao, Shuyue Stella Li, Rui Xin, Scott Geng, Yiping Wang, Sewoong Oh, Simon Shaolei Du, Nathan Lambert, Sewon Min, Ranjay Krishna, Yulia Tsvetkov, Hannaneh Hajishirzi, Pang Wei Koh, and Luke Zettlemoyer. Spurious Rewards: Rethinking Training Signals in RLVR, 2025. URL https://arxiv.org/abs/2506.10947.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models, 2024. URL http://arxiv.org/abs/2402.03300.

Chengyu Shen, Zhen Hao Wong, Runming He, Hao Liang, Meiyi Qiang, Zimo Meng, Zhengyang Zhao, Bohan Zeng, Zhengzhou Zhu, Bin Cui, and Wentao Zhang. Let's verify math questions step by step, 2025. URL https://arxiv.org/abs/2505.13903.

Zafir Stojanovski, Oliver Stanley, Joe Sharratt, Richard Jones, Abdulhakeem Adefioye, Jean Kaddour, and Andreas Köpf. REASONING GYM: Reasoning Environments for Reinforcement Learning with Verifiable Rewards, 2025. URL https://arxiv.org/abs/2505.24760.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic Attribution for Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 3319–3328, 2017. URL https://proceedings.mlr.press/v70/sundararajan17a.html.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. TRL: Transformers Reinforcement Learning, 2020. URL https://github.com/huggingface/trl.

Guoqing Wang, Sunhao Dai, Guangze Ye, Zeyu Gan, Wei Yao, Yong Deng, Xiaofeng Wu, and Zhenzhe Ying. Information Gain-based Policy Optimization: A Simple and Effective Approach for Multi-Turn Search Agents. 2020. URL https://openreview.net/forum?id=qkWP6phrvZ.

Changyi Xiao, Mengdi Zhang, and Yixin Cao. BNPO: Beta Normalization Policy Optimization, 2025. URL http://arxiv.org/abs/2506.02864.

Guofu Xie, Yunsheng Shi, Hongtao Tian, Ting Yao, and Xiao Zhang. CAPO: Towards Enhancing LLM Reasoning through Verifiable Generative Credit Assignment, 2025. URL http://arxiv.org/abs/2508.02298.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, and et al. Qwen3 Technical Report, 2025. URL http://arxiv.org/abs/2505.09388.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, and et al. DAPO: An Open-Source LLM Reinforcement Learning System at Scale, 2025a. URL http://arxiv.org/abs/2503.14476.

Tianyu Yu, Bo Ji, Shouli Wang, Shu Yao, Zefan Wang, Ganqu Cui, Lifan Yuan, Ning Ding, Yuan Yao, Zhiyuan Liu, Maosong Sun, and Tat-Seng Chua. RLPR: Extrapolating RLVR to General Domains without Verifiers, 2025b. URL http://arxiv.org/abs/2506.18254.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime), 2024.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group Sequence Policy Optimization, 2025. URL http://arxiv.org/abs/2507.18071.

Xiangxin Zhou, Zichen Liu, Anya Sims, Haonan Wang, Tianyu Pang, Chongxuan Li, Liang Wang, Min Lin, and Chao Du. Reinforcing General Reasoning without Verifiers, 2025. URL http://arxiv.org/abs/2505.21493.

A Limitations

Our proposed method relies on the knowledge of the solution to the problem and some degree of subgoal structure. Normally this would be an existing solution trace that is not atypical under the distribution of general text. While a vast majority of the post-training datasets known to us provide such information, it cannot be taken for granted. Our method cannot be simply applied to the problems for which no hint of solution path is available or the provided reference solution is uninformative. One category of such problems is constraint satisfaction problems. Mitigation of this could be attempted with the use of bootstrapping on the models own correct answers (as was assessed in the last paragraph (Sec.4/P.4) of the preceding section). This, however, will only work if a positive signal is attainable by the model and precludes the use of PR based approaches to sequence level advantage estimation.

Furthermore our method involves additional computational overhead compared to the simpler GRPO variants. In practice our optimization runs with RREDCoT required $1.5 - 2$x the amount of compute as the runs without redistribution. This is still much faster compared to the MC sampling based intermediate value estimates and we believe that it is a reasonable trade-off. Our implementation of value function estimation for the intermediate values was aggressively optimized for reuse of KV values and batched inference. The peak GPU memory utilization was unchanged from naive GRPO.

B Datasets and Models

B.1 Datasets Used

B.1.1 Dataset derived from `open-rs`

The models were trained on `open-rr` dataset. `open-rr` dataset is derived from `open-rs` dataset (Dang and Ngo, 2025). Despite the dataset having been compiled from other math datasets, most of which are supposed to be curated to any extent, our manual inspection has determined that it contained an error rate of $15\% \pm 5\%$. Almost all of the relatively trivial-looking questions designated as 'Hard' and having no step-by-step solution were found to have incorrect labels. This could additionally explain why these questions were difficult for the model to learn: the reference solutions for them were incorrect! We have filtered the dataset further from $7,000$ examples to include only the examples containing the solution path, as well as on the length of the reference solution.

B.1.2 Evaluation Datasets

We used the following evaluation datasets: AIME24/25 (Zhang and Math-AI, 2024), AMC, MATH-500 (Hendrycks et al., 2021b), Minerva (Lewkowycz et al., 2022) and OlympiadBench (He et al., 2024). The verification was done with (Shen et al., 2025). The maximum generation length was chosen to be 4096 for the experiments with 1.7B models and $25k$ tokens for the `Qwen3-4B` based experiments.

B.2 Models Used

We have made extensive use of Qwen2.5 distilled to Deepseek R1 traces (DeepSeek-AI et al., 2025) and Qwen3 (Yang et al., 2025) families of models. We avoided MoE models due to the increased complexity of training dynamics with them.

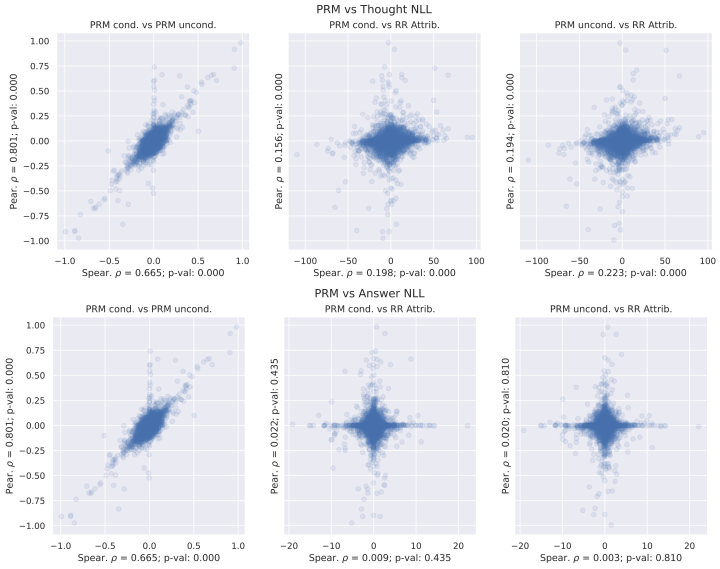


Figure 5: Correlation of PRM prediction to the different components of RREDCoT attribution. PRM models seem to be more correlated to NLL changes of the thought NLL than to the change of NLL of the reference answer even when conditioned on it.

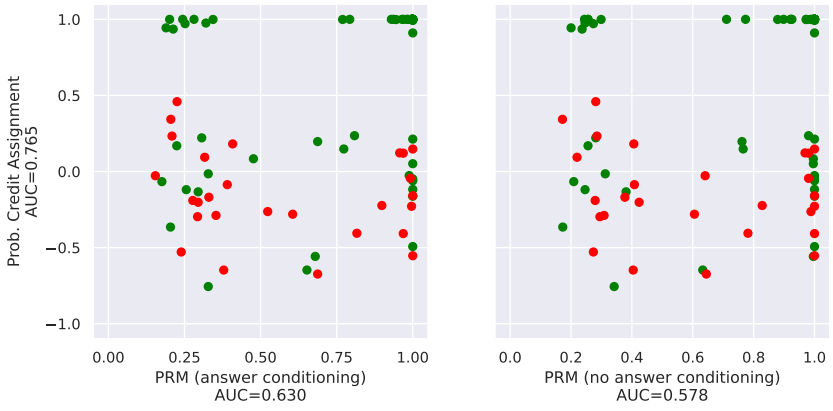


Figure 6: Correlation between the PRM and our credit assignment approach on 100 examples from `open-rs` dataset. Green indicates that the ultimate answer was correct according to the verifiers, while red points are incorrect. The x and y axes are the attribution by the PRM and our method, accordingly. The CoT traces were generated using `Deepseek-R1-Qwen2.5-7B-Distill` model while the PRM was computed using `Qwen2.5-Math-PRM-7B`. Our attribution method results in better correlation to the ground truth than the PRM approach, even when the PRM is conditioned on the correct answer.

C Additional Figures

D Additional Equations

D.1 Sequence Return Estimation

The RLPR and RLVR estimators are as follows:

$$R_{\boldsymbol{u}}^{VR} \approx \frac{1}{n}\sum_{i=0}^{n} \zeta(\boldsymbol{y}_i); \quad \boldsymbol{y}_i \sim p(. \mid \boldsymbol{u}, \boldsymbol{x}, \boldsymbol{w}) \tag{13}$$

$$R_{\boldsymbol{u}}^{PR} \approx \sum_{i=0}^{n} \zeta(\boldsymbol{y}_i)\; p(\boldsymbol{y}_i \mid \boldsymbol{u}, \boldsymbol{x}, \boldsymbol{w}); \quad \boldsymbol{y}_i \sim \{\boldsymbol{y}^\star\}^n \tag{14}$$

Theorem : if $\boldsymbol{y}^\star$ contains all $\boldsymbol{y}$ such that $\zeta(\boldsymbol{y}_i) > 0$ then Eq. 13 and Eq. 14 converge to the same $R_{\boldsymbol{y}}$.

This can be an obstacle for the practical application of RLPR when, e.g., the number of solutions is large or unbounded.

Wang et al. (2020) successfully use the probability reward in multi-turn search to estimate the rewards at each of the sequential calls without the need for final outcome.

D.2 Transformations of Bellman Equation for CoT MDP

Value function decomposition for CoT MDP:

$$\begin{aligned}
v_{\boldsymbol{w}}(\boldsymbol{s}_t) &= \mathrm{E}[G_t \mid S_t = \boldsymbol{s}_t] \\
&= \mathrm{E}[R_{t+1} + \gamma G_{t+1} \mid S_t = \boldsymbol{s}_t] \\
&= \sum_a \pi(a \mid \boldsymbol{s}_t) \sum_{\boldsymbol{s}_{t+1}, r} p(\boldsymbol{s}_{t+1}, r \mid \boldsymbol{s}_t, a)\,[r + \gamma v_\pi(\boldsymbol{s}_{t+1})] \\
&= \sum_a \pi(a \mid \boldsymbol{s}_t) \sum_{\boldsymbol{s}_{t+1}, r} p(\boldsymbol{s}_{t+1}, r \mid \boldsymbol{s}_t, a)\; r + \sum_a \pi(a \mid \boldsymbol{s}_t) \sum_{\boldsymbol{s}_{t+1}, r} p(\boldsymbol{s}_{t+1}, r \mid \boldsymbol{s}_t, a)\; \gamma\; v_\pi(\boldsymbol{s}_{t+1}) \\
&= \sum_{\boldsymbol{y}} p(\boldsymbol{y} \mid \boldsymbol{s}, \boldsymbol{w})\; r(\boldsymbol{y}) + \sum_{\boldsymbol{u}_t} p(\boldsymbol{u}_t \mid \boldsymbol{s}_t, \boldsymbol{w}) \sum_{\boldsymbol{u}_n} p(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, \boldsymbol{u}_t)\; \gamma\; v_\pi(\boldsymbol{s}_{t+1}) \\
&= \sum_{a \in \mathcal{Y}} p(a \mid \boldsymbol{s}_t, \boldsymbol{w})\; r(\boldsymbol{y}) + \sum_{a \in \mathcal{U}} p(a \mid \boldsymbol{s}_t, \boldsymbol{w})\; \gamma\; v_{\boldsymbol{w}}(\boldsymbol{s}_{t+1})
\end{aligned} \tag{15}$$

This uses the return at time step t, defined as $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$.

Alternatively, we can reformulate the Bellman equation as estimation of a quantity under the Bayesian posterior of the reasoning language model $\boldsymbol{w}$:

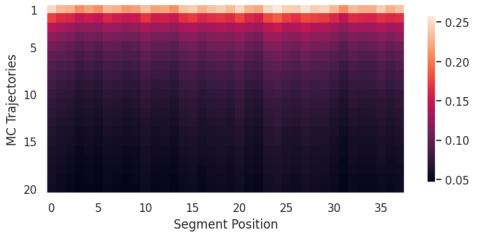


Figure 7: Standard Deviation of the MC value estimator. The values were obtained by bootstrapping the original values from 20 completions. Vertical axis is the number of MC samples - the number of rollouts selected for value estimation. The horizontal axis is the number of exit points where the value was evaluated.

$$
\begin{aligned}
v_{\boldsymbol{w}}(\boldsymbol{s}_t) &= \mathrm{E}[G_t \mid S_t = \boldsymbol{s}_t] \\
&= \mathrm{E}[R_{t+1} + \gamma G_{t+1} \mid S_t = \boldsymbol{s}_t] \\
&= \sum_a \pi(a \mid \boldsymbol{s}_t) \sum_{\boldsymbol{s}_{t+1}, r} p(\boldsymbol{s}_{t+1}, r \mid \boldsymbol{s}_t, a) \left[r + \gamma v_\pi(\boldsymbol{s}_{t+1}) \right]
\end{aligned} \tag{16}
$$

From this point onward, given that (a) the dynamics function is trivial and (b) that γ is set to 1. we can continue as follows:

$$
\begin{aligned}
&= \mathrm{E}_{\boldsymbol{y},\boldsymbol{u} \sim p(\boldsymbol{y},\boldsymbol{u} \mid \boldsymbol{x},\boldsymbol{w})} \Big[\zeta(\boldsymbol{y}) \Big] \\
&= \sum_{\boldsymbol{Y},\boldsymbol{U}} p(\boldsymbol{y}, \boldsymbol{u} \mid \boldsymbol{x}, \boldsymbol{w}) \cdot \zeta(\boldsymbol{y}) \\
&= \sum_{\boldsymbol{Y},\boldsymbol{U}} p(\boldsymbol{y} \mid \boldsymbol{u}, \boldsymbol{x}, \boldsymbol{w}) p(\boldsymbol{u} \mid \boldsymbol{x}, \boldsymbol{w}) \cdot \zeta(\boldsymbol{y})
\end{aligned} \tag{17}
$$

D.3 CoT Optimal Reward Redistribution Identity

Detailed derivation of the optimal reward redistribution for CoT generation:

$$
\begin{aligned}
\mathrm{E}[R_{t+1} \mid \boldsymbol{s}_{t-1}, a_{t-1}, \boldsymbol{s}_t, a_t] &= q^{\boldsymbol{w}}(\boldsymbol{s}_t, a_t) - q^{\boldsymbol{w}}(\boldsymbol{s}_{t-1}, a_{t-1}) \\
&= v^{\boldsymbol{w}}(\boldsymbol{s}_{t+1}) - v^{\boldsymbol{w}}(\boldsymbol{s}_t) \qquad (18) \\
&= \mathrm{E}_{\boldsymbol{y} \sim p(\boldsymbol{y} \mid \boldsymbol{s}_{t+1}, \boldsymbol{w})}[r(\boldsymbol{y})] + \mathrm{E}_{u_{t+1} \sim p(u_t \mid \boldsymbol{s}_{t+1}, \boldsymbol{w})}[v_\pi(\boldsymbol{s}_{t+2})] - \\
&\quad - \mathrm{E}_{\boldsymbol{y} \sim p(\boldsymbol{y} \mid \boldsymbol{s}_t, \boldsymbol{w})}[r(\boldsymbol{y})] - \mathrm{E}_{\boldsymbol{u}_t \sim p(\boldsymbol{u}_t \mid \boldsymbol{s}_t, \boldsymbol{w})}[v_\pi(\boldsymbol{s}_{t+1})] \\
&= \left[\mathrm{E}_{\boldsymbol{y} \sim p(\boldsymbol{y} \mid \boldsymbol{s}_{t+1}, \boldsymbol{w})}[r(\boldsymbol{y})] - \mathrm{E}_{\boldsymbol{y} \sim p(\boldsymbol{y} \mid \boldsymbol{s}_t, \boldsymbol{w})}[r(\boldsymbol{y})] \right] + \qquad (19) \\
&\quad + \left[\mathrm{E}_{\boldsymbol{u}_{t+1} \sim p(\boldsymbol{u}_{t+1} \mid \boldsymbol{s}_{t+1}, \boldsymbol{w})}[v^{\boldsymbol{w}}(\boldsymbol{s}_{t+2})] - \mathrm{E}_{\boldsymbol{u}_t \sim p(\boldsymbol{u}_t \mid \boldsymbol{s}_t, \boldsymbol{w})}[v^{\boldsymbol{w}}(\boldsymbol{s}_{t+1})] \right] \\
& \qquad (20)
\end{aligned}
$$

D.4 Other Equations

General policy gradient (Shao et al., 2024):

$$
\nabla_\theta \mathcal{J}_{\mathcal{A}} = \mathrm{E} \underbrace{[(q, o) \sim \mathcal{D}]}_{\text{Data Source}} \left(\frac{1}{|o|} \sum_{t=0}^{|o|} \underbrace{GC_{\mathcal{A}}(q, o, t, \pi_{rf})}_{\text{Gradient Coefficient}} \underbrace{\nabla_\theta \log \pi_\theta(o_t \mid q, o_{<t})}_{\text{Token-Wise Gradient}} \right) \tag{21}
$$

D.5 Value Estimator

Deriving the bias of the $\hat{v}^{our}(\boldsymbol{s}_t)$:

$$
\begin{aligned}
\mathrm{Bias}\left[v_{\boldsymbol{w}}(\boldsymbol{s}_t), \hat{v}^{our}_{\boldsymbol{w}}(\boldsymbol{s}_t) \right] &= \mathrm{E}[\hat{v}^{our}_{\boldsymbol{w}}(\boldsymbol{s}_t)] - v_{\boldsymbol{w}}(\boldsymbol{s}_t) \\
&= \frac{1}{N} \sum_{\boldsymbol{y},\boldsymbol{u} \sim q} p(\boldsymbol{y} \mid \boldsymbol{u}, \boldsymbol{x}, \boldsymbol{w}) p(\boldsymbol{u} \mid \boldsymbol{x}, \boldsymbol{w}) \cdot \zeta(\boldsymbol{y}) \\
&\quad - \frac{1}{Z} \sum_{\boldsymbol{y} \in \mathcal{Y}, \boldsymbol{u} \in \mathcal{U}} p(\boldsymbol{y} \mid \boldsymbol{u}, \boldsymbol{x}, \boldsymbol{w}) p(\boldsymbol{u} \mid \boldsymbol{x}, \boldsymbol{w}) \cdot \zeta(\boldsymbol{y}) \qquad (22) \\
&= - \sum_{\boldsymbol{y} \in \mathcal{Y} \setminus \mathcal{Y}^\star, \boldsymbol{u} \in \mathcal{U} \setminus \mathcal{U}^\star} \frac{Z - N}{NZ} p(\boldsymbol{y} \mid \boldsymbol{u}, \boldsymbol{x}, \boldsymbol{w}) p(\boldsymbol{u} \mid \boldsymbol{x}, \boldsymbol{w}) \cdot \zeta(\boldsymbol{y}) \\
&\approx - \sum_{\boldsymbol{y} \in \mathcal{Y} \setminus \mathcal{Y}^\star, \boldsymbol{u} \in \mathcal{U} \setminus \mathcal{U}^\star} \frac{1}{N} p(\boldsymbol{y} \mid \boldsymbol{u}, \boldsymbol{x}, \boldsymbol{w}) p(\boldsymbol{u} \mid \boldsymbol{x}, \boldsymbol{w}) \cdot \zeta(\boldsymbol{y}) \qquad (23)
\end{aligned}
$$

For these transformations, we use the true value of the value function in Eq. 22, which is summed over the space of all possible final outputs and solution paths. The penultimate

transition is made by moving the factor inside the sum and essentially means that for every pair $\boldsymbol{y}^\star$ and $\boldsymbol{u}^\star$ we sum the integrand over all sequences that are not the specific sequence from the reference pool. The final transition is made by using the fact that $N \ll Z$. This bias is non-positive. It is zero if our q covers the entirety of $\mathcal{Y}$,$\mathcal{U}$.

E Details on Experimental Settings

E.1 MC Sampling Variance Experiment

We have computed a single rollout for each question in the MATH-500 dataset. The maximum completion length was 8196. The generation and evaluation were performed with the Qwen3-4B-Thinking-2507 model. Segmentation was performed using the hybrid method with a maximum number of segments set to 40. 20 completions were generated at every exit point, each keeping the maximum completion lengths constraint. This procedure took approximately 80 GPU-hours on RTX 5000 PRO. 100 samples were subsampled to bootstrap the estimator.

E.2 Attribution and Credit Assignment Method Correlation

The same rollouts and model as in the previous subsection were used. In addition, we have computed gradient, LOO and RREDCoT values for the segments.

The LOO values were computed as pointwise mutual information between the segment and all of the subsequent segments under the model's predictive distribution. The gradient attribution was computed by computing the gradient of logprobs of the selected completion tokens with respect to input embeddings. To get per segment value out of that, the first-order norm of these gradients was taken and averaged per segment. This is reminiscent of the gradient propagation occurring during the model update.

The loo value computation took 0.5 GPU-hours, while RREDCoT and gradient took about 0.3 GPU-hours. The considerably more computationally expensive MC sampling was used as a proxy for ground truth value segment-wise advantage estimates.

E.3 On Policy Model Training

Small Scale Experiments with `open-rs` dataset. The setting and hyperparameters from (Dang and Ngo, 2025) were used. Hybrid segmentation was performed with the maximum segment number of 120. Completion length of 1024 were used in training.

For validation during training, we used avg@4 strategy with 4 datasets: Minerva (first 50), AIME24 (full), AIME25 (full) and MATH-500 (first 50). The generation length was limited to 4096 tokens for performance reasons. Answer extraction and validation were performed using the `math-verify` package. This experiment is estimated to have consumed on the order of 500 GPU-hours.

Experiments with `Qwen3-4B-Instruct-2507` Numina-CoT (Li et al., 2024) dataset was used to train reasoning starting with `Qwen3-4B-Instruct-2507` (Yang et al., 2025). The shared hyperparameters of GRPO and RREDCoT run were identical, with the key difference being the learning rate that was $1.e-6$ for GRPO and $5.e-7$ for RREDCoT. Note, that the learning rate was kept at the given value for the GRPO as it was found to perform worse with $5.e-7$. In other words, the LR is individually tailored for each of the two algorithm. The group size was set to 4, gradient accumulation to 1, total step size to 500 with cosine LR scaling.

For validation we used avg@8 strategy with 5 datasets: AIME24, AIME25, AIME26, Minerva-Math and MATH-500. The generation length was limited to $25K$, similar to training, to assess the method performance at larger scale. Answer extraction and validation were performed using the `math-verify` package. This experiment is estimated to have consume on the order of 1000 GPU-hours.