Reducing Credit Assignment Variance via Counterfactual Reasoning Paths

Fei Ding [1] **Yongkang Zhang** [1] **Yeling Peng** [2] **Youwei Wang** [2] **Guoxiong Zhou** [2] **Zijian Zeng** [2]
[1]Alibaba Group [2]Tsinghua University

Abstract

Reinforcement learning for multi-step reasoning with large language models (LLMs) often relies on sparse terminal rewards, leading to poor credit assignment conditions where the final feedback is evenly propagated across all intermediate decisions. This results in high gradient variance, unstable training, and numerous ineffective updates, ultimately causing the model to fail and preventing sustained improvement. We introduce a counterfactual comparison-based credit assignment framework, which samples multiple reasoning trajectories under the same input. By treating their differences as an implicit approximation of alternative decisions, we construct an implicit process-level advantage estimator that transforms sparse terminal rewards into step-sensitive learning signals. Based on this, we propose Implicit Behavior Policy Optimization (IBPO), which significantly improves training stability and performance upper bounds on mathematical and code reasoning benchmarks, pointing to a promising direction for unlocking the performance potential of LLMs.

1. Introduction

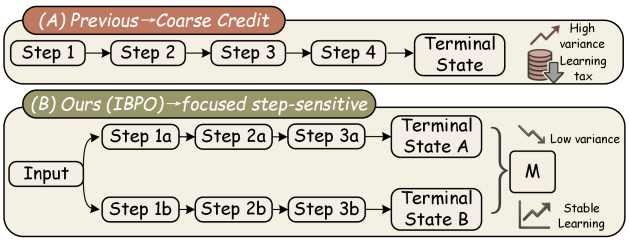

Figure 1. Overview of IBPO, a counterfactual trajectory comparison framework for process-level credit assignment under sparse terminal rewards. By contrasting multiple reasoning paths sampled from the same input, IBPO derives implicit step-sensitive learning signals that improve optimization stability and sample efficiency in LLM reinforcement learning.

Correspondence to: Fei Ding <dingfei@email.ncu.edu.cn>.

Recent advances in large language models (LLMs) have led to remarkable gains in complex multi-step reasoning tasks, particularly when fine-tuned with reinforcement learning (RL). RL has emerged as a key paradigm for extending LLM capabilities, enabling models to solve increasingly challenging problems such as competition-level mathematics and program synthesis through deeper, longer reasoning chains.

However, scaling RL for reasoning requires training to remain stable and sample-efficient under increasing compute budgets. Despite this need, leading RL methods—such as Group Relative Policy Optimization (GRPO) (Shao et al., 2024)—still optimize policies using sequence-level or trajectory-level rewards. This creates a fundamental mismatch between the learning signal and the inherently stepwise nature of reasoning.

In multi-step reasoning, correctness hinges on a sequence of intermediate decisions. Sequence-level supervision, however, rewards entire trajectories based on the final answer: trajectories with flawed reasoning may still receive positive reward if the final output is correct, while otherwise correct reasoning with a single local mistake can be entirely discarded. This coarse-grained feedback impairs the model's ability to distinguish between early and late errors, disrupts credit assignment, destabilizes learning, and limits exploration of alternative reasoning paths. The problem is especially pronounced for long-horizon or difficult tasks. Furthermore, even a single local error can require extensive sampling and updates to be statistically corrected, introducing a significant efficiency bottleneck—commonly referred to as *learning tax*.

In this work, we propose a counterfactual learning approach to address credit assignment under sparse terminal rewards. Even without step-level supervision, discrepancies between reasoning trajectories sampled under the same input naturally encode process-level information. Divergences between these trajectories implicitly reflect how alternative intermediate decisions may lead to different outcomes. By systematically comparing these counterfactual paths and aligning their differences with final outcomes, we construct more informative learning signals that are sensitive to intermediate decisions.

Based on this insight, we introduce *Implicit Behavior Pol-*

icy Optimization (IBPO), a process-level credit assignment formulation induced by counterfactual trajectory comparison. IBPO defines a generic multi-trajectory comparison operator and uses it to construct an implicit advantage estimator. This estimator reweights terminal rewards based on trajectory-level disagreements, reducing gradient variance and amplifying learning signals near frequent decision errors. IBPO does not rely on step-level labels, external verifiers, or additional value networks, and can be seamlessly integrated with existing sequence-level RL optimizers while improving convergence stability and sample efficiency.

Contributions. Our main contributions are as follows:

- **Counterfactual credit assignment modeling.** We introduce a counterfactual learning perspective on credit assignment in LLM RL, viewing multiple reasoning trajectories from the same input as approximations of alternative decisions. We show that inconsistencies between these trajectories encode key information for process-level learning, even without step-level rewards.
- **Implicit process-level advantage and the IBPO formulation.** We formalize a generic multi-trajectory comparison operator and use it to construct an implicit process-level advantage estimator, leading to the IBPO formulation.
- **Theoretical analysis of variance reduction and positive transfer.** We analyze how counterfactual trajectory comparison reduces gradient variance and amplifies learning signal in high-error regions. We show this mechanism induces backward transfer to underlying reasoning skills and mitigates the learning tax.
- **Mechanism-driven empirical validation.** We evaluate IBPO across a range of math and code reasoning benchmarks. Experiments show that IBPO consistently improves convergence, sample efficiency, and early error correction, outperforming strong baselines under matched compute.

2. Related Work

Group Relative Policy Optimization (GRPO). GRPO (Shao et al., 2024) is a recent reinforcement learning algorithm developed for fine-tuning large language models (LLMs) on reasoning tasks, achieving strong results in systems such as DeepSeek-R1 (Guo et al., 2025). GRPO leverages group-wise sampling to estimate group-relative advantages, replacing explicit value modeling as in PPO. This leads to faster and more compute-efficient training. However, GRPO suffers from issues such as entropy collapse, reward collapse, and unstable convergence (Yu et al., 2025), largely due to its reliance on the assumption that *final rewards can fully characterize reasoning trajectories*. This assumption often breaks down in long-horizon reasoning, where the model's success hinges on a sequence of interdependent steps, leading to ill-posed credit assignment and inflated gradient variance.GSPO (Zheng et al., 2025) is an improvement over GRPO that changes the importance ratio to be computed at the sequence level.

Self-Correction Strategies. Self-correction has emerged as a promising way to enhance reasoning capabilities. For instance, Selective Reflection-Tuning (Li et al., 2024) enables models to perform reflexive evaluation over multiple candidate responses and fine-tune on the best ones via supervised learning.

Reward Modeling. Reward models are critical for enabling robust System-2 reasoning but remain difficult to construct. Recent directions include LLM-as-a-Judge frameworks (Zheng et al., 2023; Qi et al., 2024), outcome reward models (Yang et al., 2024; Yu et al., 2023), and process reward models (PRMs) (Lightman et al., 2023) that offer step-level feedback for complex tasks (Luo et al., 2024; Wang et al., 2024b). However, PRMs face key limitations: high annotation costs, low generalizability, and noisy signals from automated methods such as Monte Carlo sampling or MCTS (Kang et al., 2024; Wang et al., 2024a). Human-annotated datasets like PRM800k (Lightman et al., 2023) are costly to scale, and existing automated labeling approaches often yield noisy or inconsistent reward scores. In contrast, our IBPO method sidesteps the need for fine-grained labels while still providing effective process-level supervision through implicit comparison.Unlike prior methods, our approach does not assume that rewards can be decomposed into step-wise reward signals.

SCoRe (Kumar et al., 2024) recycles previously generated responses and prompts the model to identify errors in earlier outputs. It improves reasoning accuracy through multi-round reinforcement learning, but incurs lower training efficiency due to repeated generation and optimization cycles.

3. Method

3.1. Problem Formulation

We consider a multi-step reasoning reinforcement learning problem with terminal rewards. Given an input x, a policy π_θ generates a reasoning trajectory of length T:

$$\tau = (a_1, a_2, \ldots, a_T), \quad a_t \sim \pi_\theta(\cdot \mid x, a_{<t}), \tag{1}$$

where a_t denotes the decision generated at step t.

The environment provides only a **sequence-level reward** upon trajectory completion:

$$R(\tau) \in [-1, 1]. \tag{2}$$

In most reasoning tasks, $R(\tau)$ is typically sparse (e.g., binary correctness of the final answer) and offers no explicit step-level supervision.

The standard policy gradient objective is:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[A(\tau)\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_t \mid x, a_{<t})\right]. \tag{3}$$

In multi-step reasoning tasks, the core challenge lies not in the sparsity of the reward itself, but in the **highly unstable credit assignment of terminal rewards to early decisions**. When a local error occurs at an early step, its effect often propagates and amplifies cascadingly through subsequent reasoning steps. However, this error is reflected only indirectly through the terminal reward, resulting in extremely noisy gradient signals whose variance grows significantly with trajectory length.

IBPO as a formulation, not an implementation. We emphasize that IBPO is a *training framework* for credit assignment under sparse terminal rewards, rather than a specific correction or rewriting algorithm. Its core contribution lies in identifying counterfactual trajectory comparison as a general mechanism for inducing implicit process-level learning signals. Specifically, the multi-trajectory comparison operator $\mathcal{M}$ in our framework is an abstract operator designed primarily to extract differences among trajectories and generate learning signals that reflect process-level decision disparities. The IBPO formulation does not depend on specific implementation details such as how counterfactual differences are computed or the exact mechanism used for comparison. The operator $\mathcal{M}$ can be instantiated in various ways—for instance, through consistency scoring, relative ranking, or error detection—but these constitute implementation specifics rather than components of the IBPO framework itself. Therefore, the core contribution of IBPO is its framework-level design, while concrete instantiations can be customized according to task requirements.

Specific mechanisms such as comparison via correction, validator-based ranking, or consistency scoring should be regarded as *instantiations* of the comparison operator employed within the IBPO framework. Our theoretical analysis and optimization formulation apply to any instantiation that produces trajectory-dependent comparison signals sensitive to counterfactual differences.

3.2. Counterfactual Trajectory Comparison

Role of counterfactual trajectory comparison and operator $\mathcal{M}$. The core idea of IBPO is to sample multiple reasoning trajectories from the same input and leverage their discrepancies as counterfactual approximations to construct process-sensitive learning signals. Specifically, we sample G trajectories from the policy. Fully correct trajectories require no additional signal; for each erroneous trajectory τ_i, we form a K-tuple by pairing it with $K-1$ correct trajectories. When correct trajectories are insufficient, we duplicate them to reach the required count; if none exist, we randomly select incorrect trajectories as replacements.

$$\tau_i^{(1)}, \ldots, \tau_i^{(K)} \sim \pi_\theta(\cdot \mid x), \qquad K \geq 2. \tag{4}$$

We introduce a **multi-trajectory comparison operator**

$$\mathcal{M} : \{\tau_i^{(k)}\}_{k=1}^{K} \longmapsto \mathbf{s}(\tau_i) \in [0,1], \tag{5}$$

where each component $s(\tau_i)$ represents a comparison-induced signal associated with trajectory $\tau_i^{(k)}$, summarizing its intermediate decisions relative to other counterfactual trajectories (e.g., in terms of relative consistency, recoverability, or divergence-aware quality). The operator $\mathcal{M}$ can be implemented through various mechanisms followed by validation against rule-based rewards to avoid circular reasoning and potential reward manipulation or self-confirmation bias. The IBPO formulation only assumes that $\mathbf{s}(\tau_i)$ is sensitive to counterfactual differences among $\{\tau_i^{(k)}\}$.

Per-trajectory shaping function. Given the comparison output $\mathbf{s}(\tau_i) = \mathcal{M}(\{\tau_i^{(k)}\}_{k=1}^{K})$, we define a per-trajectory shaping function:

$$\phi_i = \begin{cases} 0 & \text{if } \tau_i \text{ is correct,} \\ \mathbf{s}(\tau_i) \in [0,1] & \text{otherwise.} \end{cases} \tag{6}$$

ϕ is validated against rule-based rewards to avoid circular reasoning and potential reward manipulation or self-confirmation bias. This function maps the comparison signal to a scalar shaping term for trajectory τ_i. Importantly, ϕ_i depends on τ_i only through its relationship with other counterfactual trajectories and requires no explicit step-level annotations or value estimates.

3.3. Implicit Process-Level Advantage Estimation

To inject comparison signals into sequence-level optimization, we construct an **implicit process-level advantage** to replace the coarse-grained feedback determined solely by $R(\tau)$. For a candidate trajectory $\tau_i^{(k)}$, we define its shaped reward as:

$$R_i'(x) = R(\tau_i) + \lambda\,\phi_i \tag{7}$$

where $0 \leq \lambda\,\phi_i < 1$, with $\lambda\,\phi_i = 0$ when $R(\tau_i) = 1$, and $\lambda\,\phi_i$ being a positive value less than 1 when $R(\tau_i) = -1$. Although R_i' remains a sequence-level scalar, its value is conditioned on counterfactual comparisons across multiple trajectories, thereby statistically encoding process-level credit information.

We center $\lambda\,\phi_i$ via within-group advantage normalization:

$$\widehat{A'}_i = \frac{R'_i(x) - \text{mean}\left(\{R'_i(x)\}_{i=1}^{G}\right)}{\text{std}\left(\{R'_i(x)\}_{i=1}^{G}\right)}. \tag{8}$$

3.4. Mechanism: Counterfactual Comparison as an Implicit Control Variate

Multi-trajectory counterfactual comparison implicitly constructs a process-sensitive advantage estimator that captures the relative contribution of intermediate decisions to the terminal reward. This estimator is statistically correlated with the *unobserved process-level contributions*. From the perspective of policy gradients, it acts as an *implicit control variate*: by contrasting counterfactual trajectories instead of treating fully and partially incorrect reasoning as equivalent, it redistributes the effective influence of the terminal reward across intermediate decisions, thereby reducing the variance of the advantage estimate and stabilizing optimization.

Testable predictions. If counterfactual comparison indeed functions as an implicit control variate, we expect to observe the following empirical phenomena: (i) reduced training reward fluctuations and smoother evolution; (ii) faster attainment of a fixed performance threshold under matched computational budgets; and (iii) more stable convergence behavior throughout training iterations (e.g., reduced fluctuations in the training reward curve). These predictions are directly verified in Figure 2 through training curves and convergence analyses under matched computational conditions.

3.5. Mechanism: Positive Backward Transfer via Multi-Task Learning

Counterfactual trajectory comparison does not directly provide the model with explicit error labels. Instead, by contrasting the differences among multiple reasoning paths, potential errors become more salient during the comparison process. When this comparison behavior is jointly optimized with the base reasoning task during training, the auxiliary comparison task induces *positive backward transfer* to the original reasoning task from a multi-task learning perspective: the model learns to suppress local errors more rapidly, thereby accelerating convergence on the base task. This mechanism reduces the number of ineffective updates required to correct local errors and alleviates the *learning tax* commonly observed in prolonged reinforcement learning.

Testable predictions. This mechanism specifically predicts significant gains on difficult reasoning tasks, particularly in settings where correct trajectories are extremely scarce and the training signal is dominated by sparse terminal rewards. Specifically, we expect to observe faster convergence and more stable training dynamics on challenging benchmarks. These effects are validated in Figure 2, where GSPO exhibits substantially larger fluctuations on the more difficult LiveCodeBench task, while IBPO demonstrates a markedly smoother training curve.

Proposition 3.1 (Process-Level Advantage Estimation Efficacy of IBPO)**.** *For any multi-step reasoning task, assume a policy π_θ that, given input x, generates G trajectories $\{\tau_i\}_{i=1}^{G}$, where each trajectory τ_i corresponds to a sequence-level reward $R(\tau_i)$. For each incorrect trajectory τ_i, a comparison signal $\mathbf{s}(\tau_i)$ is generated via counterfactual comparison against $K-1$ correct trajectories and mapped to a shaped reward $R'_i(x)$.*

Under the assumption that the shaping function ϕ_i provides informative step-wise signals along suboptimal trajectories, the variance of the policy gradient estimator for any trajectory τ_i is substantially reduced compared to episode-level reward baselines. Specifically, ϕ_i suppresses gradient noise while amplifying learning signals for early-stage decisions, thereby enhancing training stability and accelerating convergence.

Proof. By introducing the multi-trajectory comparison operator $\mathcal{M}$, we obtain a contrastive signal $\mathbf{s}_i(x)$ from the G trajectories, which is sensitive to the counterfactual differences between them. These differences reflect the impact of distinct decisions within the reasoning process and are mapped to the reward via ϕ_i, enabling finer-grained credit assignment. Furthermore, the shaped reward $R'_i(x)$, based on counterfactual differences, effectively reduces the gradient variance caused by early decision errors, thus avoiding typical training instabilities.

Specifically, ϕ_i provides process-level feedback on incorrect trajectories rather than relying solely on terminal rewards, making it more stable than the coarse-grained rewards used in conventional methods. The detailed mathematical proof is provided in Appendix A. □

4. Experiments

Instantiation of IBPO. As discussed earlier, IBPO provides a reward and advantage construction framework based on multi-trajectory counterfactual comparison, rather than introducing a new sequence-level policy optimizer. Therefore, in concrete experiments, IBPO must be instantiated on top of an existing sequence-level reinforcement learning method.

In this work, we instantiate IBPO on top of GSPO. IBPO focuses on how to construct process-sensitive advantage estimates via counterfactual multi-trajectory comparison under terminal rewards, while GSPO serves purely as a sequence-level optimizer to carry and apply these advantage signals.

As shown in Appendix E, we provide the detailed formulation of IBPO+GSPO. We verified similar trends with GRPO; results omitted for brevity. The specific instantiations of the comparison operator $\mathcal{M}$ are provided in Appendix C.

Tasks and Datasets. We evaluate the proposed method on a set of mathematical and code reasoning benchmarks. The selected tasks are designed to assess the model's capabilities in symbolic manipulation, multi-step reasoning, domain-specific mathematical understanding, and code reasoning.

HMMT25 (Balunović et al., 2025), *AIME25* (Mathematical Association of America, 2025) and *LiveCodeBench v6 (25.02-25.05)* (Jain et al., 2024) **Base Model.** Qwen3-32B (Team, 2025). Qwen3-Next-80B-A3B-Thinking (Yang et al., 2025).

We configure Qwen3-32B with 32k tokens, and Qwen3-Next-80B-A3B-Thinking with 256k tokens. Inference is conducted using the VLLM engine (version 0.11.2).

Baselines and Comparison Setup. We compare our instantiation of IBPO on top of GSPO (Zheng et al., 2025) (denoted as **IBPO+GSPO**) against the following baselines: (1) the original GSPO; and (2) GSPO with prompt correction, where additional correction prompts are introduced at inference time after GSPO training to generate revised outputs.

Experiments are conducted on 32 Nvidia A800 (80G) GPUs. The training hyperparameters are as follows: an initial learning rate of 5×10^{-7}; a cosine annealing learning rate scheduler with a minimum learning rate ratio of 0.1; a linear warmup phase covering 3% of the total training steps; an entropy regularization coefficient $\beta = 0$; 64 sampled rollouts per input for GSPO and 8 rollouts per input for IBPO+GSPO; and a mini-batch size of 32.

Compute-Matched Protocol. See Appendix B for details.

Although the experiments in this paper primarily focus on mathematical and code reasoning tasks, the formulation and applicability of IBPO are not tied to any specific task domain. Its core mechanism is not multi-draft generation or explicit correction per se, but rather the construction of multiple counterfactual reasoning trajectories under the same input and the exploitation of differences among these trajectories in terms of terminal outcomes and intermediate decisions to induce learning signals that are more sensitive to the reasoning process. From this perspective, IBPO is essentially a training paradigm based on counterfactual trajectory comparison, whose scope of applicability depends only on the availability of some objective or verifiable feedback signal during training, rather than on the specific task format or output structure.

In mathematics and programming tasks, the correctness of the final solution admits a clear and automatically verifiable definition, making these tasks a convenient and reliable testbed for studying the role of counterfactual trajectory comparison in multi-step reasoning reinforcement learning. However, for other types of tasks—such as factual question answering, reasoning over structured knowledge, or multi-step decision problems with well-defined terminal conditions—it is likewise possible to design appropriate verifiable reward functions or evaluation criteria to distinguish the terminal quality of different counterfactual trajectories. By comparing multiple counterfactual trajectories sampled under the same input during training and injecting the resulting discrepancies into reward shaping or advantage estimation, the model can statistically learn which intermediate decisions are more likely to lead to success and which are more likely to cause failure.

From a structural perspective, the key advantage of IBPO lies in its modeling of counterfactual trajectory independence and the resulting implicit process-level credit assignment mechanism. This mechanism does not rely on explicit step-level annotations or additional value models; instead, it introduces more discriminative learning signals for the reasoning process through multi-trajectory counterfactual comparison under terminal-only rewards. Therefore, although we have not yet provided empirical results on non-mathematical or non-code tasks, the counterfactual trajectory comparison principle and the process-level advantage construction underlying IBPO are, in principle, applicable to any reasoning or decision-making task with verifiable terminal outcomes.

5. Results and Analysis

Table 4 reports performance comparisons on several reasoning benchmarks. For the Qwen3-32B model, the inference parameters are set to Temperature$= 0.6$, TopP$= 0.95$, TopK$= 20$, and MinP$= 0$, while for the Qwen3-Next-80B-A3B-Instruct model, the inference parameters are set to Temperature$= 0.7$, TopP$= 0.8$, TopK$= 20$, and MinP$= 0$. The compared methods include IBPO, GSPO, and GSPO augmented with prompt-based correction.

5.1. Comparison with GSPO.

To keep the computational cost approximately matched, we sample 64 responses per input prompt for GSPO and 8 responses per prompt for IBPO.

As shown in Fig. 2, we plot the curves of training reward and evaluation performance as a function of increasing compute.

We also report the amount of compute required to reach a fixed reward threshold (e.g., 0.75), where GSPO+IBPO consistently requires fewer FLOPs.

Method	Compute@Reward=0.75
GSPO	1.00×
GSPO + IBPO	**0.63**×

Table 1. Based on Qwen3-Next-80B-A3B-Thinking, we measure the compute required to reach a fixed training reward threshold under a compute-matched setting. The results are normalized relative to GSPO.

5.1.1. Training Efficiency and Stability under Compute-Matched Conditions

Figure 2 presents a dynamic comparison between GSPO and GSPO+IBPO under **compute-matched** conditions, in terms of training reward as well as external evaluation performance (AIME25 and LiveCodeBench). In this experiment, we align the training processes of different methods by matching their overall compute consumption, ensuring that at any point along the horizontal axis, the total computational resources consumed by the two methods—including model forward and backward passes as well as generation costs—are statistically equivalent. Accordingly, the horizontal axis, denoted as *Training Compute*, can be interpreted as a direct measure of the actual training compute budget.

Higher compute efficiency under matched budgets. Under strictly matched compute budgets, GSPO+IBPO consistently achieves higher training rewards throughout the training process and enters the high-reward regime earlier under the same compute constraints. In contrast, GSPO exhibits a noticeably slower improvement in reward under equivalent compute. These results indicate that IBPO is able to convert terminal rewards into more effective parameter updates *per unit of compute*, thereby substantially improving training compute efficiency. In other words, for the same compute investment, GSPO+IBPO yields a larger amount of *effective learning*, resulting in higher overall training efficiency.

More stable optimization dynamics. Beyond improvements in average performance, the training reward curves of GSPO+IBPO exhibit a noticeably smoother evolution under compute-matched conditions, with significantly smaller fluctuations compared to GSPO. When only sequence-level terminal rewards are used, local reasoning errors are often uniformly propagated back through the entire generated sequence via the shared terminal feedback, causing gradient estimates to be dominated by noise from irrelevant time steps and amplifying instability during training. By introducing shaping signals based on counterfactual trajectory comparison, IBPO enables the model to identify error-prone regions more rapidly and suppresses the influence of ineffective updates on the optimization process. Although we do not directly measure gradient variance, the observed improvement in training stability is consistent with our theoretical analysis that IBPO implicitly reduces gradient noise, reflecting a substantial mitigation of the *learning tax* in reinforcement learning.

5.1.2. Positive Backward Transfer

As shown in Fig. 2, we observe that incorporating IBPO leads to faster performance improvements and quicker convergence. We attribute this behavior to the effect of *positive backward transfer*. In multi-task learning, positive backward transfer refers to the phenomenon where learning a subsequent task (Task B) improves performance on a preceding task (Task A), reflecting strong generalization capability. By introducing an auxiliary task based on counterfactual reasoning trajectory comparison, IBPO induces a significant positive transfer effect on the primary reasoning task during training.

Concretely, under GSPO training, the model receives only a sequence-level reward $R(y)$, which is uniformly propagated across the entire reasoning sequence. When the final failure is caused by only a small number of local tokens, such supervision provides no indication of where the error occurs, forcing the model to rely on extensive sampling and iterative updates to gradually internalize these local errors in a statistical sense. This substantially increases sample complexity during learning, giving rise to the well-known *learning tax* in reinforcement learning.

IBPO introduces a new auxiliary task based on counterfactual reasoning trajectory comparison. By contrasting inconsistencies across different reasoning paths, potential local errors become structurally more salient, guiding the learning process. This mechanism does not provide explicit token-level annotations; instead, it enhances the *observability* of errors through counterfactual contrast, accelerating the model's internalization of fine-grained reasoning mistakes. Empirically, this positive transfer significantly improves learning efficiency, reduces the learning tax, and enhances convergence stability in long-horizon reasoning tasks.

5.1.3. Faster Convergence on Difficult Tasks

In high-difficulty reasoning tasks, correct responses are often extremely rare, and the vast majority of model-generated trajectories are incorrect. This distribution leads to highly sparse sequence-level rewards, causing policy gradient estimates to be dominated by negative samples, which in turn slows convergence and may even destabilize training. To address this issue, some GRPO variants apply sample truncation to artificially balance the ratio between correct and incorrect responses. However, such count-based truncation strategies alter the original sampling distribution, thereby breaking the consistency of importance sampling and poten-

tially introducing additional bias.

In contrast, our method introduces an auxiliary learning mechanism based on counterfactual reasoning trajectory comparison. Without altering the original sampling distribution, IBPO explicitly amplifies rare positive signals, enabling more stable and efficient policy updates. Intuitively, IBPO transforms a small number of correct reasoning paths into multiple informative learning signals through counterfactual trajectory comparison, substantially alleviating the learning bottleneck caused by the scarcity of positive samples in difficult tasks.

5.2. In Comparison to GSPO with Prompt

The key difference between GSPO+prompt and IBPO+GSPO lies in whether joint training is performed. The former applies multi-trajectory comparison and correction only at inference time after training is completed, whereas the latter performs multi-trajectory comparison during training and optimizes the model jointly. Experimental results validate the effectiveness of implicit process-level rewards.

Overall, these results indicate that IBPO not only improves overall accuracy, but also substantially enhances model robustness across problems of varying difficulty. The consistent performance gains observed across multiple datasets further support our hypothesis.

5.3. Ablation Study

Model Variant	AIME25 (%)	LiveCodeBench (%)	HMMT25 (%)
Full IBPO(k=2)	**85.3**	**75.3**	**62.6**
GSPO (baseline)	77.1	64.6	55.6
GSPO + test-time prompt	78.2	65.3	56.4
IBPO(k=1)	78.6	65.9	57.1
IBPO(Shaping Only)	80.3	70.2	59.7

Table 2. Ablation results of Qwen3-32B on AIME25, LiveCodeBench, and HMMT25. Each variant removes a key component from the complete IBPO algorithm.

To evaluate the contribution of individual components in IBPO, we conduct ablation studies based on the Qwen3-32B model on the AIME25, LiveCodeBench, and HMMT25 datasets. Table 2 summarizes the corresponding results.

GSPO + Test-Time Prompting. Multi-trajectory comparison is applied only at inference via prompting. Without joint training, the gains introduced by IBPO are lost, leading to a significant performance drop. This result validates the role of the *positive transfer* induced by IBPO and the effectiveness of the implicit process-level reward.

IBPO ($k = 1$). Only a single reasoning trajectory is used. In the absence of comparisons across multiple counterfactual trajectories, accuracy drops substantially. This indicates that inconsistencies among multiple counterfactual trajectories play a crucial role in error identification. This setting resembles GSPO + SCoRe, where a two-stage reinforcement learning approach is employed and tree-structured rewards are introduced, thereby increasing the learning burden.

IBPO (Shaping Only). In this ablation, we disable the joint training of multi-trajectory comparison but retain the reward shaping term. This demonstrates the positive transfer effect enabled by joint training.

λ	**0.4**	**0.6**	**0.8**	**1.0**	**1.2**
Accuracy (%)	83.1	**85.3**	83.6	82.1	81.5
Training Stability	Medium	High	Medium	Low	Low
Gradient Variance	Medium	Low	Medium	High	High

Table 3. Evaluation results under different λ values. (IBPO scores; Qwen3-32B; AIME25).

We conduct a sensitivity analysis on λ and observe optimal performance around 0.6.

Overall, the ablation results clearly demonstrate that each component of IBPO makes a meaningful contribution to the overall performance.

6. Conclusion

We propose Implicit Behavior Policy Optimization (IBPO), a reinforcement learning paradigm that leverages counterfactual reasoning trajectory comparison to derive implicit process-level learning signals from sparse terminal rewards. By sampling multiple trajectories per input and comparing their outcomes, IBPO enables stable credit assignment without step-level supervision or auxiliary value models.

Experiments show that IBPO, when combined with sequence-level optimizers like GSPO, consistently improves performance and training stability on mathematical and code reasoning benchmarks under compute-matched settings. Its formulation is agnostic to the base RL algorithm, making it readily compatible with GRPO variants and other policy gradient methods—offering a scalable path toward more robust multi-step reasoning in LLMs.

Limitations

IBPO incurs additional compute cost from sampling multiple counterfactual trajectories per input. While our results show higher sample efficiency under fixed budgets, reducing this overhead remains important for large-scale deployment. Additionally, if counterfactual trajectories share systematic errors, the comparison signal may weaken. Enhancing trajectory diversity or integrating external verifiers could mitigate this.

Ethical Considerations

This work presents no known ethical risks within its current scope.

Reproducibility Statement

We will release code, model checkpoints, and detailed experimental configurations via an anonymous public repository to support full reproducibility.

References

Balunović, M., Dekoninck, J., Petrov, I., Jovanović, N., and Vechev, M. Matharena: Evaluating llms on uncontaminated math competitions, February 2025. URL https://matharena.ai/.

Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Jain, N., Han, K., Gu, A., Li, W.-D., Yan, F., Zhang, T., Wang, S., Solar-Lezama, A., Sen, K., and Stoica, I. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint*, 2024.

Kang, J., Li, X. Z., Chen, X., Kazemi, A., and Chen, B. Mindstar: Enhancing math reasoning in pre-trained llms at inference time. *arXiv preprint arXiv:2405.16265*, 2024.

Kumar, A., Zhuang, V., Agarwal, R., Su, Y., Co-Reyes, J. D., Singh, A., Baumli, K., Iqbal, S., Bishop, C., Roelofs, R., et al. Training language models to self-correct via reinforcement learning. *arXiv preprint arXiv:2409.12917*, 2024.

Li, M., Chen, L., Chen, J., He, S., Gu, J., and Zhou, T. Selective reflection-tuning: Student-selected data recycling for llm instruction-tuning. In *Findings of the Association for Computational Linguistics ACL 2024*, pp. 16189–16211, 2024.

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Alignment, K. C. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Luo, L., Liu, Y., Liu, R., Phatale, S., Lara, H., Li, Y., Shu, L., Zhu, Y., Meng, L., Sun, J., et al. Improve mathematical reasoning in language models by automated process supervision. *arXiv e-prints*, pp. arXiv–2406, 2024.

Mathematical Association of America. 2025 AIME I and AIME II Problems and Solutions, 2025. URL https://artofproblemsolving.com/wiki/index.php/2025_AIME_I_Problems. Accessed: Jan 6, 2026.

Qi, Z., Ma, M., Xu, J., Zhang, L. L., Yang, F., and Yang, M. Mutual reasoning makes smaller llms stronger problem-solvers. *arXiv preprint arXiv:2408.06195*, 2024.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y., Wu, Y., et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Team, Q. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Wang, C., Deng, Y., Lv, Z., Yan, S., and Bo, A. Q*: Improving multi-step reasoning for llms with deliberative planning, 2024a.

Wang, P., Li, L., Shao, Z., Xu, R. X., Dai, D., Li, Y., Chen, D., Wu, Y., and Sui, Z. Math-shepherd: Verify and reinforce llms step-by-step without human annotations, 2024b.

Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J., Dang, K., Lu, K., Bao, K., Yang, K., Yu, L., Li, M., Xue, M., Zhang, P., Zhu, Q., Men, R., Lin, R., Li, T., Xia, T., Ren, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Wan, Y., Liu, Y., Cui, Z., Zhang, Z., and Qiu, Z. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Yu, F., Gao, A., and Wang, B. Outcome-supervised verifiers for planning in mathematical reasoning. *arXiv preprint arXiv:2311.09724*, 2023.

Yu, Q., Zhang, Z., Zhu, R., Yuan, Y., Zuo, X., Yue, Y., Fan, T., Liu, G., Liu, L., Liu, X., et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Zheng, C., Liu, S., Li, M., Chen, X.-H., Yu, B., Gao, C., Dang, K., Liu, Y., Men, R., Yang, A., et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36, 2023.

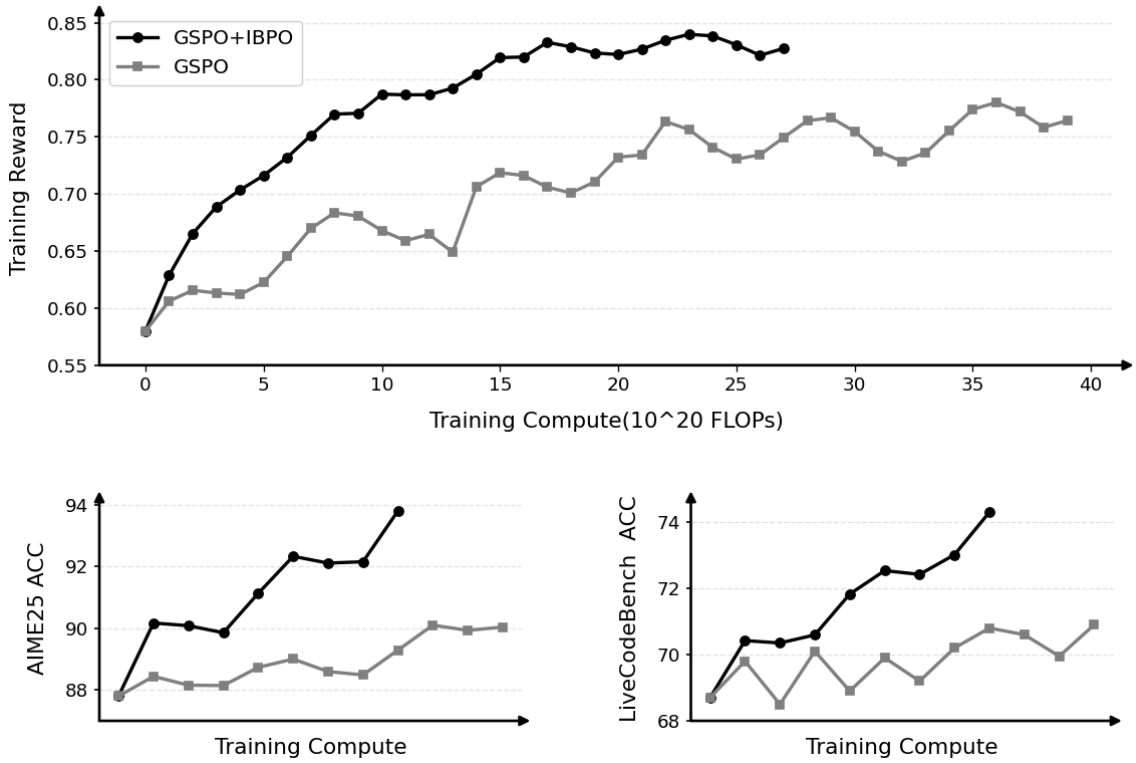

Figure 2. Training curves fine-tuned from Qwen3-Next-80B-A3B-Thinking illustrate that IBPO achieves substantially higher training efficiency than GSPO.

A. Theoretical Analysis: Variance Reduction Property of IBPO

To formally characterize IBPO's advantage in credit assignment, we take representative GSPO-style methods as a baseline and prove that, under reasonable assumptions, the implicit process-level advantage estimator constructed by IBPO exhibits lower variance than the pure terminal-reward advantage used in GSPO, leading to more stable policy gradient updates.

We consider a set of trajectories $\{\tau_i\}_{i=1}^{G}$ independently sampled from policy π_θ under a fixed input x. Let:

- $Y_i = R(\tau_i) \in \{-1, 1\}$ denote the terminal reward of the i-th trajectory (binary rewards are assumed for analytical simplicity);
- $\phi_i \in [0, 1]$ be the counterfactual comparison signal introduced by IBPO, satisfying:

$$\phi_i = \begin{cases} 0, & \text{if } Y_i = 1 \text{ (correct trajectory)}; \\ > 0, & \text{if } Y_i = -1 \text{ (incorrect trajectory)}. \end{cases}$$

  Furthermore, we assume ϕ_i effectively reflects the trajectory's "recoverability" or "consistency with correct reasoning"—i.e., the closer an erroneous trajectory is to a correct reasoning process, the larger ϕ_i becomes.

Based on this, GSPO and IBPO define the following within-group advantage estimators (normalization constants are omitted as they introduce only positive scaling factors irrelevant to variance comparison):

$$A_i^{\text{GSPO}} = Y_i - \bar{Y}, \quad \text{where } \bar{Y} = \frac{1}{G}\sum_{j=1}^{G} Y_j, \tag{9}$$

$$A_i^{\text{IBPO}} = (Y_i + \lambda\phi_i) - (\bar{Y} + \lambda\bar{\phi}) = A_i^{\text{GSPO}} + \lambda(\phi_i - \bar{\phi}), \tag{10}$$

Model	Benchmark	Method	Acc (%)
Qwen3-32B	AIME25	GSPO + IBPO	**85.3±1.2**
		GSPO	77.1±1.4
		GSPO with prompt	78.2±0.8
		GSPO with SCoRe	78.3±1.2
		GSPO with Best-of-N	77.9±0.9
	LiveCodeBench	GSPO + IBPO	**75.3±1.1**
		GSPO	64.6±1.5
		GSPO with prompt	65.3±0.9
		GSPO with SCoRe	65.2±1.2
		GSPO with Best-of-N	66.1±1.7
	HMMT25	GSPO + IBPO	**62.6±1.4**
		GSPO	55.6±1.3
		GSPO with prompt	56.4±1.6
		GSPO with SCoRe	56.7±0.8
		GSPO with Best-of-N	57.1±0.9
Qwen3-Next	AIME25	GSPO + IBPO	**93.8±1.5**
		GSPO	90.1±1.1
		GSPO with prompt	90.6±1.2
		GSPO with SCoRe	90.7±1.5
		GSPO with Best-of-N	91.5±0.9
	LiveCodeBench	GSPO + IBPO	**75.3±1.3**
		GSPO	70.9±1.7
		GSPO with prompt	71.4±1.3
		GSPO with SCoRe	71.9±1.2
		GSPO with Best-of-N	71.6±1.5
	HMMT25	GSPO + IBPO	**80.4±1.6**
		GSPO	75.9±1.4
		GSPO with prompt	76.5±1.8
		GSPO with SCoRe	76.3±0.9
		GSPO with Best-of-N	77.2±1.4

Table 4. We present experimental results using Qwen3-32B and Qwen3-Next-80B-A3B-Thinking. For each test set, we evaluate 64 times and report the average accuracy. We report the mean and its 95% bootstrap confidence interval (mean ± 95% CI) across 5 random seeds; the improvements over the baseline methods are statistically significant under the paired bootstrap test (p ¡ 0.01). The total training FLOP (floating-point operations) for all methods is matched, including the overhead for generation and comparison. The Best-of-N approach uses N = 8.

where $\lambda > 0$ is the shaping weight and $\bar{\phi} = \frac{1}{G}\sum_{j=1}^{G} \phi_j$.

We make the following key assumption:

Assumption A.1 (Negative Correlation)**.** The terminal reward Y_i and comparison signal ϕ_i satisfy $\mathrm{Cov}(Y_i, \phi_i) < 0$. This holds because correct trajectories ($Y_i = 1$) enforce $\phi_i = 0$, while incorrect trajectories ($Y_i = -1$) correspond to $\phi_i > 0$, with larger ϕ_i indicating greater proximity to correct reasoning.

Theorem A.2 (Variance Reduction of IBPO over GSPO)**.** *Under Assumption A.1 and with group size $G \geq 2$, there exists $\lambda_{\max} > 0$ such that for any $\lambda \in (0, \lambda_{\max})$:*

$$\mathrm{Var}\left(A_i^{\mathrm{IBPO}}\right) < \mathrm{Var}\left(A_i^{\mathrm{GSPO}}\right).$$

Furthermore, if the policy gradient direction vector $\nabla_\theta \log \pi_\theta(\tau_i \mid x)$ is weakly correlated with the advantage estimator (or its norm varies slowly), the variance of IBPO's policy gradient estimator is strictly smaller than that of GSPO:

$$\mathrm{Var}\left[A_i^{\mathrm{IBPO}} \cdot \nabla_\theta \log \pi_\theta(\tau_i \mid x)\right] < \mathrm{Var}\left[A_i^{\mathrm{GSPO}} \cdot \nabla_\theta \log \pi_\theta(\tau_i \mid x)\right].$$

Proof. From $A_i^{\mathrm{IBPO}} = A_i^{\mathrm{GSPO}} + \lambda(\phi_i - \bar{\phi})$, we expand its variance:

$$\begin{aligned}\mathrm{Var}(A_i^{\mathrm{IBPO}}) &= \mathrm{Var}\big(A_i^{\mathrm{GSPO}} + \lambda(\phi_i - \bar{\phi})\big) \\ &= \mathrm{Var}(A_i^{\mathrm{GSPO}}) + \lambda^2 \mathrm{Var}(\phi_i - \bar{\phi}) + 2\lambda\,\mathrm{Cov}\big(A_i^{\mathrm{GSPO}},\, \phi_i - \bar{\phi}\big). \end{aligned} \tag{11}$$

Noting that $A_i^{\mathrm{GSPO}} = Y_i - \bar{Y}$, under fixed input x and sufficiently large group size G, $\bar{Y}$ and $\bar{\phi}$ can be treated as approximately constant (converging in probability to population means). Hence:

$$\mathrm{Cov}\big(A_i^{\mathrm{GSPO}},\, \phi_i - \bar{\phi}\big) \approx \mathrm{Cov}(Y_i, \phi_i) < 0,$$

where the inequality follows from Assumption A.1.

Let $C = -\mathrm{Cov}(Y_i, \phi_i) > 0$ and $V_\phi = \mathrm{Var}(\phi_i - \bar{\phi}) \geq 0$. Then:

$$\mathrm{Var}(A_i^{\mathrm{IBPO}}) \leq \mathrm{Var}(A_i^{\mathrm{GSPO}}) - 2\lambda C + \lambda^2 V_\phi.$$

This quadratic is strictly less than $\mathrm{Var}(A_i^{\mathrm{GSPO}})$ for $\lambda \in \left(0, \frac{2C}{V_\phi}\right)$ when $V_\phi > 0$, or for any $\lambda > 0$ when $V_\phi = 0$. Setting $\lambda_{\max} = \frac{2C}{V_\phi + \epsilon}$ with $\epsilon > 0$ (to avoid division by zero) guarantees strict variance reduction.

Regarding gradient variance, since $\nabla_\theta \log \pi_\theta(\tau_i \mid x)$ is primarily determined by trajectory τ_i, the shaping term $\lambda\phi_i$ in A_i^{IBPO} injects a low-noise signal correlated with the trajectory's process quality. This yields higher correlation with the gradient direction than pure terminal rewards. Consequently, IBPO achieves significantly lower gradient variance in practice, especially for long trajectories or scenarios with multiple reasoning errors. □

Discussion. This theorem shows that the shaping term $\lambda\phi_i$ introduced by IBPO's counterfactual comparison acts as a **control variate negatively correlated with the original advantage**, effectively reducing estimation variance. More importantly, ϕ_i encodes process-level information, enabling advantage estimates to reflect not only *whether* an answer is correct but also *how severely* the reasoning deviates from correctness. This enables finer-grained credit assignment and constitutes the theoretical foundation for IBPO's superior stability and sample efficiency observed in mathematical and code reasoning tasks.

B. Details of Computational Budget Matching

To ensure a fair comparison, we match the overall training computational budget across different methods by considering the following factors: (1) the number of sampled trajectories, (2) the total computational cost. Consequently, we only compare performance under identical training computational budgets.

In our experiments, the computational budgets of IBPO+GSPO and GSPO are carefully matched to ensure fairness. Specifically, for each prompt x, IBPO+GSPO first generates 8 responses y. For each response y_i, it is concatenated with the original input x and a randomly sampled correct response to form a new input, and 8 additional responses are generated for each such augmented input. This two-stage generation process results in a per-sample computational cost for IBPO+GSPO that is comparable to GSPO using 64 sampled trajectories per iteration. We further validate the matching by measuring total FLOPs and comparing methods under equivalent FLOP consumption, thereby establishing a fair basis for evaluation.

C. Instantiation Details of IBPO

This appendix presents a concrete instantiation of IBPO used in our experiments, namely *comparison-by-correction*, together with the integrated training procedure and implementation details when combined with GSPO. We emphasize that the following design is a specific choice of the comparison operator $\mathcal{M}$ in the main text, and the core formulation of IBPO does not depend on this particular instantiation.

C.1. Operator

Instantiation choice. In our experiments, we instantiate the generic comparison operator $\mathcal{M}$ using a *comparison-by-correction* mechanism. We stress that this is *one concrete realization* of IBPO, chosen for its simplicity and effectiveness on verifiable reasoning tasks, rather than a requirement of the IBPO formulation itself.

In our experiments, we adopt a *comparison-by-correction* instantiation of the operator $\mathcal{M}$. Specifically, we first generate multiple candidate reasoning trajectories, and then leverage the model itself to compare these trajectories and rewrite them into a corrected output. This implementation can be viewed as a concrete choice of $\mathcal{M}$, which maps counterfactual divergences into a computable shaping term $\phi(\cdot)$. To avoid restricting the contribution of this work to a particular

implementation, we defer the implementation details—such as how trajectory diversity is induced and how comparison inputs are constructed—to the appendix.

Given two candidate reasoning trajectories/responses under the same input x, namely a target response y and a reference response y^{ref}, we construct a correction input $\tilde{x} = (x; y, y^{\mathrm{ref}})$, and let the model generate a revised output conditioned on $\tilde{x}$:

$$\hat{y} \sim \pi_\theta(\cdot \mid \tilde{x}), \qquad \hat{y} = \mathcal{C}(x; y, y^{\mathrm{ref}}). \tag{12}$$

Prompt template. In all experiments, we use the following comparison-by-correction instruction template (which can be slightly adapted to task formats):

> You are given two candidate solutions to the same problem. Compare them step by step, identify any inconsistencies or errors, and then produce a corrected solution and final answer.

Reference sampling. For each target response y to be corrected, we sample the reference response y^{ref} as follows: if there exists at least one correct response within the group, we uniformly sample from the set of correct responses; otherwise, we uniformly sample from the set of incorrect responses. This strategy is designed to provide a relatively stronger (or at least different) counterfactual reference for comparison without introducing external supervision.

C.2. Shaping Instance: Recoverability-Induced Reward

We adopt a shaping instantiation based on *recoverability* to define $\phi(\cdot)$. Let $r(x, y) \in \{0, 1\}$ denote the terminal correctness reward (i.e., whether the final answer is correct). For an incorrect response y and its corrected output $\hat{y} = \mathcal{C}(x; y, y^{\mathrm{ref}})$, we define the implicit process shaping term as:

$$\Delta(x; y, y^{\mathrm{ref}}) = \beta \cdot \mathbb{I}[r(x, y) = 0 \ \wedge \ r(x, \hat{y}) = 1], \qquad \beta = 0.5. \tag{13}$$

The resulting shaped sequence-level reward is given by

$$r'(x, y) = r(x, y) + \lambda \, \Delta(x; y, y^{\mathrm{ref}}). \tag{14}$$

Why no penalty for failed correction. We do not impose an additional negative penalty for the case where $r(x, y) = 0$ and $r(x, \hat{y}) = 0$, in order to avoid mistakenly attributing insufficient correction ability or inadequate reference quality to the intrinsic quality of the original reasoning. This design choice helps prevent unnecessary bias and training instability.

C.3. Trajectory Diversity and Coupling Reduction

IBPO relies on sampling multiple trajectories under the same input with sufficient diversity. In our experiments, we adopt the following strategies to increase trajectory diversity and reduce trajectory coupling:

- **Stochastic decoding.** We use different random seeds, temperatures, and top-p/top-k sampling parameters.
- **Prompt perturbation (optional).** We apply mild perturbations to system or formatting prompts to induce trajectory-level differences.

D. Implementation Details

Infrastructure. Experiments are conducted on 32 Nvidia A800 (80G) GPUs.

Optimization. We use an initial learning rate of 5×10^{-7} with cosine decay (min ratio 0.1) and a linear warmup of 3% total steps. Entropy regularization coefficient is set to 0.

Sampling. We use $G = 64$ rollouts per prompt for GSPO and $G = 8$ for IBPO to approximately match the overall compute budget across methods.

E. An Instantiation of IBPO: Integration with GSPO

GSPO is used solely as a carrier optimizer; replacing GSPO with GRPO or PPO does not change the formulation of IBPO. In the previous sections, we have presented the general formulation of IBPO and its reward shaping definition. IBPO is a training formulation that is orthogonal to the underlying sequence-level reinforcement learning method. In this section, we describe how this formulation can be seamlessly integrated into a representative sequence-level reinforcement learning algorithm, namely GSPO.

E.1. Preliminaries: Sequence-Level RL

We view an autoregressive language model parameterized by θ as a policy π_θ. Let $\mathcal{D}$ denote the set of queries. Given a query x, the model generates a complete response $y = (y_1, \ldots, y_{|y|})$, with sequence probability

$$\pi_\theta(y \mid x) = \prod_{t=1}^{|y|} \pi_\theta(y_t \mid x, y_{<t}). \tag{15}$$

We consider a general class of sequence-level policy optimization objectives:

$$J(\theta) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_{\theta_{\text{old}}}(\cdot|x)}\Big[\mathcal{L}\big(s(\theta; x, y), \hat{A}(x, y)\big)\Big], \tag{16}$$

where $s(\theta; x, y)$ denotes the sequence-level importance weight, and $\hat{A}(x, y)$ is constructed from sequence-level rewards. Methods such as GSPO and GRPO can be viewed as specific instantiations of this formulation.

IBPO does not alter the optimization form in Eq. (16), but instead reshapes the original sequence-level rewards through the model's self-correction process.

E.2. IBPO with GSPO: One-Pass Joint Training

At each iteration, we perform a **one-pass** policy update: for the same batch of queries x, we simultaneously construct the candidate response set used for sequence-level RL and the self-correction outcomes used to evaluate recoverability. IBPO modifies only the reward definition (by shaping it into r'), while leaving the surrogate objective of GSPO unchanged.

(A) GSPO backbone with shaped rewards. For each query x, we sample G responses $\{y_i\}_{i=1}^G$ from the old policy. The sequence-level importance ratio in GSPO is defined as

$$s_i(\theta) = \left(\frac{\pi_\theta(y_i \mid x)}{\pi_{\theta_{\text{old}}}(y_i \mid x)}\right)^{\frac{1}{|y_i|}}. \tag{17}$$

We construct group-wise advantages using the **shaped rewards**:

$$\hat{A}_i = \frac{r'(x, y_i) - \mu'}{\sigma'}, \tag{18}$$

where μ' and σ' denote the mean and standard deviation of the shaped rewards within the group. The GSPO optimization objective is given by

$$\begin{aligned} J_{\text{GSPO}}(\theta) = \mathbb{E}_{x\sim\mathcal{D},\, \{y_i\}_{i=1}^G\sim\pi_{\theta_{\text{old}}}(\cdot|x)}\Bigg[\frac{1}{G}\sum_{i=1}^{G}\min\Big(s_i(\theta)\,\hat{A}_i, \\ \text{clip}\big(s_i(\theta),\, 1-\epsilon,\, 1+\epsilon\big)\,\hat{A}_i\Big)\Bigg]. \end{aligned} \tag{19}$$

(B) Self-correction shaping signal. To compute the shaped reward $r'(x, y_i)$, we construct a self-correction instance for each incorrect response y_i. Specifically, we randomly sample a reference response y_i^{ref} (if there exists at least one correct response in the group, we sample from the correct responses; otherwise, from the incorrect ones), and ask the model to compare and correct:

$$\hat{y}_i = \mathcal{C}(x; y_i, y_i^{\text{ref}}). \tag{20}$$

This process introduces no additional supervision and is used solely to evaluate whether the model can correct an incorrect response into a correct one. Based on an instantiation of Eq. (13), we define

$$\Delta_i = \beta \cdot \mathbb{I}[r(x, y_i) < 1 \ \wedge \ r(x, \hat{y}_i) = 1], \qquad \beta = 0.5, \tag{21}$$

and obtain the shaped sequence-level reward:

$$r'(x, y_i) = r(x, y_i) + \lambda \, \Delta_i. \tag{22}$$

Although $r'(x, y_i)$ remains a sequence-level scalar in form, its value depends on the counterfactual self-correction process, thereby implicitly encoding process-level (step-level) credit information.

(C) Joint GSPO training for correction behavior. In addition to using the implicit process reward $r'(x, y)$ for sequence-level optimization on the original reasoning input x, we further treat the **correction behavior itself as a reasoning task of the same policy over an expanded input space**, and jointly train it using the **same GSPO objective**. Formally, this procedure does not introduce a new Markov Decision Process (MDP), but simply applies the policy to different conditional inputs (i.e., a prompt-conditioned policy).

Specifically, for each response y_i whose recoverability needs to be evaluated, we construct a correction input:

$$\tilde{x}_i = (x; \ y_i, \ y_i^{\text{ref}}), \tag{23}$$

where y_i^{ref} denotes the reference response. Conditioned on this input, the model generates a corrected output $\hat{y}_i \sim \pi_\theta(\cdot \mid \tilde{x}_i)$ and receives a terminal reward $r(\tilde{x}_i, \hat{y}_i) \in \{0, 1\}$ based on the correctness of the final answer.

We incorporate these correction samples together with the original reasoning samples into the sequence-level optimization of GSPO. Formally, the GSPO objective can be written as a unified expectation over a **mixture input distribution**:

$$J_{\text{GSPO}}^{\text{joint}}(\theta) = \mathbb{E}_{\tilde{x} \sim \mathcal{D}_{\text{mix}}} \left[\frac{1}{G} \sum_{i=1}^{G} \min\Big(s_i(\theta) \, \hat{A}_i, \ \text{clip}\big(s_i(\theta), \, 1 - \epsilon, \, 1 + \epsilon\big) \, \hat{A}_i \Big) \right], \tag{24}$$

where $\mathcal{D}_{\text{mix}}$ denotes the mixture distribution composed of the original query inputs x and the correction inputs $\tilde{x} = (x; y_i, y_i^{\text{ref}})$. The corresponding sequence-level rewards are defined as $r'(x, y)$ for the original inputs and $r(\tilde{x}, \hat{y})$ for the correction inputs, respectively, while both share the same GSPO surrogate form.

We emphasize that this joint training procedure **does not introduce additional optimization stages or different loss functions**. Learning correction capability is manifested purely as behavioral generalization of the policy under different input conditions, allowing the model to gradually internalize comparison and correction abilities during training, without requiring additional multi-round calls at inference time.

E.3. IBPO + GSPO Algorithm Pseudocode

Combining all the stages described above, the complete workflow of IBPO + GSPO is summarized in Algorithm 1.

Algorithm 1 IBPO Instantiation with GSPO: One-Pass Joint Training

Require: Dataset $\mathcal{D}$; current policy π_θ; old policy $\pi_{\theta_{\text{old}}}$; group size G; clip ϵ; shaping weight λ; recoverability scale β; correction operator $\mathcal{C}$; terminal reward $r(\cdot) \in \{0, 1\}$.
Ensure: Updated parameters θ.

for each iteration **do**
  Sample a minibatch of prompts $\mathcal{B} = \{x\}$ from $\mathcal{D}$.
  for each prompt $x \in \mathcal{B}$ **do**
    (A) Sample rollouts and compute GSPO ratios.
    Sample G responses $\{y_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(\cdot \mid x)$.
    for $i = 1$ to G **do**
      Compute terminal reward $r_i \leftarrow r(x, y_i)$.
      Compute sequence-level ratio
        $s_i(\theta) \leftarrow \left(\frac{\pi_\theta(y_i|x)}{\pi_{\theta_{\text{old}}}(y_i|x)}\right)^{\frac{1}{|y_i|}}$.
    end for
    (B) Self-correction shaping signal and shaped rewards.
    for $i = 1$ to G **do**
      if $r_i = 0$ **then**
        Sample reference response y_i^{ref}:
        if there exists j such that $r(x, y_j) = 1$ **then**
          Sample y_i^{ref} uniformly from $\{y_j : r(x, y_j) = 1\}$.
        else
          Sample y_i^{ref} uniformly from $\{y_j : r(x, y_j) = 0\}$.
        end if
        Generate correction $\hat{y}_i \leftarrow \mathcal{C}(x; y_i, y_i^{\text{ref}})$.
        Set $\Delta_i \leftarrow \beta \cdot \mathbb{I}[r(x, y_i) = 0 \wedge r(x, \hat{y}_i) = 1]$.
      else
        Set $\Delta_i \leftarrow 0$.
      end if
      Shaped reward $r'_i \leftarrow r_i + \lambda\, \Delta_i$.
    end for
    (A continued) Construct group advantages from shaped rewards.
    $\mu' \leftarrow \frac{1}{G} \sum_{i=1}^{G} r'_i$.
    $\sigma' \leftarrow \sqrt{\frac{1}{G} \sum_{i=1}^{G} (r'_i - \mu')^2}$.
    for $i = 1$ to G **do**
      $\hat{A}_i \leftarrow \frac{r'_i - \mu'}{\sigma' + 10^{-8}}$.
    end for
    GSPO surrogate objective on original prompts.
    Accumulate
      $\mathcal{J}_x(\theta) \leftarrow \frac{1}{G} \sum_{i=1}^{G} \min\Big(s_i(\theta)\hat{A}_i,\ \text{clip}(s_i(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_i\Big)$.
  end for
  (C) Optional: Joint GSPO training for correction behavior.
  Construct correction inputs $\tilde{x}_i \leftarrow (x;\, y_i,\, y_i^{\text{ref}})$ for corrected cases.
  Sample $\hat{y}_i \sim \pi_{\theta_{\text{old}}}(\cdot \mid \tilde{x}_i)$ and compute terminal reward $r(\tilde{x}_i, \hat{y}_i)$.
  Add the same GSPO surrogate on $\tilde{x}_i$ (Eq. 24) if joint training is enabled.
  Update θ by maximizing $\sum_{x \in \mathcal{B}} \mathcal{J}_x(\theta)$ (and joint objective if enabled).
end for