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]

Abstract

Reinforcement learning for multi-step reasoning with large language models (LLMs) typically relies on sparse terminal rewards, leading to poor credit assignment—the final feedback is uniformly propagated to all intermediate decision steps. This produces high gradient variance, training instability, and numerous ineffective updates, ultimately preventing the model from achieving sustained improvement. We propose a counterfactual comparison-based credit assignment framework that samples multiple reasoning trajectories from the same input, treats inter-trajectory differences as implicit approximations of alternative decisions, and thereby constructs an implicit process-level advantage estimator that converts sparse terminal rewards into step-sensitive learning signals. Building on this, we introduce Implicit Behavior Policy Optimization (IBPO), which significantly improves training stability and performance ceilings on mathematical and code reasoning benchmarks, pointing toward a promising direction for unlocking the performance potential of LLMs.

1. Introduction

Recent advances in large language models (LLMs) have achieved remarkable breakthroughs on complex multi-step reasoning tasks, particularly after fine-tuning with reinforcement learning (RL). RL has become a key paradigm for scaling LLM capabilities, enabling models to solve increasingly complex problems through deeper and longer chains of reasoning, such as competition-level mathematics and program synthesis.

However, scaling RL for reasoning tasks requires maintaining training stability and sample efficiency under ever-increasing compute budgets. Despite this, mainstream RL methods—such as Group Relative Policy Optimization (GRPO) (Shao et al., 2024)—still use sequence-level or trajectory-level rewards to optimize policies. This creates a fundamental mismatch between the learning signal and the inherently step-by-step nature of reasoning.

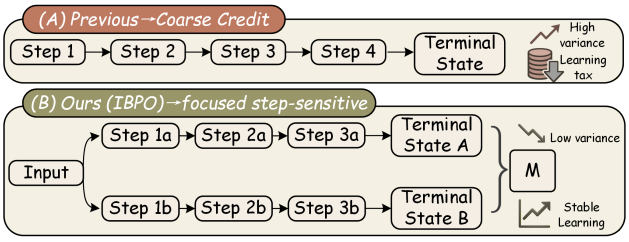

Figure 1. Overview of IBPO: a counterfactual trajectory comparison framework for process-level credit assignment under sparse terminal rewards. By contrasting multiple reasoning paths sampled from the same input, IBPO derives implicit step-sensitive learning signals, improving optimization stability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

In multi-step reasoning, correctness depends on a sequence of intermediate decisions. Yet sequence-level supervision rewards entire trajectories based solely on the final answer: trajectories with flawed reasoning processes can receive positive rewards if they happen to produce the correct final output, while trajectories with largely correct reasoning but a single local error may be entirely discarded. This coarse-grained feedback undermines the model's ability to distinguish early from late errors, disrupts credit assignment, destabilizes learning, and limits exploration of alternative reasoning paths. This problem is particularly pronounced in long-horizon or difficult tasks. Moreover, even a single local error may require extensive sampling and updates to be statistically corrected, introducing a significant efficiency bottleneck—commonly referred to as the *learning tax*.

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

[1]Alibaba Group [2]Tsinghua University. Correspondence to: Fei Ding <dingfei@mails.tsinghua.edu.cn>.

Preprint. April 26, 2026.

Building on this insight, we introduce *Implicit Behavior Policy Optimization (IBPO)*—a process-level credit assignment framework induced by counterfactual trajectory comparison. IBPO defines a general multi-trajectory comparison operator and uses it to construct an implicit advantage estimator. This estimator reweights terminal rewards based on trajectory-level differences, thereby reducing gradient variance and amplifying learning signals at points of frequent decision errors. IBPO does not rely on step-level annotations, 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 formulation.** We introduce a counterfactual learning perspective on credit assignment in LLM reinforcement learning, treating multiple reasoning trajectories from the same input as approximations of alternative decisions. We show that the inconsistencies among these trajectories contain key information for process-level learning, even without step-level rewards.
- **Implicit process-level advantage and the IBPO framework.** We formalize a general multi-trajectory comparison operator and use it to construct an implicit process-level advantage estimator, from which we derive the IBPO framework.
- **Theoretical analysis of variance reduction and positive transfer.** We analyze how counterfactual trajectory comparison reduces gradient variance and amplifies learning signals in high-error-rate regions. We prove that this mechanism induces backward transfer to underlying reasoning skills and mitigates the learning tax problem.
- **Mechanism-driven empirical validation.** We evaluate IBPO on multiple mathematical and code reasoning benchmarks. Experiments demonstrate that IBPO consistently improves convergence, sample efficiency, and early error correction ability over strong baselines under compute-matched conditions.

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 within-group sampling to estimate group-relative advantages, replacing explicit value modeling in PPO, thereby enabling faster and more efficient training. However, GRPO suffers from entropy collapse, reward collapse, and unstable convergence (Yu et al., 2025), largely stemming from its reliance on the assumption that *the terminal reward sufficiently characterizes the reasoning trajectory*. This assumption often fails in long-horizon reasoning—where the model's success depends 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 computes importance ratios at the sequence level.

Self-Correction Strategies. Self-correction has emerged as a promising direction for enhancing reasoning capabilities. For example, selective reflection fine-tuning (Li et al., 2024) enables models to perform reflective evaluation over multiple candidate responses and fine-tune on the optimal response through supervised learning.

Reward Modeling. Reward models are crucial for achieving robust System-2 reasoning but are 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; Luo et al., 2024; Wang et al., 2024b) that provide step-level feedback for complex tasks. However, PRMs have critical limitations: high annotation costs, weak generalization, and noisy signals produced by 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 difficult to scale, and existing automatic annotation methods typically produce noisy or inconsistent reward scores. In contrast, our IBPO approach bypasses the need for fine-grained annotation through implicit comparison while still providing effective process-level supervision. Unlike existing methods, our approach does not assume that rewards can be decomposed into stepwise reward signals.

SCoRe (Kumar et al., 2024) iteratively leverages previously generated responses, prompting the model to identify errors in earlier outputs. It improves reasoning accuracy through multi-round reinforcement learning, but suffers from low training efficiency due to repeated generation and optimization cycles.

3. Method

3.1. Problem Formulation

We consider multi-step reasoning reinforcement learning problems with terminal rewards. Given input x, the 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 a **sequence-level reward** only 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 does not provide 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 **extreme instability of credit assignment from terminal rewards to early decisions**. When local errors occur at early steps, their effects tend to cascade and amplify through subsequent reasoning steps. However, such errors are only indirectly reflected through the terminal reward, resulting in extremely noisy gradient signals whose variance grows significantly with trajectory length.

IBPO as a Framework Rather Than an Implementation. We emphasize that IBPO is a *training framework* for credit assignment under sparse terminal rewards, rather than a specific error-correction or rewriting algorithm. Its core contribution is establishing 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 primarily used to extract differences between trajectories and generate learning signals reflecting process-level decision differences. The IBPO framework does not depend on specific implementation details, such as how counterfactual differences are computed or what comparison mechanism is used. The operator $\mathcal{M}$ can be instantiated through various means—such as consistency scoring, relative ranking, or error detection—but these are implementation details rather than components of the IBPO framework itself. Therefore, IBPO's core contribution lies in its framework-level design, while specific instantiations can be customized according to task requirements.

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

3.2. Counterfactual Trajectory Comparison

Counterfactual Trajectory Comparison and the Role of Operator $\mathcal{M}$. The core idea of IBPO is to sample multiple reasoning trajectories from the same input and leverage their differences as counterfactual approximations to construct process-sensitive learning signals. Specifically, we sample G trajectories from the policy. Completely correct trajectories do not require additional signals; for each incorrect trajectory τ_i, we pair it with $K-1$ correct trajectories to form a K-tuple. When correct trajectories are insufficient, we duplicate them to reach the required number; if no correct trajectories exist, incorrect trajectories are randomly selected as substitutes.

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

We introduce the **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 the comparison-induced signal associated with trajectory $\tau_i^{(k)}$, summarizing its intermediate decision quality relative to other counterfactual trajectories (e.g., relative consistency, recoverability, or difference-aware quality). The operator $\mathcal{M}$ can be implemented through various mechanisms and is subsequently validated through rule-based rewards to avoid circular reasoning and potential reward hacking or self-confirmation bias. The IBPO framework only assumes that $\mathbf{s}(\tau_i)$ is sensitive to counterfactual differences in $\{\tau_i^{(k)}\}$.

The output of $\mathcal{M}$ is not limited to sequence-level scalar signals; $\mathcal{M}$ can also produce token-level signals, such as the proportion of unmodified tokens for reward shaping (IBPO-ratio), or a token-level mask for blocking gradients on unmodified tokens (IBPO-mask). Detailed definitions of these variants are provided in Appendix C.2.

Per-Trajectory Shaping Function. Given the comparison output $\mathbf{s}(\tau_i) = \mathcal{M}(\{\tau_i^{(k)}\}_{k=1}^{K})$, we define the 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 through rule-based rewards to avoid circular reasoning and potential reward hacking 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 the relationship between τ_i and other counterfactual trajectories, requiring no explicit step-level annotation or value estimation.

Token-Level Masking. When the operator $\mathcal{M}$ can localize which tokens in the trajectory should receive gradient updates and which should not, the comparison signal can be refined from the sequence level to a token-level mask. Let $\mathbf{m}_i = (m_1, \ldots, m_T) \in \{0,1\}^T$ be the token-level mask produced by $\mathcal{M}$, where $m_t = 1$ indicates that the t-th token should receive gradient updates and $m_t = 0$ indicates that the token should not be updated. Based on this, the policy gradient can selectively update only the designated tokens:

$$\nabla_\theta \mathcal{J}_i^{\text{mask}} = \sum_{t=1}^{T} m_t \cdot \widehat{A'}_i \cdot \nabla_\theta \log \pi_\theta(y_t \mid y_{<t}, x). \quad (7)$$

This masking mechanism concentrates gradient updates on potentially erroneous tokens, avoiding unnecessary penalization of correct reasoning steps, thereby achieving more fine-grained token-level credit assignment. The specific construction of the mask (e.g., based on edit distance) is detailed in Appendix C.2.

3.3. Implicit Process-Level Advantage Estimation

To inject comparison signals into the optimization process, we provide two complementary paths to replace the coarse-grained feedback determined solely by $R(\tau)$.

Path One: Sequence-Level Reward Shaping. For candidate trajectory $\tau_i^{(k)}$, we define its shaped reward as:

$$R'_i(x) = R(\tau_i) + \lambda\, \phi_i \quad (8)$$

where $0 \le \lambda\, \phi_i < 1$, 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 $\lambda\, \phi_i$ through within-group advantage normalization:

$$\widehat{A'}_i = \frac{R'_i(x) - \text{mean}\left(\{R'_i(x)\}_{i=1}^G\right)}{\text{std}\left(\{R'_i(x)\}_{i=1}^G\right)}. \quad (9)$$

Path Two: Token-Level Gradient Masking. When $\mathcal{M}$ produces a token-level mask $\mathbf{m}_i$ (see previous section), the policy gradient can perform selective updates at the token granularity (Equation (7)). This path combines sequence-level advantage $\widehat{A'}_i$ with token-level masking: the advantage is still determined at the sequence level by counterfactual comparison, but gradients flow only through tokens marked for update. Neither path requires explicit step-level annotation or a value model.

3.4. Mechanism: Positive Backward Transfer in Multi-Task Learning

Counterfactual trajectory comparison does not directly provide the model with explicit error labels. Instead, by contrasting 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, from a multi-task learning perspective, the auxiliary comparison task induces *positive backward transfer* to the original reasoning task: the model learns to suppress local errors more quickly, thereby accelerating convergence on the base task. This mechanism reduces the number of ineffective updates required to correct local errors and mitigates the *learning tax* commonly observed in long-horizon reinforcement learning.

Testable Predictions. This mechanism specifically predicts significant improvements on difficult reasoning tasks, particularly in scenarios where correct trajectories are extremely scarce and training signals are 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—GSPO exhibits significantly greater fluctuation on the more difficult LiveCodeBench tasks, while IBPO demonstrates noticeably smoother training curves.

Proposition 3.1 (Effectiveness of IBPO's Process-Level Advantage Estimation)**.** *For any multi-step reasoning task, assume the policy π_θ generates G trajectories $\{\tau_i\}_{i=1}^G$ given input x, where each trajectory τ_i corresponds to a sequence-level reward $R(\tau_i)$. For each incorrect trajectory τ_i, comparison signals $\mathbf{s}(\tau_i)$ are generated through counterfactual comparison with $K-1$ correct trajectories, and injected into the optimization process through two paths: (i) mapping to shaped rewards $R'_i(x)$; (ii) when $\mathcal{M}$ produces token-level masks $\mathbf{m}_i$, selectively updating only the marked tokens.*

Under the assumption that the shaping function ϕ_i provides informative stepwise signals for suboptimal trajectories, the variance of the policy gradient estimator for any trajectory τ_i is significantly reduced compared to the baseline based on episode-level rewards. Specifically, in Path One, ϕ_i suppresses gradient noise through reward shaping; in Path Two, token-level masking further concentrates gradient updates on potentially erroneous tokens, avoiding unnecessary penalization of correct reasoning steps. The two paths synergistically enhance training stability and accelerate convergence.

Proof. By introducing the multi-trajectory comparison operator $\mathcal{M}$, we obtain contrastive signals $\mathbf{s}_i(x)$ from G trajectories, which are sensitive to counterfactual differences between trajectories. These differences reflect the impact of different decisions during reasoning and are injected

into optimization through two paths: (i) ϕ_i is mapped to shaped rewards $R'_i(x)$, achieving finer-grained sequence-level credit assignment; (ii) token-level masks $\mathbf{m}_i$ restrict gradient updates to modified tokens (Equation (7)), achieving selective credit assignment at the token granularity. Both paths effectively reduce gradient variance caused by early decision errors, thereby avoiding typical training instability.

Specifically, ϕ_i provides process-level feedback for incorrect trajectories rather than relying solely on terminal rewards; $\mathbf{m}_i$ further filters out gradient contributions from correct tokens, making the update signal more precise. Detailed mathematical proofs are provided in Appendix A. □

4. Experiments

Instantiation of IBPO. As discussed above, IBPO provides a framework for reward and advantage construction 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 existing sequence-level reinforcement learning methods.

In this work, we instantiate IBPO on top of GSPO. IBPO focuses on constructing process-sensitive advantage estimates through counterfactual multi-trajectory comparison under terminal reward conditions, while GSPO purely serves as a sequence-level optimizer to carry and apply these advantage signals. As shown in Appendix E, we provide the detailed framework of IBPO+GSPO. We also verified similar trends on GRPO; results are omitted for brevity. The specific instantiation of the comparison operator $\mathcal{M}$ is detailed 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 Models.** 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 performed using the VLLM engine (version 0.11.2).

Baselines and Comparison Setup. We compare IBPO instantiated on GSPO (Zheng et al., 2025) (denoted as **IBPO+GSPO**) against the following baselines: (1) vanilla GSPO; and (2) GSPO with prompt-based error correction, which introduces additional correction prompts at inference time after GSPO training to generate revised outputs.

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

Compute-Matched Protocol. See Appendix B for details.

Complete Rewrite Filtering. To prevent the model from completely rewriting rather than locally fixing during correction (leading to reward hijacking), we detect complete rewrites via edit distance and set their shaped reward to zero. This mechanism serves as a defensive safeguard rather than a core component; see Appendix C.3 for details.

Although the experiments in this paper primarily focus on mathematical and code reasoning tasks, the framework design and applicability of IBPO are not limited to specific task domains. Its core mechanism is not multi-draft generation or explicit error correction per se, but rather constructing multiple counterfactual reasoning trajectories under the same input and leveraging the differences between these trajectories in 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 applicability depends solely on whether some objective or verifiable feedback signal exists during training, rather than relying on specific task formats or output structures.

In mathematical and programming tasks, the correctness of the final solution has a clear and automatically verifiable definition, making these tasks convenient and reliable testbeds 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, structured knowledge reasoning, or multi-step decision problems with well-defined termination conditions—appropriate verifiable reward functions or evaluation criteria can similarly be designed 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 differences 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 lead to failure.

From a structural perspective, the key advantage of IBPO lies in its modeling of counterfactual trajectory indepen-

dence and the resulting implicit process-level credit assignment mechanism. This mechanism does not rely on explicit step-level annotations or additional value models; rather, under the condition of only terminal rewards, it introduces more discriminative learning signals for the reasoning process through multi-trajectory counterfactual comparison. Therefore, although we have not yet provided empirical results on non-mathematical or non-code tasks, the counterfactual trajectory comparison principle and 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 1 reports performance comparisons across multiple reasoning benchmarks. For the Qwen3-32B model, inference parameters are set to temperature= 0.6, TopP= 0.95, TopK= 20, MinP= 0; for the Qwen3-Next-80B-A3B-Instruct model, inference parameters are set to temperature= 0.7, TopP= 0.8, TopK= 20, MinP= 0. Compared methods include IBPO, GSPO, and GSPO augmented with prompt-based error correction.

In addition to the base IBPO (sequence-level reward shaping), we also evaluate two variants based on token-level edit distance for counterfactual analysis: **IBPO(ratio)** uses the proportion of unmodified tokens between the original response and the corrected output as a shaping coefficient—a higher unmodified proportion indicates the original reasoning is closer to correct, resulting in a larger shaped reward; **IBPO(mask)** uses edit distance to locate modified tokens, masks the gradient contribution of unmodified tokens, and applies policy gradient updates only to modified tokens. Detailed definitions of both variants are provided in Appendix C.2.

5.1. Comparison with GSPO.

To maintain approximately matched computational cost, GSPO samples 64 responses per input prompt, while IBPO samples 8 responses per prompt.

As shown in Figure 2, we plot the training reward and evaluation performance curves as compute increases.

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

5.1.1. Training Efficiency and Stability under Compute-Matched Conditions

Figure 2 dynamically compares GSPO and GSPO+IBPO under **compute-matched** conditions from two dimensions: training reward and external evaluation performance

Model	Benchmark	Method	Acc (%)
Qwen3-32B	AIME25	GSPO + IBPO	85.3±1.2
		GSPO + IBPO(ratio)	85.0±1.3
		GSPO + IBPO(mask)	**85.9±1.1**
		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 + IBPO(ratio)	74.9±1.2
		GSPO + IBPO(mask)	**76.0±1.0**
		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 + IBPO(ratio)	62.2±1.5
		GSPO + IBPO(mask)	**63.1±1.3**
		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 + IBPO(ratio)	93.4±1.4
		GSPO + IBPO(mask)	**94.4±1.3**
		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 + IBPO(ratio)	74.7±1.4
		GSPO + IBPO(mask)	**75.8±1.2**
		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 + IBPO(ratio)	80.0±1.5
		GSPO + IBPO(mask)	**81.1±1.4**
		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 1. 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 95% bootstrap confidence interval (mean ± 95% CI) over 5 random seeds; improvements over baseline methods are statistically significant under paired bootstrap tests (p ¡ 0.01). Total training FLOPs are matched across all methods, including generation and comparison overhead. The Best-of-N method uses N = 8.

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

Table 2. Based on Qwen3-Next-80B-A3B-Thinking, we measure the compute required to reach a fixed training reward threshold under the **compute-matched** setting. Results are normalized relative to GSPO.

(AIME25 and LiveCodeBench). In this experiment, we align the training process by matching the overall compute consumption of different methods, ensuring that at any point on the horizontal axis, the total computational resources consumed by both methods—including model forward passes, backward passes, and generation overhead—are statistically equivalent. Therefore, the horizontal axis is labeled *Training Compute*, which can be understood as a direct measure of the actual training compute budget.

Higher Compute Efficiency under the Same Budget. Under a strictly matched compute budget, GSPO+IBPO consistently achieves higher training rewards throughout the training process and enters the high-reward regime earlier under the same compute constraint. In comparison, GSPO shows notably slower reward improvement under equivalent compute. These results indicate that IBPO can convert terminal rewards into more effective parameter updates *per unit of compute*, thereby substantially improving training compute efficiency. In other words, under the same compute investment, GSPO+IBPO produces more *effective learning*, significantly enhancing overall training efficiency.

More Stable Optimization Dynamics. Beyond improvements in average performance, under compute-matched conditions, the training reward curve of GSPO+IBPO exhibits a notably smoother evolution, with significantly smaller fluctuations compared to GSPO. When using only sequence-level terminal rewards, local reasoning errors tend to be propagated backward uniformly through the shared terminal feedback across the entire generated sequence, causing gradient estimates to be dominated by noise from irrelevant time steps, thereby exacerbating training instability. By introducing shaped signals based on counterfactual trajectory comparison, IBPO enables the model to more rapidly identify error-prone regions and suppress the impact of ineffective updates on the optimization process. Gradient variance measurements in the appendix further confirm this: IBPO reduces policy gradient variance by approximately 30% on average, which directly corresponds to the observed improvement in training stability, thereby substantially alleviating the *learning tax* in reinforcement learning.

Evolution of Correction Success Rate. The informativeness of the shaped signal depends on whether the correction success rate falls within a meaningful intermediate range—if correction almost always succeeds or always fails, the signal degenerates to a constant. We tracked the evolution of correction success rate during training of Qwen3-32B (AIME25): at the beginning of training, the correction success rate is approximately 12%, gradually rising to approximately 67% as training progresses. This indicates that the shaped signal remains in an information-rich regime throughout the training process, being neither constantly zero nor constantly one, thereby continuously providing discriminative process-level feedback for policy optimization.

5.1.2. Positive Backward Transfer

As shown in Figure 2, we observe that introducing IBPO leads to faster performance improvement and faster convergence. We attribute this to a *positive backward transfer* effect. In multi-task learning, positive backward transfer refers to the phenomenon where learning a subsequent task (Task B) improves performance on a previous 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 main reasoning task during training.

Specifically, under GSPO training, the model receives only the sequence-level reward $R(y)$, which propagates uniformly across the entire reasoning sequence. When the final failure is caused by only a few local tokens, this supervisory signal cannot indicate where the error occurred, forcing the model to rely on extensive sampling and iterative updates to statistically internalize these local errors over time. This substantially increases the sample complexity of the learning process, giving rise to the well-known *learning tax* in reinforcement learning.

IBPO introduces a novel auxiliary task based on counterfactual reasoning trajectory comparison. By contrasting inconsistencies across different reasoning paths, potential local errors become structurally more salient, thereby guiding the learning process. This mechanism does not provide explicit token-level annotations; rather, it enhances the *observability* of errors through counterfactual contrast, accelerating the model's internalization of fine-grained reasoning errors. From the experimental results, 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, with the vast majority of model-generated trajectories being incorrect. This distribution

leads to highly sparse sequence-level rewards, causing policy gradient estimates to be dominated by negative samples, which slows convergence and may even lead to training instability. To address this issue, some GRPO variants use sample truncation to artificially balance the ratio of correct to incorrect responses. However, this count-based truncation strategy alters the original sampling distribution, thereby violating the consistency of importance sampling and potentially 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 converts a small number of correct reasoning paths into multiple information-rich learning signals through counterfactual trajectory comparison, substantially alleviating the learning bottleneck caused by the scarcity of positive samples in difficult tasks.

5.2. Comparison with GSPO with Prompt

The core difference between GSPO+Prompt and IBPO+GSPO lies in whether joint training is performed. The former conducts multi-trajectory comparison and correction only at inference time through prompts after training is complete, while the latter performs multi-trajectory comparison and jointly optimizes the model during training. 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 the model's robustness across problems of varying difficulty. The consistent performance improvements 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 3. Ablation results of Qwen3-32B on AIME25, LiveCodeBench, and HMMT25. Each variant removes one key component from the full IBPO algorithm.

To evaluate the contribution of each component in IBPO, we conduct ablation experiments based on the Qwen3-32B model on the AIME25, LiveCodeBench, and HMMT25 datasets. Table 3 summarizes the corresponding results.

GSPO + Test-Time Prompt. Multi-trajectory comparison is performed only at inference time through prompts. Due to the absence of joint training, the gains introduced by IBPO vanish, leading to significant performance degradation. This result validates the *positive transfer* effect induced by IBPO and the effectiveness of implicit process-level rewards.

IBPO ($k = 1$). Only a single reasoning trajectory is used. Without multi-trajectory counterfactual comparison, accuracy drops substantially. This indicates that inconsistencies across multiple counterfactual trajectories play a critical role in error identification. This setting is similar to GSPO + SCoRe, which adopts a two-stage reinforcement learning approach and introduces tree-structured rewards, thereby increasing the learning burden.

IBPO (Shaping Only). In this ablation, we disable joint training with multi-trajectory comparison but retain the reward shaping term. This demonstrates the positive transfer effect brought 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 4. Evaluation results under different λ values. (IBPO score; 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 overall performance.

6. Conclusion

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

Experiments demonstrate that when combined with sequence-level optimizers such as GSPO, IBPO consistently improves performance and training stability on mathematical and code reasoning benchmarks under compute-matched settings. Its formulation is agnostic to the underlying RL algorithm and directly compatible with GRPO variants and other policy gradient methods—providing a scalable path toward more robust multi-step reasoning in large language models.

Limitations

IBPO incurs additional computational overhead due to sampling multiple counterfactual trajectories per input. Although our results demonstrate higher sample efficiency under a fixed budget, reducing this overhead remains important for large-scale deployment. Furthermore, if counterfactual trajectories exhibit systematic errors, the comparison signal may weaken. Enhancing trajectory diversity or incorporating external verifiers can mitigate this issue.

Ethics Statement

This work does not present any known ethical risks within its current scope.

Reproducibility Statement

We will release code, model checkpoints, and detailed experimental configurations through 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.

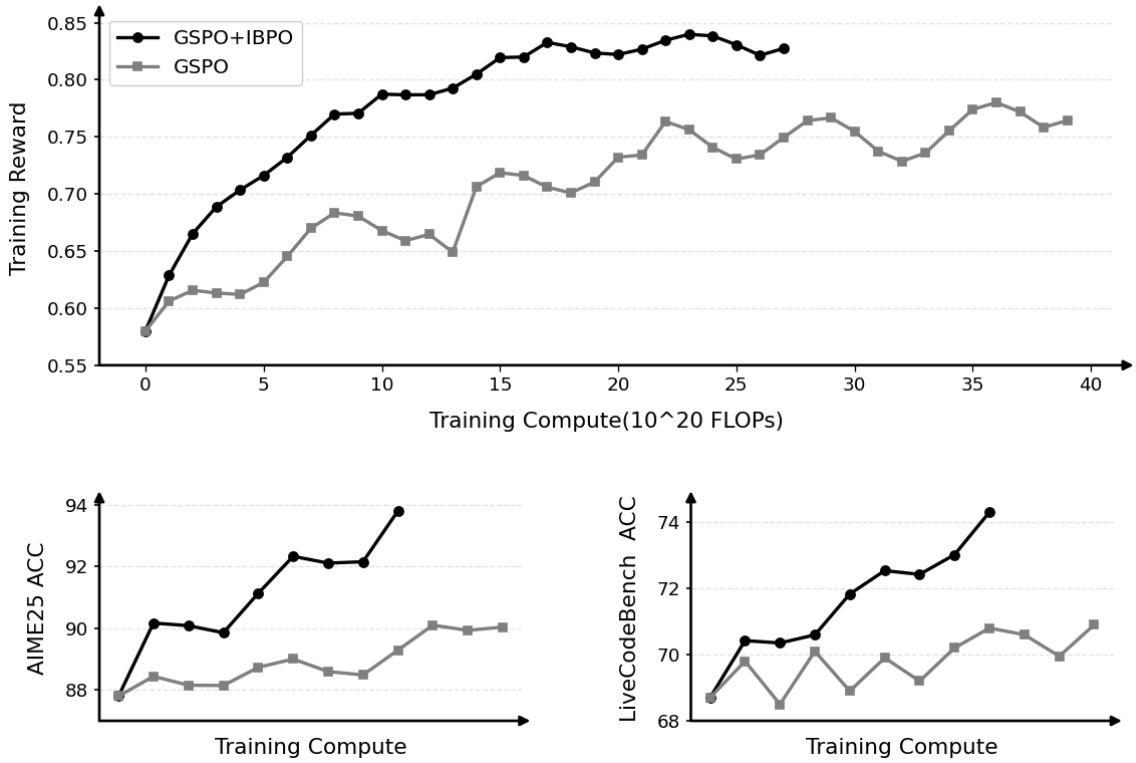

Figure 2. Training curves based on fine-tuning Qwen3-Next-80B-A3B-Thinking indicate that IBPO achieves significantly higher training efficiency compared to GSPO.

A. Theoretical Analysis: Variance Reduction Properties of IBPO

To formally characterize the advantage of IBPO in credit assignment, we use the representative GSPO-class method as a baseline and prove under reasonable assumptions that the implicit process-level advantage estimator constructed by IBPO has lower variance than the pure terminal reward advantage used in GSPO, thereby leading to more stable policy gradient updates.

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

- $Y_i = R(\tau_i) \in \{-1, 1\}$ denote the terminal reward of the i-th trajectory (assumed to be binary for analytical convenience);
- $\phi_i \in [0, 1]$ 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 that ϕ_i effectively reflects the "recoverability" or "consistency with correct reasoning" of a trajectory—i.e., the closer an incorrect trajectory is to the correct reasoning process, the larger ϕ_i is.

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

$$A_i^{\text{GSPO}} = Y_i - \bar{Y}, \quad \text{where } \bar{Y} = \frac{1}{G}\sum_{j=1}^{G} Y_j, \tag{10}$$

$$A_i^{\text{IBPO}} = (Y_i + \lambda\phi_i) - (\bar{Y} + \lambda\bar{\phi}) = A_i^{\text{GSPO}} + \lambda(\phi_i - \bar{\phi}), \tag{11}$$

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 the 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 $\phi_i > 0$, and a larger ϕ_i indicates closer proximity to correct reasoning.

Theorem A.2 (Variance Reduction of IBPO Relative to GSPO)**.** *Under Assumption A.1 and 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), then the variance of the IBPO policy gradient estimator is strictly less 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:

$$\begin{aligned}\mathrm{Var}(A_i^{\mathrm{IBPO}}) &= \mathrm{Var}\big(A_i^{\mathrm{GSPO}} + \lambda(\phi_i - \bar{\phi})\big) \\ &= \mathrm{Var}(A_i^{\mathrm{GSPO}}) + \lambda^2\mathrm{Var}(\phi_i - \bar{\phi}) + 2\lambda\,\mathrm{Cov}\big(A_i^{\mathrm{GSPO}},\, \phi_i - \bar{\phi}\big). \end{aligned} \tag{12}$$

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

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

where the inequality is guaranteed by Assumption A.1.

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

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

When $V_\phi > 0$, this quadratic is strictly less than $\mathrm{Var}(A_i^{\mathrm{GSPO}})$ for $\lambda \in \left(0, \frac{2C}{V_\phi}\right)$; when $V_\phi = 0$, it holds for any $\lambda > 0$. Setting $\lambda_{\max} = \frac{2C}{V_\phi + \epsilon}$ ($\epsilon > 0$ 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 the trajectory τ_i, the shaping term $\lambda\phi_i$ in A_i^{IBPO} injects a low-noise signal related to the trajectory's process quality, making it more correlated with the gradient direction than the pure terminal reward. Consequently, IBPO achieves significantly lower gradient variance in practice, especially in scenarios with longer trajectories or multiple reasoning errors. □

Discussion. This theorem shows that, under Assumption A.1, the shaping term $\lambda\phi_i$ introduced by IBPO's counterfactual comparison can reduce the variance of advantage estimation. It is worth noting that the negative correlation in Assumption A.1 is directly implied by the construction of ϕ_i (correct trajectories have $\phi_i = 0$, incorrect trajectories have $\phi_i > 0$), so this assumption is essentially a corollary of the definition rather than an additional constraint. The main value of the theorem lies in quantifying the range of λ for which variance reduction holds, providing theoretical guidance for hyperparameter selection. More importantly, ϕ_i encodes process-level information, enabling the advantage estimate to reflect not only whether the answer is correct but also the *degree* to which the reasoning deviates from correctness. This achieves finer-grained credit assignment and empirically supports the optimization stability and sample efficiency demonstrated by IBPO in mathematical and code reasoning tasks.

Empirical Verification. To verify the practical significance of the above theoretical analysis, we directly measured the policy gradient variance of GSPO and GSPO+IBPO during Qwen3-32B training on AIME25. The results show that: (i) the negative correlation condition in Assumption A.1 consistently holds during actual training ($\mathrm{Cov}(Y_i, \phi_i)$ is negative at all checkpoints); (ii) IBPO reduces policy gradient variance by approximately 30% on average, which is consistent with the smoother reward evolution observed in the training curves (Figure 2) and faster convergence speed.

B. Details of Compute Budget Matching

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

In our experiments, the compute budget of IBPO+GSPO and GSPO is matched via actual GPU usage time. Specifically, for each prompt x, IBPO+GSPO first generates 8 responses y. For each incorrect response y_i, it is concatenated with the original input x and a randomly sampled correct response to form a new input, and a correction output is generated. We note that the input sequences in the correction phase are longer due to the concatenated context, and the quadratic complexity of attention computation makes each correction trajectory more expensive than base sampling. Therefore, we do not assume that the FLOPs of the two-stage generation are identical to those of GSPO's 64 samples. Instead, we adopt a more direct matching approach: we measure the actual GPU usage time of GSPO with 64 samples and run IBPO+GSPO within the same GPU time budget. This means that under the same wall-clock training time constraint, the two methods consume equivalent actual computational resources. The horizontal axis in Figure 2 corresponds to this actual training compute budget.

C. Instantiation Details of IBPO

This appendix presents a specific instance of IBPO used in our experiments, namely the *compare-and-correct* mechanism, along with the integrated training pipeline and implementation details when combined with GSPO. We emphasize that the following design is one specific choice for the comparison operator $\mathcal{M}$ described 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 use the *compare-and-correct* mechanism to instantiate the general comparison operator $\mathcal{M}$. We emphasize that this is *one specific implementation* 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 the *compare-and-correct* instantiation of the operator $\mathcal{M}$. Specifically, we first generate multiple candidate reasoning trajectories, then use the model itself to compare these trajectories and rewrite them into corrected outputs. This implementation can be viewed as a specific choice of $\mathcal{M}$ that maps counterfactual differences to computable shaping terms $\phi(\cdot)$. To avoid limiting the contribution of this work to a specific implementation, we defer implementation details—such as how to induce trajectory diversity and how to construct comparison inputs—to the appendix.

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

$$\hat{y} \sim \pi_\theta(\cdot \mid \tilde{x}), \qquad \hat{y} = \mathcal{C}(x; y, y^{\mathrm{ref}}). \tag{13}$$

Prompt Template. In all experiments, we use the following compare-and-correct instruction template (with minor adjustments depending on the task format):

> 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 aims 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 *recoverability*-based shaping instance 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:

$$\Delta(x; y, y^{\text{ref}}) = \beta \cdot \mathbb{I}[r(x, y) = 0 \ \wedge \ r(x, \hat{y}) = 1], \qquad \beta = 0.5. \tag{14}$$

The resulting shaped sequence-level reward is

$$r'(x, y) = r(x, y) + \lambda \, \Delta(x; y, y^{\text{ref}}). \tag{15}$$

Token-Level Edit Distance Variants. When correction succeeds ($r(x, y) = 0 \wedge r(x, \hat{y}) = 1$), we can further localize the specific tokens that were modified by computing the token-level edit distance between the original response y and the corrected output $\hat{y}$. Let $y = (y_1, \ldots, y_T)$, $\hat{y} = (\hat{y}_1, \ldots, \hat{y}_T)$ (after alignment), and define the set of unmodified tokens $\mathcal{U} = \{t : y_t = \hat{y}_t\}$.

IBPO-ratio (unmodified ratio reward shaping): The proportion of unmodified tokens $|\mathcal{U}|/T$ is used as the recoverability measure, directly serving as β in Equation (14):

$$\Delta^{\text{ratio}}(x; y, y^{\text{ref}}) = \frac{|\mathcal{U}|}{T} \cdot \mathbb{I}[r(x, y) = 0 \ \wedge \ r(x, \hat{y}) = 1]. \tag{16}$$

Intuitively, a higher unmodified ratio indicates that the original reasoning is closer to being correct and should receive a higher shaping reward.

IBPO-mask (token-level gradient masking): Gradient contributions from unmodified tokens are masked, and policy gradient updates are applied only to modified tokens. Define the token-level mask $m_t = \mathbb{I}[t \notin \mathcal{U}]$, then the gradient coefficient for the t-th token in the policy gradient is multiplied by m_t:

$$\nabla_\theta \mathcal{J}^{\text{mask}} \propto \sum_{t=1}^{T} m_t \cdot A_t \cdot \nabla_\theta \log \pi_\theta(y_t \mid y_{<t}, x), \tag{17}$$

where unmodified tokens ($m_t = 0$) are not penalized, as they were retained after correction, indicating that these tokens are likely correct reasoning steps.

Why No Penalty for Failed Corrections. For cases where $r(x, y) = 0$ and $r(x, \hat{y}) = 0$, we do not impose additional negative penalties, to avoid misattributing insufficient correction capability or poor reference quality to the intrinsic quality of the original reasoning. This design choice helps prevent unnecessary bias and training instability.

C.3. Full Rewrite Detection and Suppression

Problem. During the counterfactual compare-and-correct process, the model may not perform local repairs on the original erroneous trajectory but instead completely ignore the original reasoning and generate an entirely new solution from scratch. This "full rewrite" behavior leads to reward hijacking: when the corrected output happens to be correct, the shaping reward is incorrectly attributed to the "recoverability" of the original trajectory, while in reality the original reasoning process was not utilized.

Detection Mechanism. We use Python's token-level edit distance to detect full rewrites. Let $d(a, b)$ denote the normalized edit distance between sequences a and b (valued in $[0, 1]$). When correction succeeds ($r(x, y) = 0 \wedge r(x, \hat{y}) = 1$), if both of the following conditions are satisfied, the case is classified as a full rewrite and $\Delta = 0$ is set:

$$d(y, \hat{y}) > \alpha \quad \text{and} \quad d(y, \hat{y}) > d(\hat{y}, y^{\text{ref}}), \tag{18}$$

where α is the edit distance threshold. The first condition detects the degree of deviation of the corrected output from the original trajectory; the second condition confirms that the corrected output is closer to the reference answer than to the original trajectory, i.e., the model tends to copy the reference rather than repair the original reasoning.

Combining with Equation (14), the complete shaping term with rewrite filtering is:

$$\Delta(x; y, y^{\text{ref}}) = \beta \cdot \mathbb{I}\big[r(x, y) = 0 \wedge r(x, \hat{y}) = 1 \wedge \neg\text{rewrite}(y, \hat{y}, y^{\text{ref}})\big]. \tag{19}$$

Threshold α	Flagged as Rewrite (%)	AIME25 (%)
50%	27.45	85.3
55%	23.36	85.5
60%	18.42	85.6
65%	13.37	85.5
70%	9.82	85.4
75%	5.73	85.3
80%	2.06	85.3

Table 5. Rewrite detection rate and AIME25 accuracy under different edit distance thresholds α (Qwen3-32B). Performance remains stable in the 55%–70% range, and 60% is selected as the midpoint of this plateau.

Threshold Sensitivity Analysis. We conducted a sensitivity analysis on the threshold α using Qwen3-32B (AIME25), with results shown in Table 5.

Adaptive Threshold. A fixed threshold is not conducive to cross-domain generalization. A more robust approach is distribution-based anomaly detection: compute the mean μ_d and standard deviation σ_d of all edit distances within the current batch, and set the threshold as $\alpha = \mu_d + 2\sigma_d$. By Chebyshev's inequality, even if the distribution is non-normal, at least 75% of the data falls within $\mu_d \pm 2\sigma_d$, making this criterion conservatively effective under any distribution.

Suppressing Rewrites at the Source via Reinforcement Learning. Beyond post-hoc detection, we also incorporate the edit distance constraint directly into the reward function: when the edit distance between the corrected output and the original erroneous trajectory is too large, an additional negative reward penalty is imposed. Specifically, multiple correction results are generated simultaneously, and contrastive reinforcement learning is applied within the group, so that the model naturally learns during training that local repair yields higher returns than full rewriting, thereby suppressing rewrite tendencies at the behavioral policy level. This is a more fundamental solution than threshold filtering: rather than discarding samples after rewrites occur, the incentive mechanism encourages the model to actively avoid rewrites.

Positioning Note. The edit distance check is a defensive safeguard against reward hijacking, not a core component of the IBPO framework. When the model possesses sufficient base capability, full rewrites are inherently rare events, and the specific threshold choice has negligible impact on the main experimental results. Combined with the above RL training mechanism, as training progresses, the model's rewrite tendency further decreases, and the dependence on the threshold correspondingly diminishes.

C.4. Trajectory Diversity and Coupling Reduction

IBPO relies on sampling multiple trajectories with sufficient diversity under the same input. 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 slight perturbations to the system prompt or format prompt 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 (minimum ratio 0.1) and linear warmup over 3% of total steps. The entropy regularization coefficient is set to 0.

Sampling. GSPO uses $G = 64$ rollouts per prompt, while IBPO uses $G = 8$, to approximately match the overall compute budget across methods.

E. An Instantiation of IBPO: Integration with GSPO

GSPO is used solely as the carrier optimizer; replacing GSPO with GRPO or PPO does not alter the IBPO formulation. In the preceding sections, we have introduced the general formulation of IBPO and its reward shaping definition. IBPO is a training formulation orthogonal to the underlying sequence-level reinforcement learning method. This section describes how to seamlessly integrate this formulation into a representative sequence-level reinforcement learning algorithm—GSPO.

E.1. Preliminaries: Sequence-Level Reinforcement Learning

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

$$\pi_\theta(y \mid x) = \prod_{t=1}^{|y|} \pi_\theta(y_t \mid x, y_{<t}). \tag{20}$$

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

$$J(\theta) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_{\theta_{\text{old}}}(\cdot|x)}\Big[\mathcal{L}\big(s(\theta; x, y), \hat{A}(x, y)\big)\Big], \tag{21}$$

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

IBPO does not alter the optimization form in Equation (21), but instead shapes the original sequence-level reward through the model's self-correction process.

E.2. IBPO and GSPO: Single-Pass Joint Training

In each iteration, we perform a **single** policy update: for the same batch of queries x, we simultaneously construct the candidate response set for sequence-level reinforcement learning, as well as the self-correction results for evaluating recoverability. IBPO only modifies the reward definition (shaping it into r'), while keeping the GSPO surrogate objective unchanged.

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

$$s_i(\theta) = \left(\frac{\pi_\theta(y_i \mid x)}{\pi_{\theta_{\text{old}}}(y_i \mid x)}\right)^{\frac{1}{|y_i|}}. \tag{22}$$

We use the **shaped reward** to construct the within-group advantage:

$$\hat{A}_i = \frac{r'(x, y_i) - \mu'}{\sigma'}, \tag{23}$$

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

$$J_{\text{GSPO}}(\theta) = \mathbb{E}_{x\sim\mathcal{D},\, \{y_i\}_{i=1}^G\sim\pi_{\theta_{\text{old}}}(\cdot|x)}\Bigg[\frac{1}{G}\sum_{i=1}^{G}\min\Big(s_i(\theta)\,\hat{A}_i,\; \text{clip}\big(s_i(\theta),\, 1-\epsilon,\, 1+\epsilon\big)\,\hat{A}_i\Big)\Bigg]. \tag{24}$$

(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} (sampled from correct responses if at least one exists in the group; otherwise sampled from incorrect responses), and ask the model to compare and correct:

$$\hat{y}_i = \mathcal{C}(x; y_i, y_i^{\text{ref}}). \tag{25}$$

This process does not introduce any additional supervision and is used solely to evaluate whether the model can correct an incorrect response to a correct one. Based on an instantiation of Equation (14), we define

$$\Delta_i = \beta \cdot \mathbb{I}[r(x, y_i) < 1 \ \wedge \ r(x, \hat{y}_i) = 1], \qquad \beta = 0.5, \tag{26}$$

and obtain the sequence-level shaped reward:

$$r'(x, y_i) = r(x, y_i) + \lambda \, \Delta_i. \tag{27}$$

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 assignment information.

(C) Joint GSPO Training on 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 on an extended input space**, and train it using the **same GSPO objective** jointly. Formally, this process does not introduce a new Markov Decision Process (MDP), but merely 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:

$$\tilde{x}_i = (x; \ y_i, \ y_i^{\text{ref}}), \tag{28}$$

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 based on the correctness of the final answer $r(\tilde{x}_i, \hat{y}_i) \in \{0, 1\}$.

We include these correction samples together with the original reasoning samples in GSPO's sequence-level optimization. Formally, the GSPO objective can be written as a unified expectation over a **mixed input distribution**:

$$\begin{aligned} J_{\text{GSPO}}^{\text{joint}}(\theta) = \mathbb{E}_{\tilde{x} \sim \mathcal{D}_{\text{mix}}} \Bigg[\frac{1}{G} \sum_{i=1}^{G} \min\Big(& s_i(\theta) \, \hat{A}_i, \\ & \text{clip}\big(s_i(\theta), \, 1-\epsilon, \, 1+\epsilon\big) \, \hat{A}_i \Big) \Bigg], \end{aligned} \tag{29}$$

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

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

E.3. IBPO + GSPO Algorithm Pseudocode

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

Algorithm 1 Instantiation of IBPO with GSPO: Single-Pass Joint Training

Require: Dataset $\mathcal{D}$; current policy π_θ; old policy $\pi_{\theta_{\text{old}}}$; group size G; clipping parameter 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 mini-batch of prompts $\mathcal{B} = \{x\}$ from $\mathcal{D}$.
  for each prompt $x \in \mathcal{B}$ **do**
    (A) Sample rollout trajectories 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 reward.
    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**
          Uniformly sample y_i^{ref} from $\{y_j : r(x, y_j) = 1\}$.
        else
          Uniformly sample y_i^{ref} from $\{y_j : r(x, y_j) = 0\}$.
        end if
        Generate correction result $\hat{y}_i \leftarrow \mathcal{C}(x; y_i, y_i^{\text{ref}})$.
        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 advantage 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 on correction behavior.
  Construct correction input $\tilde{x}_i \leftarrow (x;\ y_i,\ y_i^{\text{ref}})$ for corrected cases.
  Sample $\hat{y}_i \sim \pi_{\theta_{\text{old}}}(\cdot \mid \tilde{x}_i)$ and compute terminal reward $r(\tilde{x}_i, \hat{y}_i)$.
  If joint training is enabled, add the same GSPO surrogate on $\tilde{x}_i$ (Equation 29).
  Update θ by maximizing $\sum_{x \in \mathcal{B}} \mathcal{J}_x(\theta)$ (plus the joint objective if enabled).
end for