BPPO: Binary Prefix Policy Optimization for Efficient GRPO-Style Reasoning RL with Concise Responses

Anonymous ACL submission

Abstract

Group Relative Policy Optimization (GRPO) is widely used for training reasoning models, but updating all sampled completions in each group incurs substantial cost and can reinforce verbose reasoning trajectories. In this paper, we study whether all completions provide equally useful update signals in GRPO-style reasoning RL. Our gradient-similarity analysis shows that, within the same prompt group, same-class completions often induce highly similar update directions, whereas correct–incorrect pairs provide more distinct contrastive signals. Motivated by this observation, we propose Binary Prefix Policy Optimization (BPPO), which uses the shortest correct completion and the shortest incorrect completion as a compact update unit while preserving full-group advantage normalization. BPPO further improves efficiency with adaptive completion scheduling and prefix-focused optimization; by updating only response prefixes, it avoids reinforcing redundant suffixes and encourages more concise responses. Experiments on GSM8K, MATH, and Geo3K show that BPPO achieves up to $6.08\times$ speedup over GRPO while maintaining competitive accuracy, and reduces mean response length by approximately 30–50% without modifying the reward with an explicit length penalty.

1 Introduction

Large language models (LLMs) have achieved strong performance on multi-step reasoning tasks, especially in mathematics and structured problem solving (Wei et al., 2022; Kojima et al., 2022; Cobbe et al., 2021; Hendrycks et al., 2021). Reinforcement learning has become an important approach for improving reasoning performance, with policy optimization methods such as PPO and its variants demonstrating substantial empirical gains in instruction following, alignment, and reasoning-oriented optimization (Schulman et al., 2017; Stiennon et al., 2020; Ouyang et al., 2022; Bai et al., 2022). Building on this line of methods, Group Relative Policy Optimization (GRPO) avoids training a separate critic model, thereby simplifying the reinforcement learning pipeline and making it a widely adopted approach for reasoning model optimization (Shao et al., 2024; Guo et al., 2025; Team, 2025; Yu et al., 2026; Zheng et al., 2025).

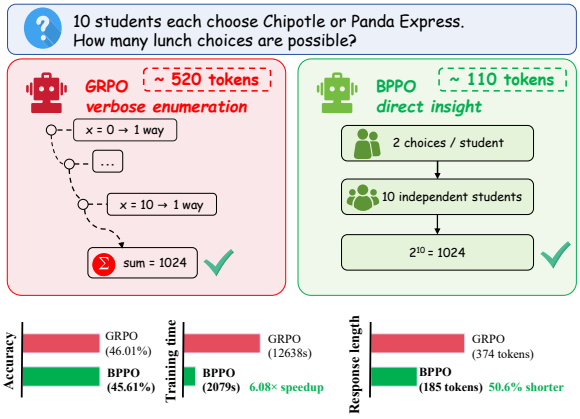


Figure 1: A MATH case study from Llama3.2-3B-Instruct trained with GRPO and BPPO. Both methods produce the correct answer, but GRPO relies on verbose enumeration, whereas BPPO exploits the problem structure and follows a shorter reasoning path. In the corresponding MATH setting, BPPO maintains comparable accuracy while achieving a $6.08\times$ training speedup and reducing mean response length by 50.6%.

Despite its success, GRPO incurs substantial training cost. For each prompt, it samples a group of G completions, computes rewards for all of them, normalizes advantages within the group, and performs policy updates using the entire group. This design is expensive in two ways. First, the cost grows linearly with the number of sampled completions. Second, reasoning traces are often long, and Transformer attention scales quadratically with sequence length (Vaswani et al., 2017; Tay et al., 2022). Beyond training efficiency, full-response updates may also reinforce verbose reasoning patterns (Samineni et al., 2025; Shen et al., 2023; Bu

et al., 2025; Li et al., 2024; Park et al., 2024; Dumitru et al., 2025). Although efficient attention and memory-management techniques have reduced some practical costs of long-sequence computation (Dao et al., 2022; Dao, 2024; Kwon et al., 2023), updating on all completions and all response tokens in every sampled group remains computationally expensive and may encourage unnecessarily long reasoning trajectories. This motivates more sample- and token-efficient reinforcement learning methods for reasoning models (Lin et al., 2026; Lee and Tong, 2025; Sang et al., 2026), and also highlights response length as an important dimension of efficient reasoning optimization.

A natural question is which parts of a sampled group provide the most useful update signals. Our analysis shows that same-prompt completions from the same correctness class often induce highly similar gradients, suggesting overlapping optimization directions. In contrast, correct and incorrect completions yield more distinct update directions, motivating a correct–incorrect pair as a compact update unit. Among such pairs, we select the shortest correct–incorrect pair to preserve cross-class contrast while favoring compact reasoning trajectories. Since early reasoning steps often shape later solution direction, response prefixes can preserve useful token-level update signals without full-response updates. Moreover, avoiding updates on redundant suffixes also reduces reinforcement of unnecessary reflection or repeated verification, which may encourage more concise reasoning trajectories.

Based on these observations, we propose Binary Prefix Policy Optimization (BPPO), which keeps full-group reward and advantage computation unchanged but applies policy updates only to the response prefixes of the shortest correct and shortest incorrect completions. This design reduces update-bearing data at both the completion and token levels, while avoiding direct reinforcement of long reasoning trajectories that contain redundant suffixes. To avoid reduced hardware utilization after pair selection, we introduce adaptive completion scheduling, which discards all-correct and all-incorrect groups before importance-ratio computation and packs selected pairs from multiple prompts to maintain efficient device usage.

We evaluate BPPO on GSM8K, MATH, and Geo3K using models that span different scales, model families, and modalities, including Qwen2.5-1.5B-Instruct, Llama3.2-3B-Instruct, and Qwen3-VL-4B. Across these settings, BPPO achieves up to 5.90× speedup over GRPO on GSM8K, 6.08× on MATH, and 3.86× on Geo3K, while maintaining competitive accuracy. In addition, compared with GRPO, BPPO reduces the mean response length by approximately 30–50% without modifying the reward with an explicit length penalty.

2 Related Work

RL for Reasoning. Reinforcement learning has become an important paradigm for improving LLMs. Early successes were largely driven by reinforcement learning with human feedback (RLHF), which substantially improved instruction following and safety (Ouyang et al., 2022; Bai et al., 2022; Stiennon et al., 2020; Rafailov et al., 2023; Meng et al., 2024). More recently, automatically verifiable supervision and reward signals have shown strong potential for enhancing reasoning capabilities, especially in mathematical and other structured domains where correctness can be checked reliably (Lightman et al., 2024; Shao et al., 2024; Guo et al., 2025; Team, 2025; Wang et al., 2024; Trung et al., 2024). In terms of policy optimization, PPO remains a standard choice (Schulman et al., 2017), while GRPO has become an important approach for reasoning training because it avoids explicit value modeling through group-relative advantage normalization (Shao et al., 2024). Subsequent work further demonstrated that GRPO-style training can induce strong reasoning behaviors in LLMs (Guo et al., 2025; Team, 2025), and also inspired a number of follow-up variants and refinements. For example, DAPO improves scalability and training stability for large-scale reasoning RL (Yu et al., 2026), while GSPO revisits the optimization objective from a sequence-level perspective and reports improved efficiency and stability over GRPO (Zheng et al., 2025).

Acceleration Methods for Reasoning RL. The high cost of reasoning RL has motivated recent efforts to improve the efficiency and scalability of GRPO-style training (Zheng et al., 2026; Sun et al., 2025; Fu et al., 2026; Chen et al., 2026). DAPO improves reasoning RL through several system and algorithmic refinements, including dynamic sampling and overlong filtering (Yu et al., 2026). Other work reduces unnecessary computation more directly. CPPO accelerates GRPO by pruning completions with low contribution to policy updates, reducing the number of trajectories used in optimization (Lin et al., 2026). Token-efficient RL

methods further update only selected subsets of response tokens to reduce computation cost (Lee and Tong, 2025; Sang et al., 2026; Mundada et al., 2026). These studies suggest that efficiency in reasoning RL can be improved not only through system-level optimization, but also by selectively reducing completion- or token-level updates while preserving useful learning signals, highlighting the need to identify which trajectories and tokens are most informative for policy optimization.

Response-Length Control. Long reasoning traces can improve multi-step problem solving, but they also increase training and inference cost and may contain redundant derivations or repeated verification (Liu et al., 2026; Fang et al., 2026; Dai et al., 2026). Prior work has observed that RLHF and RL post-training can favor longer or more verbose responses, partly because sequence-level rewards provide coarse feedback, resulting in coarse credit assignment that may reinforce all sampled tokens in positively rewarded completions (Singhal et al., 2024; Samineni et al., 2025; Liu et al., 2025). To control response length, recent methods introduce explicit length-aware mechanisms, including overlong filtering (Yu et al., 2026), adaptive length penalties and reward shaping (Li et al., 2025; Xiang et al., 2025), length-constrained RL objectives (Aggarwal and Welleck, 2025), RL-based chain-of-thought pruning (Hou et al., 2026), and dynamic allocation of inference compute (Arora and Zanette, 2026). These works highlight response length as an important dimension of reasoning-RL optimization, alongside accuracy and training efficiency.

3 Method

3.1 Group Relative Policy Optimization

GRPO simplifies reinforcement learning by removing the need for a separate critic and computing relative advantages from rewards within each sampled completion group. For each prompt $q \sim \mathcal{D}$, it samples G completions $\{o_1, \ldots, o_G\} \sim \pi_{\theta_{\text{old}}}(\cdot \mid q)$, computes rewards for all completions, and normalizes them within the group to obtain completion-level advantages. The advantage $\hat{A}_i$ is then applied to completion o_i during token-level policy optimization. The token-level importance ratio is $\rho_{i,t}(\theta) = \pi_\theta(o_{i,t} \mid q, o_{i,<t}) / \pi_{\theta_{\text{old}}}(o_{i,t} \mid q, o_{i,<t})$, where $o_{i,<t}$ denotes the response prefix before token t. The current policy π_θ is optimized against a frozen reference policy π_{ref} with a clipped policy-gradient objective and an additional KL regularization term:

$$J(\theta) = \mathbb{E}_{q\sim\mathcal{D},\, \{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q)} \Bigg[\frac{1}{G} \sum_{i=1}^{G} \Bigg(\frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \Big[\min\Big(\rho_{i,t}(\theta)\hat{A}_i,\ \text{clip}(\rho_{i,t}(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_i\Big) - \beta\, \mathbb{D}_{\text{KL}}[\pi_\theta \,\|\, \pi_{\text{ref}}] \Big] \Bigg) \Bigg]. \tag{1}$$

where ε and β are hyperparameters, the KL term is defined as

$$\mathbb{D}_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}}) = \frac{\pi_{\text{ref}}(o_{i,t} \mid q, o_{i,<t})}{\pi_\theta(o_{i,t} \mid q, o_{i,<t})} - \log \frac{\pi_{\text{ref}}(o_{i,t} \mid q, o_{i,<t})}{\pi_\theta(o_{i,t} \mid q, o_{i,<t})} - 1. \tag{2}$$

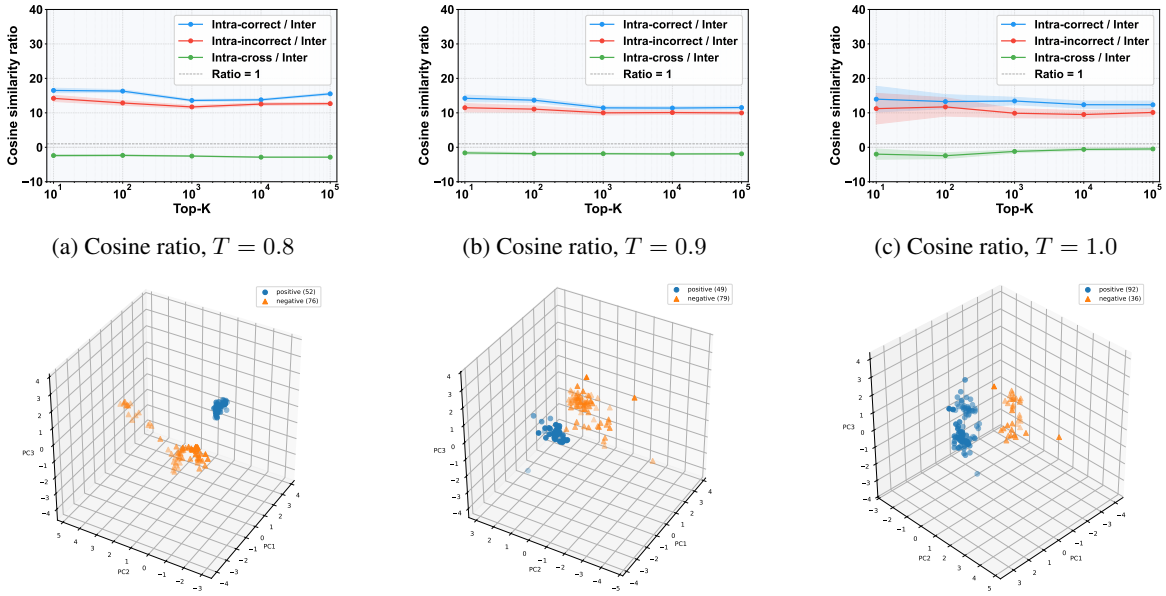
and the advantage $\hat{A}_i$ is computed using a group of rewards $\{r_1, r_2, \ldots, r_G\}$:

$$\hat{A}_i = \frac{r_i - \text{mean}(\{r_1, r_2, \ldots, r_G\})}{\text{std}(\{r_1, r_2, \ldots, r_G\})}. \tag{3}$$

3.2 Gradient Redundancy

To better understand how useful update signals are distributed within a sampled group, we analyze gradient similarity among completions sampled for the same prompt in GRPO training. Using Qwen2.5-1.5B-Instruct on GSM8K, we conduct this analysis under three sampling temperatures: $T = 0.8$, 0.9, and 1.0. At each temperature, we sample $G = 16$ completions per prompt.

Top-K Cosine Similarity Ratios. For each completion, we compute its gradient, retain the top-K entries with the largest absolute values for $K \in \{10^1, 10^2, 10^3, 10^4, 10^5\}$, and measure cosine similarity on the truncated vector. We consider three types of same-prompt completion pairs: correct–correct, incorrect–incorrect, and correct–incorrect. For each type, we report the ratio between its cosine similarity and the average similarity of completion pairs from different prompts. Figure 2 reports these ratios as Intra-correct / Inter, Intra-incorrect / Inter, and Intra-cross / Inter. Correct–correct and incorrect–incorrect pairs show much higher similarity ratios across temperatures, while correct–incorrect pairs remain low and negative. The PCA projections further reveal a class-dependent gradient structure, suggesting that same-class completions provide overlapping update directions, whereas correct–incorrect pairs retain distinctive contrastive signals.



(a) Cosine ratio, $T = 0.8$ (b) Cosine ratio, $T = 0.9$ (c) Cosine ratio, $T = 1.0$

(d) PCA, $T = 0.8$ (e) PCA, $T = 0.9$ (f) PCA, $T = 1.0$

Figure 2: **Gradient similarity under different sampling temperatures.** (a)–(c) show top-K gradient cosine-similarity ratios for $T = 0.8$, 0.9, and 1.0, with shaded regions indicating 3σ intervals. Each ratio normalizes same-prompt gradient similarity by the average similarity between completions from different prompts. (d)–(f) show PCA projections of gradient vectors from 128 completions sampled from same prompt. Across temperatures, same-class pairs exhibit substantially higher similarity, whereas correct–incorrect pairs remain more distinct.

3.3 Pair Selection

The gradient-similarity analysis suggests that same-class completions often provide overlapping update directions, while correct–incorrect pairs provide more distinctive contrastive signals. Therefore, BPPO restricts policy updates to one selected correct completion and one selected incorrect completion from each sampled group.

Let $\mathcal{I}^+$ and $\mathcal{I}^-$ denote the sets of correct and incorrect completions, respectively, and let $\ell(o_i)$ be the response length of completion o_i. BPPO selects one completion from each class to form the update set $\mathcal{S}(q) = \{i^*_+, i^*_-\}$. Specifically, it selects the shortest completion from each correctness class, preserving correctness contrast while favoring compact update trajectories. As shown in Section 4.4, this strategy achieves a favorable trade-off among accuracy, training efficiency, and response conciseness:

$$i^*_+ = \arg\min_{i \in \mathcal{I}^+} \ell(o_i), \quad i^*_- = \arg\min_{i \in \mathcal{I}^-} \ell(o_i). \quad (4)$$

3.4 FIRSTN: Prefix-Focused Gradient Update

While pair selection reduces the number of update-bearing completions, long reasoning traces can still make token-level optimization costly. Since early tokens often guide reasoning direction, we introduce FIRSTN, which restricts policy-gradient updates to the first n response tokens of each selected completion. Full-response generation and reward computation remain unchanged, while updates are applied only to response prefixes. This preserves full-rollout reward supervision, reduces optimization cost on long responses, and avoids reinforcing redundant suffixes such as unnecessary reflection or repeated verification, which may encourage more concise responses. We study the effect of prefix length in Section 4.4.

3.5 BPPO

Binary Prefix Policy Optimization (BPPO) combines selected-pair updates at the completion level with prefix-focused updates at the token level. Specifically, BPPO restricts policy updates to the shortest correct completion and the shortest incorrect completion from each sampled group, using them as a compact contrastive update unit, and further applies policy-gradient updates only to their response prefixes. Relative to GRPO, BPPO preserves the same sampling process, reward computation, and group-relative advantage normalization, changing only where policy updates are applied. Let $\mathcal{S}(q) = \{i^*_+, i^*_-\}$ denote the selected shortest correct–incorrect pair for prompt q, and let n de-

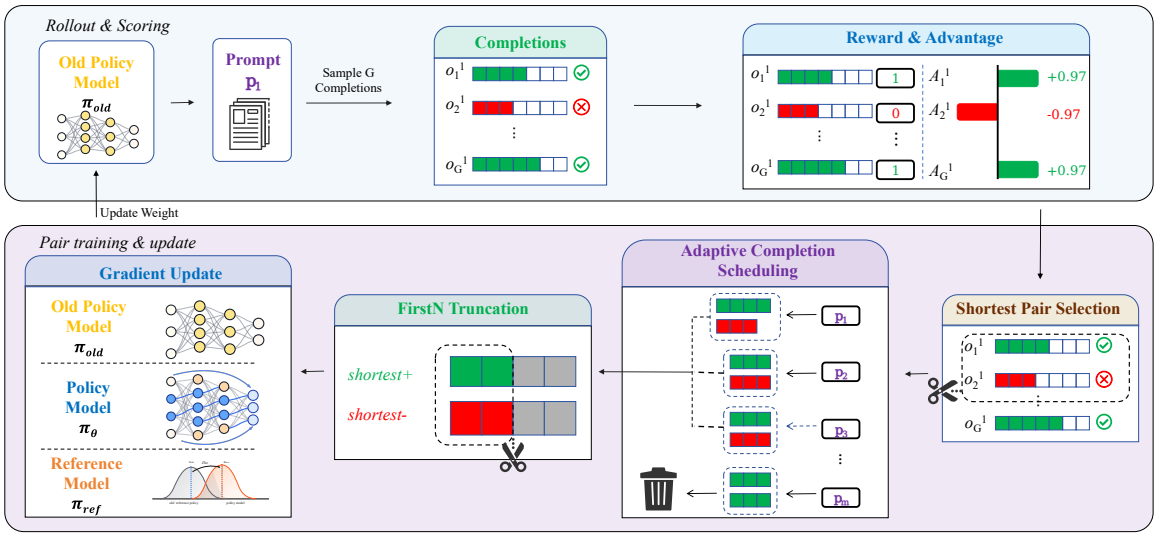


Figure 3: **Overview of the BPPO training workflow.** The top panel shows rollout and scoring with full-group rewards and advantages as in GRPO. The bottom panel shows pair-based training: each prompt first retains the shortest correct–incorrect pair, discards groups without both correctness classes, packs retained pairs from multiple prompts into the update batch, and applies prefix-focused updates to the retained completions.

note the number of response tokens used for prefix-focused optimization. The resulting BPPO objective is:

$$J(\theta) = \mathbb{E}_{q\sim\mathcal{D},\,\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q)}\Bigg[\frac{1}{|\mathcal{S}(q)|}\sum_{i\in\mathcal{S}(q)}\Bigg(\frac{1}{n}\sum_{t=1}^{n}\Big[\min\Big(\rho_{i,t}(\theta)\hat{A}_i,\ \text{clip}(\rho_{i,t}(\theta),1-\varepsilon,1+\varepsilon)\hat{A}_i\Big) - \beta\,\mathbb{D}_{\text{KL}}[\pi_\theta \,\|\, \pi_{\text{ref}}]\Big]\Bigg)\Bigg]. \tag{5}$$

Adaptive Completion Scheduling. Pair selection leaves each prompt with at most one retained correct–incorrect pair, reducing update-bearing completions under fixed prompt batches. To maintain GPU utilization, adaptive completion scheduling aggregates retained pairs from additional prompts until the per-device update budget matches the original budget, while discarding all-correct and all-incorrect groups before importance-ratio computation. This improves utilization and reduces optimization steps per epoch without changing the pair-selection rule or prefix-update objective. The procedure is shown in Figure 3 and Appendix A.1; Appendix B.3 reports matched-step results without adaptive scheduling.

4 Experiments

4.1 Experimental Settings

Training Details. We implement BPPO within the verl framework (Sheng et al., 2025) and use vLLM (Kwon et al., 2023) for efficient completion generation. We experiment with Qwen2.5-1.5B-Instruct (Qwen et al., 2025), Llama3.2-3B-Instruct (Grattafiori et al., 2024), and Qwen3-VL-4B (Bai et al., 2025), using one H100 GPU for the 1.5B model and four H100 GPUs for larger models. Unless otherwise specified, all models are trained for one epoch with sampling temperature 1.0 and learning rate 1×10^{-6}. We use rollout group sizes $G \in \{8, 16, 32\}$, report the $G = 8$ and $G = 16$ results in the main text, and provide $G = 32$ results in Appendix B.2. We set $G = 16$ by default for ablations. For CPPO (Lin et al., 2026), we use pruning rates of 75% for $G = 8$ and 87.5% for $G = 16$ to match BPPO's retained-completion budget. Each setting is trained three times independently, and we report mean $\pm$ standard deviation. The maximum completion length is set to 2048 tokens for GSM8K and MATH, and to 8192 tokens for Geo3K. For all experiments involving FIRSTN, we update only the first n response tokens, where n is set to 50% of the average response length.

Evaluation Details. We evaluate the main results on three reasoning benchmarks: MATH (Hendrycks et al., 2021), GSM8K (Cobbe et al., 2021), and Geo3K (Lu et al., 2021). We further report additional evaluations on AIME24 (Zhang and Math-AI, 2024), MinervaMATH (Lewkowycz et al., 2022), and OlympiadBench (He et al., 2024) in Appendix B.5. Performance is measured by Pass@1 accuracy under greedy decoding.

Table 1: **Results on GSM8K with Qwen2.5-1.5B-Instruct under different rollout group sizes G.** BPPO achieves up to $5.90\times$ speedup and reduces mean response length by up to 43.9% while maintaining competitive accuracy. Negative length reduction indicates longer responses than GRPO.

Method	**Group Size G**	**Accuracy (%)**	**Mean Response Tokens**	**Length Reduction (%)**	**Training Time (s)**	**Speedup (times)**
Qwen2.5-1.5B-Instruct	-	23.63	-	-	-	-
GRPO	8	75.65 ± 0.43	259.05 ± 1.72	-	7186 ± 1339	1.00
CPPO		75.50 ± 0.40	285.72 ± 6.84	-10.3	2361 ± 93	3.04 ± 0.12
GRPO+FIRSTN		75.84 ± 0.54	232.01 ± 7.60	10.4	6002 ± 531	1.20 ± 0.17
Pair		76.24 ± 0.16	177.21 ± 2.06	31.6	2008 ± 322	3.58 ± 0.57
BPPO		75.34 ± 0.37	155.06 ± 3.32	**40.1**	1812 ± 20	$\mathbf{3.97} \pm 0.04$
GRPO	16	76.57 ± 0.57	262.64 ± 2.38	-	11816 ± 515	1.00
CPPO		76.85 ± 0.59	265.70 ± 6.68	-1.2	2541 ± 312	4.65 ± 0.54
GRPO+FIRSTN		76.55 ± 0.50	253.17 ± 7.10	3.6	11197 ± 174	1.06 ± 0.02
Pair		76.48 ± 0.42	161.03 ± 1.77	38.7	2271 ± 18	5.20 ± 0.04
BPPO		75.92 ± 0.38	147.25 ± 4.21	**43.9**	2002 ± 49	$\mathbf{5.90} \pm 0.24$

Table 2: **Results on MATH with Llama3.2-3B-Instruct under different rollout group sizes G.** BPPO achieves up to $6.08\times$ speedup and reduces mean response length by up to 50.6% while maintaining competitive accuracy.

Method	**Group Size G**	**Accuracy (%)**	**Mean Response Tokens**	**Length Reduction (%)**	**Training Time (s)**	**Speedup (times)**
Llama3.2-3B-Instruct	-	33.12	-	-	-	-
GRPO	8	45.33 ± 0.49	360.54 ± 3.01	-	9070 ± 244	1.00
CPPO		45.88 ± 0.35	417.70 ± 3.12	-15.9	3502 ± 63	2.59 ± 0.05
GRPO+FIRSTN		44.91 ± 0.20	319.77 ± 37.44	11.3	7904 ± 596	1.15 ± 0.06
Pair		45.68 ± 0.32	233.00 ± 8.73	35.4	3207 ± 42	2.83 ± 0.04
BPPO		45.23 ± 0.14	190.00 ± 7.50	**47.3**	2788 ± 22	$\mathbf{3.25} \pm 0.03$
GRPO	16	46.01 ± 0.77	374.83 ± 17.24	-	12638 ± 878	1.00
CPPO		46.19 ± 0.50	414.48 ± 11.60	-10.6	2408 ± 179	5.25 ± 0.38
GRPO+FIRSTN		46.38 ± 0.50	362.06 ± 26.02	3.4	11775 ± 813	1.07 ± 0.14
Pair		46.14 ± 0.21	192.74 ± 14.06	48.6	2249 ± 46	5.62 ± 0.41
BPPO		45.61 ± 0.15	185.20 ± 2.95	**50.6**	2079 ± 73	$\mathbf{6.08} \pm 0.55$

4.2 Main Results

We evaluate BPPO on GSM8K, MATH, and Geo3K. We compare **GRPO**, **CPPO** (Lin et al., 2026), and three controlled variants: **GRPO+FIRSTN**, which applies prefix-focused updates to all completions; **Pair**, which updates only on the shortest correct–incorrect pair; and **BPPO**, which combines pair selection with prefix-focused optimization. Further larger-model results and case studies are provided in Appendices B.1 and C.

Training on GSM8K. On GSM8K, Pair substantially reduces training time under both group sizes while maintaining competitive accuracy. Building on Pair, BPPO further applies prefix-focused optimization and achieves up to $5.90\times$ speedup at $G = 16$, with only minor accuracy changes relative to GRPO. Compared with CPPO, BPPO achieves higher speedups and much shorter responses across group sizes. At $G = 16$, Pair already reduces mean response length by 38.7% relative to GRPO, while adding FIRSTN further reduces it from 161.03 to 147.25 tokens, yielding a total reduction of 43.9%. Prefix-focused optimization further shortens responses by concentrating updates on early reasoning decisions and avoiding direct reinforcement of redundant suffixes, such as unnecessary reflection or repeated verification. Overall, these results suggest that updating only the response prefixes of the shortest pair can achieve competitive accuracy with GRPO, while improving training efficiency and producing more concise responses.

Training on MATH. As shown in Table 2, BPPO remains effective on MATH, a more challenging reasoning benchmark, achieving up to $6.08\times$ speedup over GRPO while maintaining competitive accuracy. Compared with CPPO,

Table 3: **Results on Geo3K with Qwen3-VL-4B under different rollout group sizes G.** BPPO achieves up to $3.86\times$ speedup and reduces mean response length by up to 45.7% while maintaining competitive accuracy.

Method	Group Size G	Accuracy (%)	Mean Response Tokens	Length Reduction (%)	Training Time (s)	Speedup (times)
Qwen3-VL-4B	-	31.82	-	-	-	-
GRPO	8	48.12 ± 0.82	3049.95 ± 19.66	-	25071 ± 166	1.00
CPPO		48.22 ± 0.90	4073.86 ± 106.60	-33.6	7750 ± 142	3.23 ± 0.07
GRPO+FIRSTN		47.12 ± 0.44	3016.77 ± 104.00	1.1	22981 ± 39	1.09 ± 0.01
Pair		49.67 ± 0.52	2136.67 ± 19.65	29.9	6912 ± 94	3.63 ± 0.04
BPPO		49.43 ± 0.50	2063.23 ± 22.77	**32.4**	6490 ± 172	$\mathbf{3.86} \pm 0.10$
GRPO	16	48.87 ± 0.23	3066.45 ± 27.28	-	29264 ± 184	1.00
CPPO		49.43 ± 0.85	3851.84 ± 63.57	-25.6	11773 ± 125	2.49 ± 0.03
GRPO+FIRSTN		47.97 ± 0.48	3006.12 ± 54.54	2.0	27007 ± 256	1.08 ± 0.01
Pair		49.75 ± 0.36	1830.37 ± 21.08	40.3	11388 ± 253	2.57 ± 0.04
BPPO		48.93 ± 0.48	1665.13 ± 7.38	**45.7**	10710 ± 108	$\mathbf{2.73} \pm 0.01$

BPPO achieves higher speedups and much more concise responses across both rollout group sizes; at $G = 16$, BPPO improves the speedup from $5.25\times$ to $6.08\times$ and reduces the mean response length by 50.6%, whereas CPPO increases response length by 10.6%. These results show that BPPO provides a favorable efficiency–conciseness trade-off even on a more challenging mathematical reasoning benchmark.

Training on Geo3K. We further evaluate BPPO on the multimodal Geo3K benchmark using Qwen3-VL-4B. As shown in Table 3, BPPO remains effective in this setting, achieving up to $3.86\times$ speedup over GRPO while maintaining competitive accuracy and producing substantially more concise responses. Compared with CPPO, BPPO achieves higher speedups and much shorter responses under both rollout group sizes; at $G = 16$, BPPO improves the speedup from $2.49\times$ to $2.73\times$ and reduces the mean response length by 45.7% relative to GRPO, whereas CPPO increases the mean response length by 25.6%. These results show that BPPO provides consistent efficiency and conciseness gains in the multimodal reasoning setting.

4.3 Compatibility with Other RL Objectives

Since BPPO only restricts update-bearing completions and tokens, it can be naturally combined with other reasoning-RL methods. Applying it to DAPO and GSPO preserves their original reward and advantage designs while substantially reducing training time with only minor accuracy changes, as shown in Table 4. Full results are provided in Appendix B.4.

Table 4: Compatibility results of BPPO with DAPO on MATH using Qwen3-8B and with GSPO on GSM8K using DeepSeek-R1-Distill-Qwen-7B.

Method	Accuracy (%)	Training Time (s)	Speedup (times)
DAPO	87.13	42062	1.00
DAPO+BPPO	86.74	20430	2.06
GSPO	92.72	15287	1.00
GSPO+BPPO	91.99	2669	5.73

Table 5: Efficiency contribution of BPPO components on GSM8K with Qwen2.5-1.5B-Instruct.

Method	Accuracy (%)	Training Time (s)	Speedup (times)
GRPO	76.63	12409	1.00
GRPO+FIRSTN	76.44	11398	1.09
Pair w/o ACS.	76.93	3942	3.15
Pair w/ ACS.	76.62	2270	5.47
BPPO	75.93	2021	6.14

4.4 Ablation Study

Efficiency Contributions of BPPO. Table 5 decomposes the efficiency gain of BPPO on GSM8K with Qwen2.5-1.5B-Instruct and $G = 16$. Applying FIRSTN alone gives a $1.09\times$ speedup, while restricting updates to a single correct–incorrect pair already increases the speedup to $3.15\times$ with comparable accuracy. Adaptive completion scheduling further improves the speedup to $5.47\times$ by maintaining device utilization, and adding prefix-focused updates yields the full BPPO variant with the largest speedup of $6.14\times$. Appendix B.6 provides a wall-clock timing breakdown on MATH without adaptive completion scheduling.

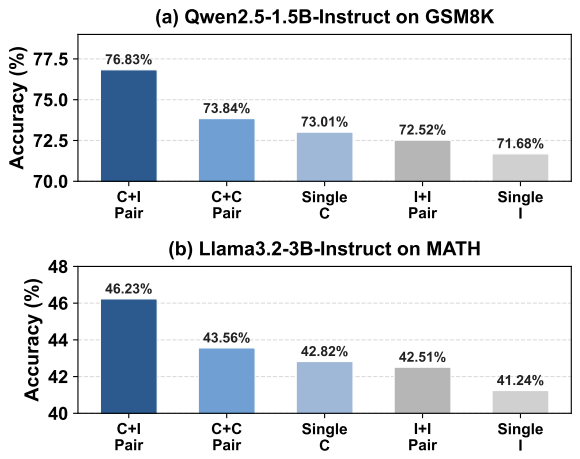


Figure 4: **Ablation on retained completion types.** (a) and (b) show results on GSM8K and MATH. The correct–incorrect (C+I) pair achieves the best accuracy, highlighting the importance of cross-class contrast.

Table 6: Ablation of pair-selection strategies on the GSM8K.

Strategy	Accuracy (%)	Mean Response Tokens	Training Time (s)
Random	76.83	248.31	2637
Shortest	76.62	159.15	2270
Long-C + Short-I	74.45	493.23	3630
Short-C + Long-I	39.83	116.87	1870
Longest	75.51	296.83	2847

Retained Completion Types. In Figure 4, C and I denote correct and incorrect completions. All variants in this ablation randomly sample retained completions from the specified correctness classes. **C+I Pair** outperforms same-class variants that retain only correct or only incorrect completions on both GSM8K and MATH.

Pair-Selection Strategy. We compare five strategies: **Random**, **Shortest**, **Long-C + Short-I**, **Short-C + Long-I**, and **Longest**. As shown in Table 6, **Random** gives the highest accuracy, while **Shortest** achieves comparable accuracy with shorter responses and lower training time. This is because **Shortest** provides a symmetric contrast: concise correct reasoning is reinforced, while concise but incorrect shortcuts are penalized, preventing collapse toward overly short responses. By contrast, **Short-C + Long-I** overemphasizes brevity and causes a large accuracy drop.

Prefix Length. The prefix length in FIRSTN controls how many response tokens contribute to the policy-gradient update. As shown in Figure 5, accuracy improves rapidly as the retained prefix ratio increases from 5% to 25%, and then gradually approaches full-response performance. At a 50% prefix ratio, performance reaches 75.93% on GSM8K and 45.69% on MATH, close to the full-response results of 76.62% and 45.97%, respectively.

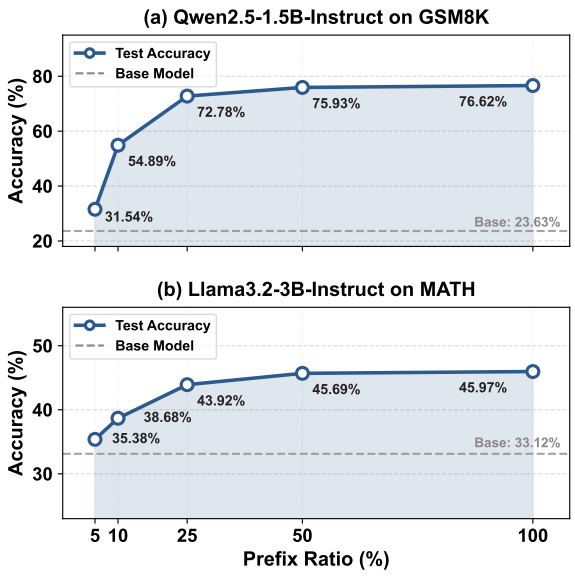


Figure 5: **Ablation on the retained prefix ratio in FIRSTN.** (a) and (b) show results on GSM8K and MATH, respectively. Accuracy improves with longer retained prefixes and largely saturates around 50%, approaching full-response performance.

5 Conclusion

In this paper, we study how update signals are distributed within sampled groups in GRPO-style reasoning RL, where full-group and full-response updates are costly and may reinforce verbose trajectories. Our gradient-similarity analysis shows that, among completions sampled for the same prompt, same-class completions often induce similar update directions, while correct–incorrect pairs provide more distinct contrastive signals. Motivated by this, we propose BPPO, which retains the shortest correct–incorrect pair from each group as a compact update unit and applies policy updates only to the corresponding prefixes. This reduces update-bearing completions and tokens while avoiding direct reinforcement of redundant reasoning patterns. Across GSM8K, MATH, and Geo3K, BPPO achieves up to 5.90×, 6.08×, and 3.86× training speedup over GRPO, respectively, while maintaining competitive accuracy and reducing mean response length by approximately 30–50% without adding an explicit length penalty to the reward.

Limitations

BPPO primarily accelerates training by reducing update-bearing completions and tokens, but it does not modify the generation procedure itself; thus, generation may still dominate the cost under long rollouts or expensive decoding. The prefix length is set to 50% of the average response length as a simple heuristic, which may not be optimal across tasks, models, or training stages. BPPO also assumes that completions can be reliably divided into correct and incorrect classes; extending it to preference-based, partial-credit, or multi-dimensional rewards requires defining suitable contrastive pairs. Due to computational constraints, our experiments remain limited in scale, benchmarks, and modalities.

References

Pranjal Aggarwal and Sean Welleck. 2025. L1: Controlling how long a reasoning model thinks with reinforcement learning. In *Second Conference on Language Modeling*.

Daman Arora and Andrea Zanette. 2026. Training language models to reason efficiently. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, and 45 others. 2025. Qwen3-vl technical report. *Preprint*, arXiv:2511.21631.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, and 12 others. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *Preprint*, arXiv:2204.05862.

Yuyan Bu, Liangyu Huo, Yi Jing, and Qing Yang. 2025. Beyond excess and deficiency: Adaptive length bias mitigation in reward models for RLHF. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 3091–3098, Albuquerque, New Mexico. Association for Computational Linguistics.

Weize Chen, Jiarui Yuan, Tailin Jin, Ning Ding, Huimin Chen, Zhiyuan Liu, and Maosong Sun. 2026. The overthinker's DIET: Cutting token calories with DIfficulty-aware training. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Mz Dai, Chenxu Yang, and Qingyi Si. 2026. S-GRPO: Early exit via reinforcement learning in reasoning models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Tri Dao. 2024. Flashattention-2: Faster attention with better parallelism and work partitioning. In *The Twelfth International Conference on Learning Representations*.

Tri Dao, Daniel Y Fu, Stefano Ermon, Atri Rudra, and Christopher Re. 2022. Flashattention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*.

Razvan-Gabriel Dumitru, Darius Peteleaza, Vikas Yadav, and Liangming Pan. 2025. ConciseRL: Conciseness-guided reinforcement learning for efficient reasoning models. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 17099–17123, Suzhou, China. Association for Computational Linguistics.

Gongfan Fang, Xinyin Ma, and Xinchao Wang. 2026. Thinkless: LLM learns when to think. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Wei Fu, Jiaxuan Gao, Xujie Shen, Chen Zhu, Zhiyu Mei, Chuyi He, Shusheng Xu, Guo Wei, Jun Mei, WANG JIASHU, Tongkai Yang, Binhang Yuan, and Yi Wu. 2026. AREAL: A large-scale asynchronous reinforcement learning system for language reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024. OlympiadBench: A challenging benchmark for promoting AGI with

olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850, Bangkok, Thailand. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. 2026. Thinkprune: Pruning long chain-of-thought of LLMs via reinforcement learning. *Transactions on Machine Learning Research*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, page 611–626, New York, NY, USA. Association for Computing Machinery.

Alan Lee and Harry Tong. 2025. Token-efficient RL for LLM reasoning. In *Tiny Titans: The next wave of On-Device Learning for Foundational Models (TTODLer-FM)*.

Aitor Lewkowycz, Anders Johan Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Venkatesh Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models. In *Advances in Neural Information Processing Systems*.

Wendi Li, Wei Wei, Kaihe Xu, Wenfeng Xie, Dangyang Chen, and Yu Cheng. 2024. Reinforcement learning with token-level feedback for controllable text generation. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 1704–1719, Mexico City, Mexico. Association for Computational Linguistics.

Yanhao Li, Lu Ma, Jiaran Zhang, Lexiang Tang, Wentao Zhang, and Guibo Luo. 2025. Leash: Adaptive length penalty and reward shaping for efficient large reasoning model. *Preprint*, arXiv:2512.21540.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

ZhiHang Lin, Mingbao Lin, Yuan Xie, and Rongrong Ji. 2026. CPPO: Accelerating the training of group relative policy optimization-based reasoning models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Wei Liu, Ruochen Zhou, Yiyun Deng, Yuzhen Huang, Junteng Liu, Yuntian Deng, Yizhe Zhang, and Junxian He. 2026. Learn to reason efficiently with adaptive length-based reward shaping. In *The Fourteenth International Conference on Learning Representations*.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025. Understanding r1-zero-like training: A critical perspective. In *Second Conference on Language Modeling*.

Pan Lu, Ran Gong, Shibiao Jiang, Liang Qiu, Siyuan Huang, Xiaodan Liang, and Song-Chun Zhu. 2021. Inter-GPS: Interpretable geometry problem solving with formal language and symbolic reasoning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6774–6786, Online. Association for Computational Linguistics.

Yu Meng, Mengzhou Xia, and Danqi Chen. 2024. SimPO: Simple preference optimization with a reference-free reward. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Gagan Mundada, Zihan Huang, Rohan Surana, Sheldon Yu, Jennifer Yuntong Zhang, Xintong Li, Tong Yu, Lina Yao, Jingbo Shang, Julian McAuley, and Junda Wu. 2026. Ws-grpo: Weakly-supervised group-relative policy optimization for rollout-efficient reasoning. *Preprint*, arXiv:2602.17025.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*.

Ryan Park, Rafael Rafailov, Stefano Ermon, and Chelsea Finn. 2024. Disentangling length from quality in direct preference optimization. *Preprint*, arXiv:2403.19159.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Soumya Rani Samineni, Durgesh Kalwar, Karthik Valmeekam, Kaya Stechly, and Subbarao Kambhampati. 2025. RL in name only? analyzing the structural assumptions in RL post-training for LLMs. In *First Workshop on Foundations of Reasoning in Language Models*.

Hejian Sang, Yuanda Xu, Zhengze Zhou, Ran He, and Zhipeng Wang. 2026. Not all tokens are needed(nat): token efficient reinforcement learning. *Preprint*, arXiv:2603.06619.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Wei Shen, Rui Zheng, Wenyu Zhan, Jun Zhao, Shihan Dou, Tao Gui, Qi Zhang, and Xuanjing Huang. 2023. Loose lips sink ships: Mitigating length bias in reinforcement learning from human feedback. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, EuroSys '25, page 1279–1297, New York, NY, USA. Association for Computing Machinery.

Prasann Singhal, Tanya Goyal, Jiacheng Xu, and Greg Durrett. 2024. A long way to go: Investigating length correlations in RLHF. In *First Conference on Language Modeling*.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. In *Advances in Neural Information Processing Systems*, volume 33, pages 3008–3021. Curran Associates, Inc.

Yifan Sun, Jingyan Shen, Yibin Wang, Tianyu Chen, Zhendong Wang, Mingyuan Zhou, and Huan Zhang. 2025. Improving data efficiency for llm reinforcement fine-tuning through difficulty-targeted online data selection and rollout replay. In *Advances in Neural Information Processing Systems*, volume 38, pages 173449–173476. Curran Associates, Inc.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2022. Efficient transformers: A survey. *ACM Comput. Surv.*, 55(6).

Qwen Team. 2025. Qwq-32b: Embracing the power of reinforcement learning.

Luong Trung, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. 2024. ReFT: Reasoning with reinforced fine-tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7601–7614, Bangkok, Thailand. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Violet Xiang, Chase Blagden, Rafael Rafailov, Nathan Lile, Sang T. Truong, Chelsea Finn, and Nick Haber. 2025. Just enough thinking: Efficient reasoning with adaptive length penalties reinforcement learning. In *NeurIPS 2025 Workshop on Efficient Reasoning*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, YuYue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, and 17 others. 2026. DAPO: An open-source LLM reinforcement learning system at scale. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

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

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. 2025. Group sequence policy optimization. *Preprint*, arXiv:2507.18071.

Haizhong Zheng, Yang Zhou, Brian R. Bartoldson, Bhavya Kailkhura, Fan Lai, Jiawei Zhao, and Beidi Chen. 2026. Act only when it pays: Efficient reinforcement learning for LLM reasoning via selective rollouts. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

A Appendix

Algorithm 1 BPPO Training

Input: dataset $\mathcal{D}$, size N, initial policy π_{θ_0}, group size G, target budget C_{tar}, prefix length n
Policy model $\pi_\theta \leftarrow \pi_{\theta_0}$
Reference model $\pi_{\text{ref}} \leftarrow \pi_\theta$
Set $B_{\text{sch}} \leftarrow \lceil C_{\text{tar}}/2 \rceil$, $T \leftarrow \lceil N/B_{\text{sch}} \rceil$
for step $= 1, \ldots, T$ **do**
  Update old policy $\pi_{\theta_{\text{old}}} \leftarrow \pi_\theta$
  # Adaptive completion scheduling
  Sample scheduled prompt batch $\mathcal{D}_b$ of B_{sch} prompts
  Initialize update batch $\mathcal{B} \leftarrow \emptyset$
  # Pair selection
  for each prompt $q^{(j)} \in \mathcal{D}_b$ **do**
    Sample completions $\{o_i^{(j)}\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(\cdot \mid q^{(j)})$
    Compute rewards $\{r_i\}_{i=1}^G$ and advantages $\{\hat{A}_i\}_{i=1}^G$
    $\mathcal{I}_j^+ = \{i : r_i > 0\}$, $\mathcal{I}_j^- = \{i : r_i \leq 0\}$
    if $\mathcal{I}_j^+ = \emptyset$ or $\mathcal{I}_j^- = \emptyset$ **then**
      continue
    end if
    $i_+^* \leftarrow \text{argmin}_{i \in \mathcal{I}_j^+} |o_i^{(j)}|$
    $i_-^* \leftarrow \text{argmin}_{i \in \mathcal{I}_j^-} |o_i^{(j)}|$
    $\mathcal{P}_j \leftarrow \{i_+^*, i_-^*\}$
    Add selected pair $\{(q^{(j)}, o_i^{(j)}, \hat{A}_i) : i \in \mathcal{P}_j\}$ to $\mathcal{B}$
  end for
  # Prefix-focused update
  Keep the first n response tokens for each completion in $\mathcal{B}$
  Update θ with the BPPO objective on $\mathcal{B}$
end for
Output: π_θ

A.1 Algorithm

Algorithm 1 summarizes the BPPO training procedure. At each step, BPPO samples a scheduled prompt batch $\mathcal{D}_b$ so that the retained update-bearing completions match C_{tar}. For each prompt, it samples G completions, computes full-group rewards and advantages, retains the shortest correct–incorrect pair, and discards all-correct or all-incorrect groups before importance-ratio computation. The retained pairs form $\mathcal{B}$, aggregating completions from more prompts and reducing optimization steps per epoch. Policy updates use only the first n response tokens, while full completions are used for reward and advantage computation.

B Experiments

B.1 Additional Results on a Larger Model

As an additional scalability check, we evaluate BPPO on a larger model, Qwen3-14B, and report the results in Table 7. The results show that BPPO also remains effective at a larger scale, achieving a 5.22× speedup over GRPO while maintaining competitive accuracy.

Table 7: Results on GSM8K with Qwen3-14B. BPPO achieves comparable accuracy to GRPO with substantially lower training time.

Method	Accuracy (%)	Training Time (s)	Speedup (times)
Qwen3-14B	46.55	–	–
GRPO	93.33	36598	1.00
BPPO	93.86	7006	5.22

Optimization Stability and Convergence. To assess stability at a larger scale, we train both GRPO and BPPO for 10 epochs and plot their reward and response-length curves in Figures 6 and 7. BPPO shows convergence behavior comparable to GRPO without signs of instability or reward collapse. Its response length also remains stable and substantially lower than that of GRPO throughout training, indicating that BPPO maintains stable optimization behavior during large-model training.

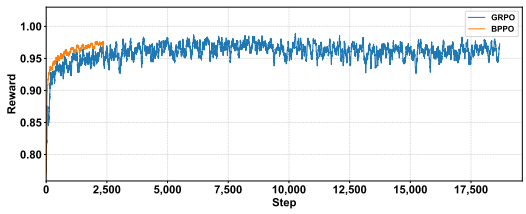


Figure 6: Reward curves of GRPO and BPPO on GSM8K with Qwen3-14B over 10 training epochs.

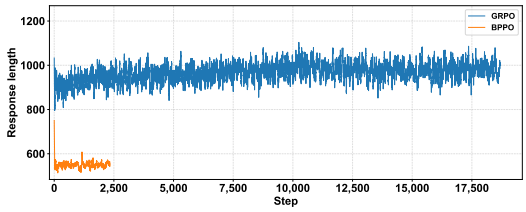


Figure 7: Response-length curves of GRPO and BPPO on GSM8K with Qwen3-14B over 10 training epochs.

Table 8: Additional $G = 32$ results on GSM8K, MATH, and Geo3K. Each setting is trained three times independently, and we report mean $\pm$ standard deviation. BPPO achieves the largest speedup and response-length reduction across all three benchmarks, with slight accuracy drops in some settings.

Benchmark Model	Method	Accuracy (%)	Mean Response Tokens	Length Reduction (%)	Training Time (s)	Speedup (times)
GSM8K Qwen2.5-1.5B -Instruct	GRPO	76.96 ± 0.35	265.08 ± 2.58	-	23519 ± 4033	1.00
	CPPO	73.64 ± 0.72	232.62 ± 3.08	12.2	3587 ± 552	6.56 ± 1.01
	GRPO+FIRSTN	76.85 ± 0.28	237.77 ± 6.93	10.3	20205 ± 860	1.16 ± 0.18
	Pair	74.44 ± 0.21	144.77 ± 3.71	45.4	3196 ± 256	7.36 ± 0.59
	BPPO	73.84 ± 1.05	122.39 ± 5.46	**53.8**	2672 ± 51	$\mathbf{8.80} \pm 0.17$
MATH Llama3.2-3B -Instruct	GRPO	46.07 ± 0.43	403.94 ± 5.93	-	17763 ± 1776	1.00
	CPPO	45.28 ± 0.15	359.80 ± 14.21	10.9	2553 ± 180	6.96 ± 0.49
	GRPO+FIRSTN	46.74 ± 0.17	371.49 ± 6.07	8.0	15519 ± 187	1.14 ± 0.12
	Pair	45.02 ± 0.30	163.15 ± 0.94	59.6	2382 ± 162	7.46 ± 0.51
	BPPO	44.76 ± 0.18	154.10 ± 2.75	**61.9**	2041 ± 26	$\mathbf{8.70} \pm 0.11$
Geo3K Qwen3-VL-4B	GRPO	49.34 ± 0.25	3001.24 ± 29.15	-	39447 ± 177	1.00
	CPPO	49.56 ± 0.68	4005.67 ± 42.29	-33.5	22162 ± 253	1.78 ± 0.02
	GRPO+FIRSTN	48.89 ± 0.19	3025.72 ± 15.25	-0.8	36419 ± 73	1.08 ± 0.01
	Pair	49.73 ± 0.08	1618.41 ± 45.07	46.1	20403 ± 541	1.93 ± 0.05
	BPPO	49.85 ± 0.86	1438.18 ± 50.32	**52.1**	18952 ± 376	$\mathbf{2.08} \pm 0.04$

Table 9: Results without adaptive completion scheduling under rollout group size $G = 32$. BPPO still achieves accuracy comparable GRPO while updating only the prefixes of two selected completions from each $G = 32$ group.

Benchmark Model	Method	Accuracy (%)	Mean Response Tokens	Length Reduction (%)	Training Time (s)	Speedup (times)
GSM8K Qwen2.5-1.5B -Instruct	GRPO	76.96	265.08	-	23519	1.00
	GRPO+FIRSTN	76.85	237.77	10.3	20205	1.16
	Pair w/o ACS	77.86	175.28	33.9	4506	5.22
	BPPO w/o ACS	77.48	164.31	**38.0**	4201	**5.60**
MATH Llama3.2-3B -Instruct	GRPO	46.07	403.94	-	17763	1.00
	GRPO+FIRSTN	46.74	371.49	8.0	15519	1.14
	Pair w/o ACS	48.62	236.03	41.6	10948	1.62
	BPPO w/o ACS	48.16	214.14	**47.0**	10092	**1.76**
Geo3K Qwen3-VL-4B	GRPO	49.34	3001.24	-	39447	1.00
	GRPO+FIRSTN	48.89	3025.72	-0.8	36419	1.08
	Pair w/o ACS	49.27	1853.20	38.3	28394	1.39
	BPPO w/o ACS	50.16	1548.00	**48.4**	25354	**1.56**

B.2 Additional Results for Larger Rollout Groups

We further report results under a larger rollout group size $G = 32$ in Table 8. Larger rollout groups provide more candidates for shortest-pair selection, allowing BPPO to produce shorter responses. On text-only benchmarks, this also leads to stronger training speedups. BPPO consistently achieves the shortest responses, but also shows minor accuracy drops in some settings.

B.3 Effect of Adaptive Completion Scheduling

Adaptive completion scheduling reduces the number of optimization steps needed to process one training epoch. To isolate the effect of this scheduling component in a controlled comparison, we directly evaluate Pair and BPPO without adaptive completion scheduling, keeping the number of optimization steps matched with GRPO for the same training epoch under identical training settings. As shown in Table 9, Pair and BPPO remain comparable to full-group GRPO in accuracy, while reducing response length and training time. These results show that, at $G = 32$, updating only the prefixes of the shortest correct–incorrect pair can still achieve accuracy comparable to full-group GRPO updates. Table 10 further reports the corresponding matched-step results for $G = 8$ and $G = 16$.

Table 10: Results without adaptive completion scheduling under rollout group sizes $G = 8$ and $G = 16$. Pair and BPPO reduce response length and training time.

Method	Benchmark Model	Group Size G	Accuracy (%)	Mean Response Tokens	Length Reduction (%)	Training Time (s)	Speedup (times)
Pair w/o ACS	GSM8K Qwen2.5-1.5B -Instruct	8	76.82	180.25	30.4	3388	2.12
BPPO w/o ACS			76.50	129.61	**50.0**	3147	**2.28**
Pair w/o ACS		16	77.33	192.07	26.9	3942	3.00
BPPO w/o ACS			76.12	125.84	**52.1**	3602	**3.28**
Pair w/o ACS	MATH Llama3.2-3B -Instruct	8	46.08	253.84	29.6	8132	1.12
BPPO w/o ACS			45.84	234.85	**34.9**	7820	**1.16**
Pair w/o ACS		16	47.78	231.00	38.4	8692	1.45
BPPO w/o ACS			46.20	224.79	**40.0**	8382	**1.51**
Pair w/o ACS	Geo3K Qwen3-VL-4B	8	49.72	2333.00	23.5	22152	1.13
BPPO w/o ACS			48.22	2119.56	**30.5**	21085	**1.19**
Pair w/o ACS		16	48.51	1916.98	37.5	22809	1.28
BPPO w/o ACS			51.94	1822.16	**40.6**	21665	**1.35**

Table 11: Compatibility of BPPO with DAPO and GSPO on GSM8K and MATH. BPPO provides consistent efficiency gains across both methods.

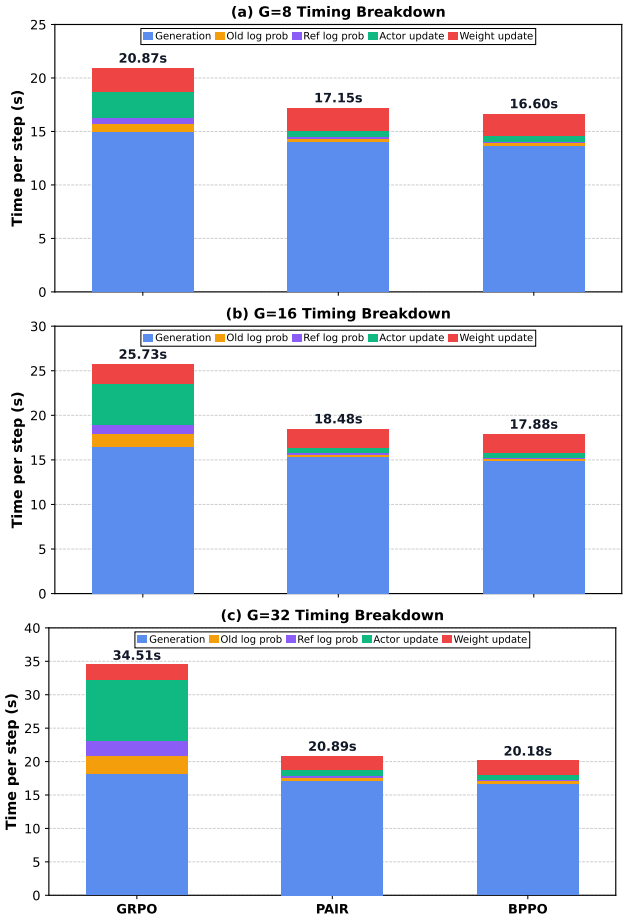
Model	Benchmark	Method	Accuracy (%)	Training Time (s)	Speedup (times)
Qwen3-8B	GSM8K	Baseline	32.90	–	–
		DAPO	95.15	12991	1.00
		DAPO+FIRSTN	94.16	11573	1.12
		DAPO+Pair	95.38	5187	2.51
		DAPO+BPPO	95.07	4441	**2.92**
	MATH	Baseline	60.64	–	–
		DAPO	87.13	42062	1.00
		DAPO+FIRSTN	87.04	38506	1.09
		DAPO+Pair	86.86	22225	1.89
		DAPO+BPPO	86.74	20430	**2.06**
DeepSeek-R1-Distill Qwen-7B	GSM8K	Baseline	27.41	–	–
		GSPO	92.72	15287	1.00
		GSPO+FIRSTN	93.18	13463	1.14
		GSPO+Pair	92.34	3093	4.94
		GSPO+BPPO	91.99	2669	**5.73**
	MATH	Baseline	75.38	–	–
		GSPO	83.08	37738	1.00
		GSPO+FIRSTN	82.14	34333	1.10
		GSPO+Pair	82.64	13475	2.80
		GSPO+BPPO	81.18	12581	**3.00**

B.4 Additional Compatibility Results

Table 11 provides the complete compatibility results for DAPO and GSPO. Together with the representative results in Table 4, these results show that the proposed pair-selection and prefix-focused update strategy is not tied to a specific policy objective. Instead, it can be combined with different reasoning-RL methods, consistently reducing training time across multiple reasoning benchmarks while largely preserving the accuracy of the original methods.

Table 12: Evaluation on additional mathematical reasoning benchmarks using Llama3.2-3B-Instruct after training on MATH for one epoch. We compare GRPO and BPPO under rollout group sizes $G = 8, 16, 32$. BPPO achieves slightly higher average performance across the evaluated group sizes.

Method	Group Size G	AIME24	MinervaMATH	OlympiadBench	Average
GRPO	8	13.33	13.24	10.83	12.47
BPPO		13.33	13.24	11.42	**12.66**
GRPO	16	6.67	14.71	10.36	10.58
BPPO		10.00	13.24	11.57	**11.60**
GRPO	32	3.33	12.50	11.42	9.08
BPPO		6.67	12.13	11.72	**10.17**



$G = 8$			
Stage	GRPO	PAIR	BPPO
Generation	14.95	14.03	13.72
Old log prob	0.79	0.25	0.21
Ref log prob	0.59	0.17	0.13
Actor update	2.38	0.64	0.50
Weight update	2.17	2.06	2.04
Total	20.87	17.15	16.60

$G = 16$			
Stage	GRPO	PAIR	BPPO
Generation	16.47	15.38	14.94
Old log prob	1.38	0.24	0.19
Ref log prob	1.12	0.17	0.13
Actor update	4.55	0.63	0.49
Weight update	2.21	2.06	2.13
Total	25.73	18.48	17.88

$G = 32$			
Stage	GRPO	PAIR	BPPO
Generation	18.19	17.12	16.72
Old log prob	2.69	0.44	0.34
Ref log prob	2.26	0.30	0.23
Actor update	9.11	0.95	0.71
Weight update	2.26	2.09	2.18
Total	34.51	20.89	20.18

Figure 8: Per-step wall-clock timing breakdown under different rollout group sizes.

B.5 Results on Additional Math Benchmarks

We further evaluate Llama3.2-3B-Instruct models trained on MATH for one epoch on additional mathematical reasoning benchmarks, including AIME24, MinervaMATH, and OlympiadBench. As shown in Table 12, BPPO remains comparable to GRPO and achieves slightly higher performance across the evaluated rollout group sizes. These evaluations further support that shortest-pair prefix updates can match full-group GRPO performance.

B.6 Wall-clock Timing Breakdown

We report the per-step wall-clock timing breakdown of Llama-3.2-3B-Instruct on MATH under different rollout group sizes, as shown in Figure 8. Adaptive Completion Scheduling is disabled in this analysis to isolate the effect of pair selection and prefix truncation. The breakdown shows that BPPO substantially reduces the actor-update time across all rollout group sizes, which is the main source of per-step efficiency gains.

C Case Study

We provide case studies from GSM8K, MATH, and Geo3K to illustrate the qualitative differences between responses generated by GRPO and BPPO. On GSM8K, as shown in Figure 9, BPPO directly computes the daily total and scales it to a week, avoiding unnecessary unit conversions and intermediate explanations. On MATH, as shown in Figure 10, BPPO captures the independent-choice structure directly, whereas GRPO enumerates all binomial cases, illustrating that a shorter path to the right answer can be the smarter choice. On Geo3K, as shown in Figure 11, BPPO focuses on the essential Pythagorean relation and gives a more compact derivation. Together, these examples show that BPPO preserves correctness while removing redundant reasoning steps.

Question. John takes care of 10 dogs. Each dog takes 0.5 hours a day to walk and take care of their business. How many hours a week does he spend taking care of dogs?

Ground Truth. 35

GRPO Response ✓

To determine how many hours John spends taking care of dogs each week, we can break down the problem into smaller steps:
1. Calculate the daily time spent on one dog:
- Each dog takes 0.5 hours per day.
2. Calculate the total daily time for all dogs:
- John takes care of 10 dogs.
- Total daily time = Number of dogs $\times$ Time per dog
- Total daily time = 10 $\times$ 0.5 hours
3. Convert the daily time from hours to minutes (optional but useful):
- There are 60 minutes in an hour.
- Convert 0.5 hours to minutes:

$$0.5 \text{ hours} \times 60 \text{ minutes/hour} = 30 \text{ minutes}$$

- So, each dog takes 30 minutes per day.
4. Calculate the weekly time spent on all dogs:
- There are 7 days in a week.
- Total weekly time = Daily time $\times$ Days per week
- Total weekly time = (10 dogs) $\times$ (30 minutes/dog/day) $\times$ (7 days/week)
5. Perform the multiplication:
- First, multiply the number of dogs by the time per dog per day:

$$10 \times 30 = 300 \text{ minutes}$$

- Then, multiply this result by the number of days per week:

$$300 \text{ minutes/day} \times 7 \text{ days/week} = 2100 \text{ minutes/week}$$

6. Convert the weekly time back to hours:
- There are 60 minutes in an hour.
- Convert 2100 minutes to hours:

$$2100 \text{ minutes} \div 60 \text{ minutes/hour} = 35 \text{ hours}$$

Therefore, the final answer is:

$$\boxed{35}.$$

BPPO Response ✓

To determine how many hours John spends taking care of dogs each week, we can break down the problem into smaller steps:
1. Time per dog per day = 0.5 hours.
2. Total time per day for 10 dogs:

$$0.5 \times 10 = 5 \text{ hours}.$$

3. Weekly time:

$$5 \times 7 = 35 \text{ hours}.$$

Therefore, the final answer is:

$$\boxed{35}.$$

John spends 35 hours a week taking care of his dogs.

Figure 9: Case study on GSM8K. Both GRPO and BPPO generate the correct answer, while BPPO produces a more concise and direct reasoning trajectory.

Question. A group of 10 Caltech students go to Lake Street for lunch. Each student eats at either Chipotle or Panda Express. In how many different ways can the students collectively go to lunch?

Ground Truth. 1024

GRPO Response ✓

To find the total number of ways the students can collectively go to lunch, we need to consider all possible combinations of students eating at Chipotle and Panda Express.
Let's use a variable to represent the number of students eating at Chipotle. We can let x be the number of students eating at Chipotle and $10 - x$ be the number of students eating at Panda Express.
Since each student eats at either Chipotle or Panda Express, the total number of ways the students can collectively go to lunch is the sum of the number of ways for each possible value of x.
We can use the binomial coefficient formula to calculate the number of ways for each value of x:

$$
\begin{aligned}
x = 0 &: 1, \\
x = 1 &: 10, \\
x = 2 &: \binom{10}{2} = 45, \\
x = 3 &: \binom{10}{3} = 120, \\
x = 4 &: \binom{10}{4} = 210, \\
x = 5 &: \binom{10}{5} = 252, \\
x = 6 &: \binom{10}{6} = 210, \\
x = 7 &: \binom{10}{7} = 120, \\
x = 8 &: \binom{10}{8} = 45, \\
x = 9 &: \binom{10}{9} = 10, \\
x = 10 &: 1.
\end{aligned}
$$

Now, we can sum up the number of ways for each value of x:
1+10+45+120+210+252+210+120+45+10+1=1024.
Therefore, the total number of ways the students can collectively go to lunch is 1024.
The final answer is

$$\boxed{1024}.$$

BPPO Response ✓

To find the total number of ways the students can go to lunch, we can consider each student's choice as an independent event.
Each student has 2 choices: Chipotle or Panda Express.
Since there are 10 students, the total number of ways the students can collectively go to lunch is given by the product of the number of choices for each student, which is 2^{10}.
Therefore, the total number of ways the students can go to lunch is

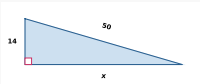
$$2^{10} = \boxed{1024}.$$

The final answer is:

$$\boxed{1024}.$$

Figure 10: Case study on MATH. Both GRPO and BPPO generate the correct answer, while BPPO directly uses the fact that each student makes an independent binary choice, avoiding unnecessary enumeration.

Question. Find x.



Ground Truth. 48

GRPO Response ✓

I need to find x in this right triangle. The triangle has a right angle, with one leg labeled 14, the hypotenuse labeled 50, and the other leg labeled x. This is a classic right triangle problem where I can use the Pythagorean theorem.
The Pythagorean theorem states that in a right triangle, the square of the hypotenuse, the side opposite the right angle, is equal to the sum of the squares of the other two sides. So, if I let the hypotenuse be c, and the legs be a and b, then

$$a^2 + b^2 = c^2.$$

In this case, the hypotenuse is 50, one leg is 14, and the other leg is x. So, I can write the equation:

$$14^2 + x^2 = 50^2.$$

Now I'll calculate the squares:

$$196 + x^2 = 2500.$$

Subtract 196 from both sides:

$$x^2 = 2500 - 196,$$
$$x^2 = 2304.$$

Now take the square root of both sides:

$$x = \sqrt{2304}.$$

I know that $48^2 = 2304$, because $50^2 = 2500$ and

$$48^2 = (50-2)^2 = 2500 - 200 + 4 = 2304.$$

So, $x = 48$.
Therefore, the value of x is 48.
I should note that since we're dealing with a length, x must be positive, so we take the positive square root.
Final answer: 48.
Final Answer

$$\boxed{48}.$$

BPPO Response ✓

I need to find x in this right triangle. I see a right angle marked at the bottom left, with the side opposite the right angle labeled 14, and the hypotenuse labeled 50. The side adjacent to the right angle is labeled x.
This is a right triangle, so I can use the Pythagorean theorem:

$$a^2 + b^2 = c^2,$$

where c is the hypotenuse.
Here, the hypotenuse is 50, one leg is 14, and the other leg is x. So:

$$14^2 + x^2 = 50^2,$$
$$196 + x^2 = 2500,$$
$$x^2 = 2500 - 196,$$
$$x^2 = 2304,$$
$$x = \sqrt{2304},$$
$$x = 48.$$

I can verify:

$$14^2 = 196, \quad 48^2 = 2304, \quad 196 + 2304 = 2500.$$

Perfect.
Final Answer

$$\boxed{48}.$$

Figure 11: Case study on Geo3K. Both GRPO and BPPO generate the correct answer on a multimodal geometry problem, while BPPO provides a more compact derivation.