

OPE: Overcoming Information Saturation in Parallel Thinking via Outline-Guided Path Exploration

Qi Guo[♠], Jianing Wang[♠], Deyang Kong[♠], Xiangyu Xi[♠], Jianfei Zhang[♠],
Yi Lu[♠], Jingang Wang[♠], Wei Wang[♠], Shikun Zhang[♠], Wei Ye^{♠*},

[♠]National Engineering Research Center for Software Engineering, Peking University

[♠]Meituan LongCat Team

qguo@stu.pku.edu.cn wye@pku.edu.cn

Abstract

Parallel thinking has emerged as a new paradigm for large reasoning models (LRMs) in tackling complex problems. Recent methods leverage Reinforcement Learning (RL) to enhance parallel thinking, aiming to address the limitations in computational resources and effectiveness encountered with supervised fine-tuning. However, most existing studies primarily focus on optimizing the aggregation phase, with limited attention to the path exploration stage. In this paper, we theoretically analyze the optimization of parallel thinking under the Reinforcement Learning with Verifiable Rewards (RLVR) setting, and identify that the mutual information bottleneck among exploration paths fundamentally restricts overall performance. To address this, we propose Outline-Guided Path Exploration (OPE), which explicitly partitions the solution space by generating diverse reasoning outlines prior to parallel path reasoning, thereby reducing information redundancy and improving the diversity of information captured across exploration paths. OPE are implemented with an iterative RL strategy that optimizes outline planning and outline-guided reasoning independently. Extensive experiments across multiple challenging mathematical benchmarks demonstrate that OPE effectively improves reasoning performance in different aggregation strategies, enabling LRMs to more reliably discover correct solutions.

1 Introduction

Despite the rapid evolution of Large Reasoning Models (LRMs) (Guo et al., 2025; Team et al., 2025; Yang et al., 2025a), their capacity to solve complex problems in a single inference pass remains constrained. Following the success of Gemini (Luong et al., 2025) in the International Mathematical Olympiad competitions, parallel thinking has emerged as a promising reasoning paradigm

for addressing such challenges. Parallel thinking allows the model to concurrently explore multiple potential solution paths, subsequently aggregating the outcomes of these explorations to synthesize the final answer. Such a paradigm encourages the model to broadly search for potential correct trajectories within the solution space, while simultaneously mitigating the risk of biased outcomes associated with a single exploration trajectory (Pan et al., 2025; Zheng et al., 2025b).

Prior approaches rely on Supervised Fine-Tuning (SFT) over extensive corpus of high-quality parallel reasoning trajectories (Wen et al., 2025; Yang et al., 2025b; Ning et al., 2023). This process not only incurs heavy resource consumption for trajectory synthesis but also risks superficial pattern imitation, thereby limiting effectiveness on challenging tasks. Consequently, recent studies have shifted toward utilizing RL to empower models to autonomously acquire parallel thinking under practical problem-solving scenarios (Zheng et al., 2025b). However, the application of RL in parallel thinking remains underexplored, with existing works predominantly concentrating on the aggregation phase over the exploration phase. Crucially, prior studies indicate that the effectiveness of the aggregation phase depends heavily on the quality of the paths found during exploration, which sets a natural limit on overall performance of parallel thinking (Wang et al., 2025).

In this paper, we investigate the impact of the path exploration stage on the performance of parallel thinking. We present a theoretical analysis within the Reinforcement Learning with Verifiable Rewards (RLVR), and reveal that overall performance is fundamentally constrained by mutual information saturation among explored paths. Specifically, under the naive paradigm where paths are sampled independently, optimizing the information gain between trajectories becomes challenging. In addition, since LRMs tend to suffer from mode

*Corresponding authors.

collapse, the generated paths often exhibit high redundancy (Shumailov et al., 2023; Shao et al., 2024). Consequently, when tackling complex problems, multiple parallel paths often converge on the same incorrect answers rather than covering diverse regions of the solution space. Based on these findings, we propose Outline-guided Parallel Exploration (OPE), a framework that requires the model to explicitly partition the solution space using outlines before path reasoning, thereby maximizing the mutual information between generated paths and the correct solution. We start from a cold-start stage to instill outline planning capabilities, where the model learns to analyze problems and generate outlines for diverse reasoning strategies. Building on this, we introduce an iterative RL strategy to jointly optimize outline planning and outline-guided reasoning capabilities. Specifically, we first conduct Outline Planning RL to encourage the model to generate diverse outlines that define multiple distinct problem-solving directions. This is followed by Path Reasoning RL, which refines the model’s ability to execute reasoning steps under the guidance of these outlines. These two training phases are interleaved at fixed step intervals to facilitate the synergistic evolution of both capabilities.

Experimental results across six widely-used mathematical benchmarks demonstrate that OPE leads to significant improvements in parallel thinking performance, with particularly substantial gains on the most challenging tasks. Further analysis shows that, compared to naive parallel thinking which samples each path independently, OPE exhibits superior scaling properties at test time and alleviates the issue of “overthinking” during path reasoning, enabling the model to reach solutions more efficiently through targeted exploration.

Our contributions can be summarized as:

- We provide a theoretical perspective on parallel thinking RL, revealing exploration-phase mutual information saturation as the fundamental performance bottleneck.
- We introduce OPE, a novel paradigm designed to mitigate mutual information saturation by explicitly partitioning the solution space with outlines. We implement OPE via an iterative RL strategy that synergistically optimizes outline planning and path reasoning capabilities.
- Extensive experiments demonstrate that OPE achieves superior performance across complex benchmarks. Further analysis reveals that

OPE exhibits better test-time scaling properties and effectively alleviates the “overthinking” issue, establishing a solid baseline and offering new perspectives for future research.

2 Related Work

2.1 Parallel Thinking

Parallel thinking typically generates and aggregates multiple reasoning trajectories. Most research focuses on the aggregation phase, primarily through two streams: (1) **Selection and Ranking** methods, which identifies the optimal path using Verifiers, Reward Models, or pairwise ranking (Cobbe et al., 2021; Lightman et al., 2023; Li et al., 2023); and (2) **Summarization** methods, which synthesizes a final answer via heuristics like Self-Consistency (Wang et al., 2022a) or LRM-based summarization (Jiang et al., 2023). Conversely, the trajectory exploration phase remains under-explored. Existing methods like Skeleton of Thought (Ning et al., 2023) and Leap (Luo et al., 2025b) improve efficiency via sub-task decomposition or fixed information exchange, but rely on complex architectural designs. Distinctly, OPE enables models to explicitly determine the exploration distribution to identify potential solutions.

2.2 RLVR

RLVR leverages verifiable outcome signals, such as rule-based answer matching or model-based evaluation, to guide model optimization, thereby circumventing the reliance on human-annotated rewards. This paradigm has demonstrated remarkable efficacy in enhancing the reasoning capabilities of Large Language Models (LLMs) (Trinh et al., 2024; Shao et al., 2024). Among RLVR algorithms, Group Relative Policy Optimization (GRPO) (Shao et al., 2024) eliminates the need for a separate value network and reduces training instability using a group-relative baseline, demonstrating significant advantages in mathematical reasoning tasks. Specifically, for a group of G outputs $\{o_1, \dots, o_G\}$ sampled for a query q , GRPO estimates the advantage of each output o_i as $A_i = (r_i - \text{mean}(\mathbf{r}))/\text{std}(\mathbf{r})$, where $\mathbf{r}$ is the reward vector. Building on this, several variants further refine sample efficiency and stability (Yue et al., 2025; Zheng et al., 2025a; Yu et al., 2025). Despite extensive exploration of RL in reasoning scenarios, its application specifically to parallel thinking remains underexplored. This paper bridges this gap by pro-

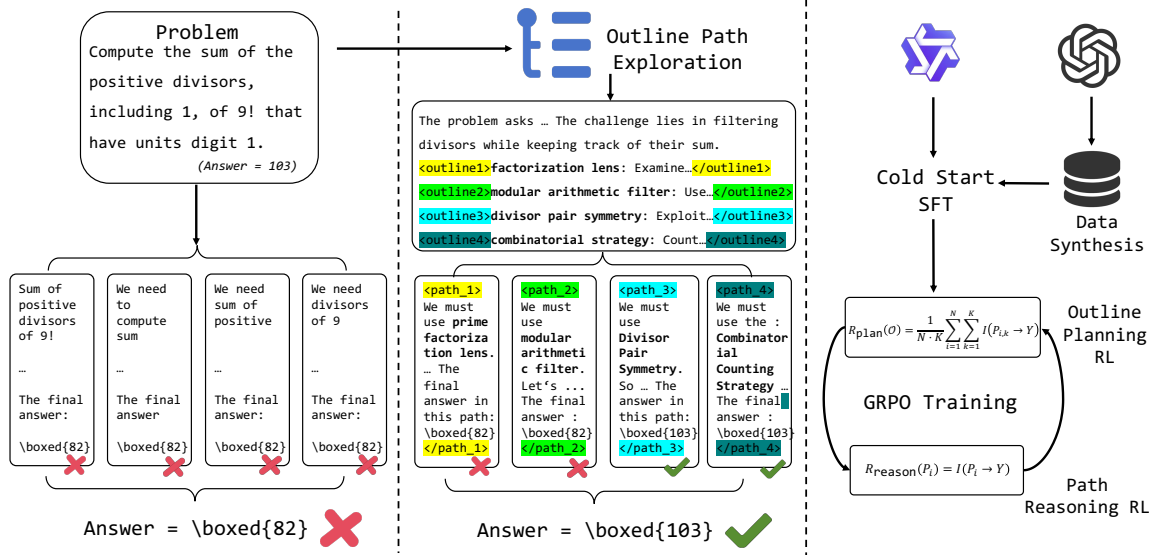


Figure 1: **Overview of OPE framework.** (Left) Naive parallel thinking samples reasoning paths independently. Due to mode collapse, these paths often exhibit high redundancy and tend to converge on the same incorrect answer. (Middle) OPE mitigates this by explicitly generating diverse outlines to partition the solution space into distinct directions (four different strategies). This structured exploration maximizes the coverage of potential solutions, enabling the model to successfully locate the correct reasoning trajectory. (Right) The OPE training pipeline consists of a *Cold Start* phase using synthesized data, followed by a novel *Iterative RL* strategy.

viding a theoretical analysis of parallel thinking and offering empirical insights for future research.

3 Parallel Thinking with RLVR

In this section, we present a theoretical analysis of the parallel thinking paradigm. We focus on the RLVR settings and identify that Mutual Information Saturation among generated paths during the exploration phase as a fundamental bottleneck that constrains the reasoning performance.

3.1 Formulation of Parallel Thinking

Let Q denote the input query drawn from the data distribution $\mathcal{D}$, and Y represent the corresponding ground-truth Answer. We define A as the answer generated by a unified policy model π_θ . The reasoning workflow in parallel thinking can be formally decomposed into two distinct phases:

1.Exploration: Given Q , the model samples a set of N parallel reasoning paths $\mathcal{P} = \{P_1, \dots, P_N\}$:

$$P_i \sim \pi_\theta(\cdot|Q), \quad \forall i \in \{1, \dots, N\} \quad (1)$$

2.Aggregation: Based on Q and explored paths $\mathcal{P}$, the model synthesizes the final answer A :

$$A \sim \pi_\theta(\cdot|\mathcal{P}, Q) \quad (2)$$

We formulate $\mathcal{P}$ as intermediate variables that bridge the query and the final answer. Consequently, the overall inference probability is derived

by marginalizing over the space of generated paths:

$$\pi_\theta(A|Q) = \sum_{\mathcal{P}} \pi_\theta(A|\mathcal{P}, Q) \pi_\theta(\mathcal{P}|Q) \quad (3)$$

3.2 Parallel Thinking Optimization

The fundamental objective of RLVR is to optimize the policy parameters θ to maximize the expected reward over the data distribution $\mathcal{D}$:

$$J(\theta) = \mathbb{E}_{Q \sim \mathcal{D}} [\mathbb{E}_{A \sim \pi_\theta(\cdot|Q)} [r(Q, A)]] \quad (4)$$

where $r(Q, A)$ serves as the reward function. In RLVR setting, this reward is usually deterministic and binary (i.e., $r = 1$ if the generated answer A matches the ground truth Y , and 0 otherwise). Consequently, maximizing $J(\theta)$ is asymptotically equivalent to maximizing the log-likelihood of generating the ground truth Y given Q :

$$\max_{\theta} J(\theta) \iff \max_{\theta} \log \pi_\theta(Y|Q) \quad (5)$$

Since the generation of Y depends on the intermediate reasoning paths $\mathcal{P}$, the log-likelihood can be expressed as $\log \sum_{\mathcal{P}} \pi_\theta(Y|\mathcal{P}, Q) \pi_\theta(\mathcal{P}|Q)$. Since directly optimizing the resulting log-sum is intractable, a standard practice consider its Evidence Lower Bound (ELBO):

$$\log \pi_\theta(Y|Q) \geq \mathbb{E}_{\mathcal{P} \sim \pi_\theta(\cdot|Q)} [\log \pi_\theta(Y|\mathcal{P}, Q)] \quad (6)$$

From the perspective of Information Theory (Reza, 1994), the expectation term

$\mathbb{E}_{\mathcal{P}}[\log \pi_{\theta}(Y|\mathcal{P}, Q)]$ corresponds to the negative Conditional Entropy, denoted as $-H(Y|\mathcal{P}, Q)$. Recall the definition of Mutual Information (MI):

$$I(\mathcal{P}; Y|Q) = H(Y|Q) - H(Y|\mathcal{P}, Q) \quad (7)$$

Since the ground truth Y is deterministic for a given query Q in the RLVR setting, the entropy $H(Y|Q)$ remains constant (Y is not a random variable given Q). Consequently, maximizing the lower bound $\mathcal{L}_{\text{ELBO}}$ is mathematically equivalent to maximizing the mutual information:

$$\max_{\theta} \mathcal{L}_{\text{ELBO}} \iff \max_{\theta} I(\mathcal{P}; Y|Q) \quad (8)$$

This derivation reveals that the optimization of parallel thinking is fundamentally to generate reasoning paths $\mathcal{P}$ that maximize the information gain regarding the ground-truth.

3.3 MI Saturation among Paths

Instead of analyzing the mutual information between the set $\mathcal{P}$ and Y directly, we can simplify the formulation by decomposing the total mutual information item into the sum of marginal contributions from each individual path using the chain rule:

$$I(\mathcal{P}; Y|Q) = \sum_{i=1}^N I(P_i; Y|P_{1:i-1}, Q) \quad (9)$$

In the naive parallel thinking paradigm described in Section 3.1, paths are sampled independently and identically (i.i.d.). However, since LRMs typically undergo extensive post-training, they often exhibit the risk of mode collapse, which leads to significant semantic redundancy among sampled trajectories. Consequently, the marginal information gain diminishes rapidly:

$$I(P_i; Y|P_{1:i-1}, Q) \rightarrow 0 \quad \text{as } i \rightarrow N \quad (10)$$

We term this phenomenon **Mutual Information Saturation**. Intuitively, this implies that when tackling complex problems, the model tends to fall into repetitive failure modes, where a large number of parallel paths converge to several incorrect answers rather than exploring diverse regions of the solution space. As a result, simply scaling the number of paths N yields diminishing returns.

To empirically validate this phenomenon, we conduct experiments on the challenging mathematical benchmark HMMT-25 (Balunović et al., 2025). Specifically, we perform $N = 256$ independent samplings for each query using DeepSeek-R1-Distill-Qwen-7B (Guo et al., 2025) and analyze the

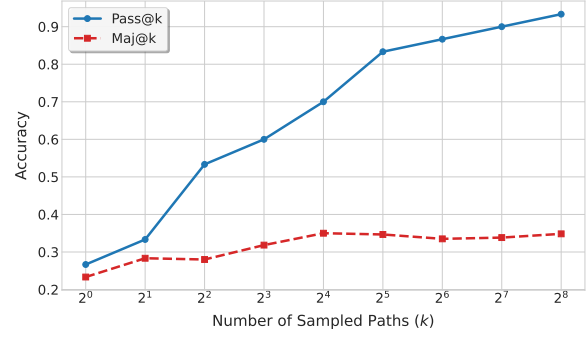


Figure 2: Comparisons of Pass@k curves and Maj@k curves on the HMMT-25 benchmark.

scaling trends of two key metrics: Pass@k, which measures the probability that at least one path in k samples is correct, and Maj@k, which represents the accuracy of the answer derived from majority voting over k samples. The results showed in Figure 2 reveal a critical discrepancy: while Pass@k continues to rise as the sample size increases, Maj@k plateaus after approximately 20 samples. This indicates that although the model possesses the intrinsic capability to solve the problems (evidenced by the rising Pass@k), the vast majority of sampled paths converge to incorrect answers. Consequently, the correct signal is drowned out by redundant errors, thereby imposing a hard ceiling on the performance of aggregation.

4 Methodology

To mitigate the bottleneck of Mutual Information Saturation, we propose Outline-Guided Path Exploration (OPE), which introduces a set of explicit reasoning outlines during the exploration process to dynamically partition the solution space. In this section, we first analyze how the introduction of outlines theoretically overcomes the saturation bottleneck. Subsequently, we detail the practical implementation of OPE, which involves a cold-start stage followed by an iterative RL strategy.

4.1 Breaking Saturation with OPE

Under naive parallel thinking, paths are sampled i.i.d., making it intrinsically difficult to optimize the marginal information gain $I(P_i; Y|P_{1:i-1}, Q)$. To address this, it is desirable to rationally partition the solution space to maximize the coverage of potential solutions before conducting concrete path exploration. OPE achieves this by guiding the model to explicitly plan the directions of exploration in the form of a set of distinct outlines $\mathcal{O} = \{O_1, \dots, O_N\}$ before executing reasoning

steps. Formally, the exploration phase $\pi_\theta(\mathcal{P}|Q)$ is reformulated as a hierarchical process:

$$\pi_\theta(\mathcal{P}, \mathcal{O}|Q) = \pi_\theta(\mathcal{O}|Q) \cdot \prod_{i=1}^N \pi_\theta(P_i|O_i, Q) \quad (11)$$

$\mathcal{O}$ serve as structural directives, guiding the subsequent parallel generation towards N distinct directions, each O_i corresponding to a different strategy for solving the problem. Figure 1 illustrates the structural differences between OPE and naive parallel exploration. Under this new paradigm, the original optimization objective transforms into maximizing the joint mutual information $I(\mathcal{P}, \mathcal{O}; Y|Q)$. Using the chain rule, this objective can be orthogonally decomposed into two components:

$$I(\mathcal{P}, \mathcal{O}; Y|Q) = \underbrace{I(\mathcal{O}; Y|Q)}_{\text{Planning Gain}} + \underbrace{I(\mathcal{P}; Y|\mathcal{O}, Q)}_{\text{Reasoning Gain}} \quad (12)$$

1. Planning Gain ($I(\mathcal{O}; Y|Q)$): This term measures the extent to which the set of outlines $\mathcal{O}$ covers the solution space containing the ground truth. Since outlines are significantly shorter than full reasoning paths, they can be generated as a coherent sequence, enabling the model to explicitly manage diversity among outlines and avoid the diminishing returns associated with i.i.d. sampling.

2. Reasoning Gain ($I(\mathcal{P}; Y|\mathcal{O}, Q)$): This term quantifies the ability to generate correct reasoning trajectories given the constraints of the outlines.

OPE aims to effectively transform the optimization of the total MI $I(\mathcal{P}, \mathcal{O}; Y|Q)$ into the optimization of the two objectives independently.

4.2 OPE with RLVR

We describe the practical implementation of OPE on Qwen3-8B-Base (Yang et al., 2025a) (the state-of-the-art base model in 8B size). The pipeline begins with a cold-start phase to instill the OPE reasoning pattern into the model. Subsequently, we present RL training tailored to the two optimization objectives analyzed in Section 4.1 using GRPO (Shao et al., 2024), and propose an iterative training strategy to further enhance the performance.

Data Given that the primary goal of parallel thinking is to address complex reasoning tasks, we utilize the mathematical training corpus in GURU (Cheng et al., 2025), a high-quality dataset derived from OR1 (He et al., 2025), DAPO (Yu et al., 2025), and DeepScaler (Luo et al., 2025a) sources after deduplication and difficulty filtering. In this paper, we specifically employ its subsets from OR1

and DAPO (36k samples in total), excluding DeepScaler data due to its relatively lower difficulty.

Cold Start Inspired by Parallel-R1 (Zheng et al., 2025b), which demonstrates that zero-shot trajectory synthesis is more stable on simpler queries, we select a subset of 5.4k queries from the full dataset where Qwen2.5-7B-Math (Team et al., 2024) achieves a pass rate greater than 0.5 for our cold-start data synthesis. We then prompt gpt-oss-20b (OpenAI, 2025) to first generate a brief analysis of the query, followed by N distinct reasoning outlines in the tags `<outline_i> ... </outline_i>`. We then concatenate the generated outlines along with a path token `<path_i>` to the original query and require model to generate a corresponding reasoning path strictly following the guidance of O_i . Considering the context window constraints and the complexity of the reasoning tasks, we set $N = 4$ and limit the maximum generation length per query to 8k tokens. To ensure fair comparisons, we instruct gpt-oss-20b to directly generate reasoning paths for these queries, thereby constructing a dataset of identical size for naive parallel thinking.

Outline Planning RL This stage aims to maximize the Planning Gain $I(\mathcal{O}; Y|Q)$ described in Section 4.1, which is equivalent to maximizing the expected reward of the generated outlines $\mathcal{O}$. However, unlike complete reasoning paths, outlines serve as high-level strategic plans and do not contain full problem resolutions, making them unsuitable for direct evaluation via verifiable reward functions. To address this, we estimate the quality of $\mathcal{O}$ by the success rate of the downstream reasoning paths generated following its guidance. Specifically, for each outline $O_i \in \mathcal{O}$, we append the `<path_i>` token and sample K distinct reasoning paths, resulting in a total of $N \times K$ trajectories. The reward function for $\mathcal{O}$ is then defined as the average accuracy of these generated paths:

$$R_{\text{plan}}(\mathcal{O}) = \frac{1}{N \cdot K} \sum_{i=1}^N \sum_{k=1}^K \mathbb{I}(P_{i,k} \rightarrow Y) \quad (13)$$

where $P \rightarrow Y$ denotes that path P successfully reaches the ground-truth answer Y . This formulation encourages the model to generate outlines that lead to high-probability success regions.

Path Reasoning RL This phase aims to optimize $I(\mathcal{P}; Y|\mathcal{O}, Q)$. Since each path P_i represents a complete reasoning trajectory, its correctness can be directly verified against the ground truth. Specifically, we optimize the generation of paths conditioned on a given outline O_i . Using the concatenation

Exploration	Aggregation	Datasets						
		Math500	AMC	AIME 24	AIME 25	HMMT-25	BeyondAIME	AVG.
<i>SFT (Cold Start)</i>								
Naive	Random	78.35	58.94	16.43	17.11	6.56	4.58	30.33
	SC	79.80	60.25	15.21	17.92	5.21	3.80	30.36
	BoN	90.00	80.75	30.63	28.54	13.12	10.60	42.27
	LRM	93.40	87.00	41.88	33.12	17.08	16.30	48.13
OPE	Random	74.55	53.37	15.81	13.59	3.67	4.58	27.60
	SC	77.80	55.00	17.71	14.27	4.48	5.10	29.06
	BoN	88.00	72.75	29.27	23.44	9.69	10.60	38.96
	LRM	<u>90.40</u>	<u>81.75</u>	<u>38.23</u>	<u>29.06</u>	<u>14.06</u>	<u>15.80</u>	<u>44.88</u>
<i>RL Approaches</i>								
Naive	Random	83.75	66.81	23.54	22.03	9.71	7.40	35.54
	SC	86.00	69.00	23.65	24.58	8.75	7.70	36.61
	BoN	91.00	85.25	43.12	32.29	16.15	15.20	47.17
	LRM	91.40	86.00	45.42	36.15	17.40	16.50	48.81
OPE	Random	86.65	71.12	29.90	21.46	10.47	11.28	38.48
	SC	90.20	75.75	31.67	22.81	11.25	11.40	40.51
	BoN	93.80	<u>87.75</u>	<u>47.50</u>	<u>34.69</u>	<u>19.17</u>	20.40	<u>50.55</u>
	LRM	<u>92.20</u>	88.75	49.17	34.17	20.62	<u>19.70</u>	50.77

Table 1: Main results under both Cold Start and RL phases. SC denotes Self-Consistency aggregation, while LRM refers to summary aggregation. The best results are highlighted in **bold**, and the second best results are underlined.

tion of the query, the outline, and the path token $\langle \text{path}_i \rangle$ as input, we employ the verifiable outcome as the reward function:

$$R_{\text{reason}}(P_i) = \mathbb{I}(P_i \rightarrow Y) \quad (14)$$

Formally, the RL training in this phase differs from standard GRPO only by the inclusion of the $\langle \text{path}_i \rangle$ token. Therefore, to make fair comparisons, we train the naive parallel baseline using GRPO with the same reward function.

Iterative OPE Training. The training for outline planning and path reasoning are formally independent. Intuitively, the effectiveness of an outline is bounded by the problem-solving capability of models, while the reasoning performance is conversely dependent on the quality of the guiding outlines. Motivated by this interdependence, we adopt an iterative optimization strategy to achieve co-optimization of outline planning and path reasoning capabilities. In practice, we first conduct Outline Planning RL for a fixed number of steps. We then utilize the final checkpoint to generate outlines for training data, which serves as the input for the subsequent Path Reasoning RL phase. This cycle is repeated multiple times to achieve further improvements. Algorithm 1 illustrates the complete workflow of the OPE training.

5 Experiments

5.1 Setup

Baselines. Our primary baseline, Naive Parallel Thinking, uses identical synthesized cold-start data

and standard GRPO training (Section 4.2). To analyze how path exploration impacts aggregation, we evaluate four strategies: **Random:** Randomly selects one generated path. **Self-Consistency (SC)** (Wang et al., 2022b): Applies majority voting across all paths. **Best-of-N (BoN):** Yields the correct answer if at least one path is correct, serving as the theoretical upper bound for exploration. **LRMs-Based Summary:** Prompts an LRM to reflect on the generated paths and synthesize a final answer.

Training Recipes. For the cold-start, we conduct SFT with $\text{lr} = 1\text{e-}5$ for 2 epochs. Subsequent RL training uses the VeRL (Sheng et al., 2025) with a batch size of 256 and a $1\text{e-}6$ learning rate. We reserve 2k queries from the 36k dataset for validation, using the rest for RL. Across all RL stages, the rollout size is 8. In outline planning RL, each aspect O_i of the outlines $\mathcal{O}$ is randomly paired with $K = 4$ sampled paths for stable reward estimation. Following (Cheng et al., 2025), naive RL training runs for 2 epochs (270 steps). To align OPE training step-wise, we alternate 70 steps of Outline Planning RL and 65 steps of Path Reasoning RL, repeating this cycle twice to match total training steps. Detailed parameters are in Appendix A.2.

Evaluation. We evaluate on six benchmarks of varying difficulty: MATH-500 (Hendrycks et al., 2021), AMC23, AIME24 & 25, BeyondAIME (ByteDance-Seed, 2025), and HMMT-25 (Balunović et al., 2025). For statistical stability, we duplicate queries in AMC23 and BeyondAIME 10 times, and those in AIME24 & 25 and HMMT-

25 32 times. We report accuracy, using the math-verify library (Kydlicek, 2025) to assess equivalence between predictions and ground truth. For aggregation, Self-Consistency ties are broken by selecting the answer with the longer average output length. For LRM-based summaries, we use Qwen3-8B (Yang et al., 2025a) with a generic prompt. Further details are in Appendix A.3.

5.2 Main Results

Table 1 compares Naive and OPE exploration across various aggregation methods and benchmarks, yielding several key insights:

OPE consistently outperforms naive baselines. While OPE initially underperforms Naive exploration in the Cold Start phase, which may be attributed to the domain shift introduced by the outline-guided reasoning format, it demonstrates superior scalability and adaptability during RL training. After RL, OPE achieves the highest average performance across all aggregation methods. Notably, OPE yields the most significant gains in Self-Consistency (SC) aggregation, improving the average accuracy from 36.61% to 40.51%. This improvement indicates that OPE effectively alleviates the bottleneck in majority voting (Figure 2) caused by mutual information saturation.

OPE demonstrates superior improvements on challenging tasks. The performance advantage of OPE becomes increasingly pronounced as task difficulty increases. While the improvement on the relatively simpler MATH-500 dataset is moderate (93.80% vs. 91.00% using BoN), the gains on the more challenging BeyondAIME benchmark are substantial, with OPE achieving 20.40% accuracy in BoN compared to the Naive baseline’s 15.20% (+5.20%). This trend suggests that by structuring the exploration process, OPE helps the model find correct paths that have low probabilities and are often missed by simple independent sampling.

OPE breaks the information bottleneck for LRMs-Based summary. Different the other rule-based aggregation methods that rely solely on the extracted answer in each path, LRMs-based summary can analyze the detailed information within each generated path and perform holistic reasoning over all candidates. While naive parallel thinking benefits from RL training under rule-based aggregation strategies, its improvement under the LRMs-Based Summary approach remains limited (from 48.13% to 48.81%). This suggests that in the naive setting, mutual information saturation among paths

Stage	Rand	Self	BoN	LRM
Cold Start	27.60	29.06	38.96	44.88
<i>Iter 1</i>				
+ Outline RL	31.82	32.55	43.61	48.41
+ Path RL	37.03	38.22	49.18	50.11
<i>Iter 2</i>				
+ Outline RL	37.32	38.90	49.97	50.47
+ Path RL	38.48	40.51	50.55	50.77

Table 2: Average results across benchmarks.

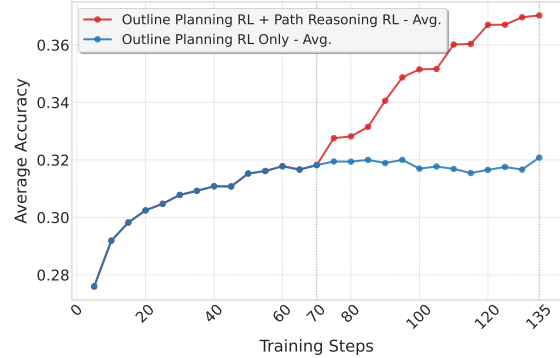


Figure 3: Continued Outline Planning RL (from 70 steps) vs. Path Reasoning RL, measured by average benchmark accuracy (Random aggregation).

restricts the amount of new knowledge available for LRM aggregation. In contrast, with OPE, the average accuracy increases from 44.88% to 50.77%. These results demonstrate that the advantage of OPE lies not only in increasing the probability of discovering correct answers, but also in providing the summary model with a diverse set of reasoning knowledge and informative traces within the explored paths, enabling more effective aggregation.

OPE expands the frontier of path explorations. The BoN metric serves as a proxy for the upper bound of the exploration phase. OPE consistently achieves the highest BoN scores across all benchmarks after RL training, improving the average accuracy by 3.38% (50.55% vs. 47.17%). Note that the parallel sampling budget is constrained to $N = 4$, which highlights the models’ outline planning capability of identify high-probability directions within the solution space, showcasing its potential in low-resource scenarios.

5.3 Ablations

We ablate our iterative co-optimization strategy to evaluate incremental gains across training stages (Table 2). The initial Outline RL significantly expands the exploration frontier, while the subsequent Path RL converts this potential into reliable execution. A second iteration yields further gains

Method	Unique Ans	Avg Tokens
Naive	23.5	2217.3
OPE	27.6	1890.9

Table 3: Statistical comparison ($N = 256$). Unique Ans: average distinct answers per query; Avg Length: average token count of correct paths.

across all metrics, confirming they are mutually reinforcing: better outlines guide effective reasoning, which enables validating more complex outlines. To further ensure Path RL gains stem from genuine path optimization rather than mere additional training steps, we compare it against extending Outline RL for an equal number of steps. As shown in Figure 3, Path RL consistently outperforms the extended Outline RL, confirming the iterative strategy is crucial for realizing OPE’s potential.

6 Analysis

6.1 Statistical Analysis

Beyond overall performance, we analyze OPE’s impact from two perspectives: outline planning and path reasoning (Table 3). From the outline perspective, explicit outlines effectively partition the solution space. OPE generates more unique answers than the naive baseline (27.6 vs. 23.5), indicating broader exploration. This diversity suggests OPE successfully mitigates mode collapse, covering more distinct solution regions. From the path reasoning perspective, outline-guided reasoning is clearer and more efficient. OPE yields significantly more concise reasoning paths, with an average correct path length of 1,891 tokens—about 10% shorter than the naive approach (2,217 tokens). This implies that well-structured outlines help the model focus its reasoning and avoid redundant thinking patterns.

6.2 Test-Time Scaling Analysis

While previous results demonstrate OPE’s effectiveness under a fixed budget (e.g., $N = 4$), we further investigate its scaling properties with increased computational resources via multiple independent samplings. As shown in Figure 4, OPE exhibits superior Pass@k scaling compared to the Naive baseline. On MATH-500 and AMC, OPE rapidly approaches 100% Pass@k, whereas the naive approach saturates at a significantly lower ceiling. On harder benchmarks, the performance gap widens as the sample count increases, highlighting OPE’s advantage in discovering difficult solutions through

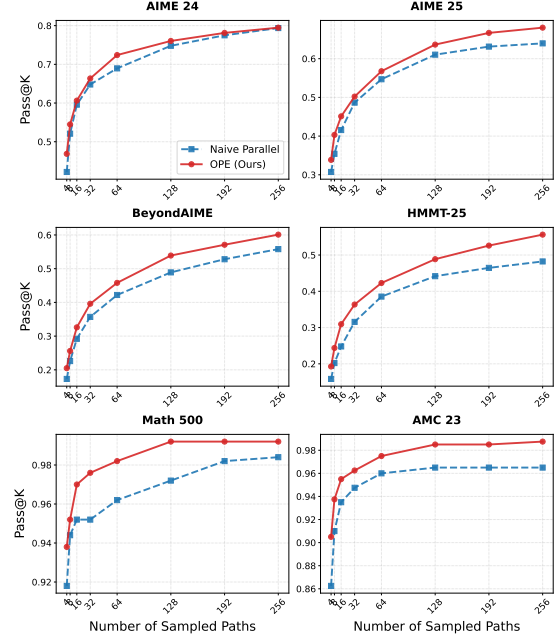


Figure 4: Pass@k scaling comparisons.

diverse exploration. This trend confirms OPE’s potential to effectively scale with additional compute, overcoming the limitations of a fixed initial budget.

6.3 Failure Modes Analysis

We analyze query-level outcomes to understand the failure modes of both strategies. OPE solves significantly more queries where the naive baseline fails (Table 4), demonstrating its enhanced capability to discover solutions in unexplored regions. To further investigate, we count the frequency of correct paths for queries where only one method succeeds (Table 5). In "naive correct, OPE wrong" cases, the correct path typically appears only once (72%), indicating success stems mainly from sampling stochasticity rather than robust reasoning. Conversely, in "OPE correct, naive wrong" instances, OPE consistently produces multiple correct paths (40% of queries have >1 correct path), highlighting outline planning’s effectiveness in reliably covering broader and more challenging solution spaces.

7 Conclusion

We introduce Outline-guided Parallel Exploration (OPE) to enhance LRM parallel thinking by explicitly partitioning the solution space via diverse outlines and iterative reinforcement learning. By mitigating mutual information saturation, OPE significantly boosts performance and scalability, especially on challenging tasks, highlighting its potential for advancing complex parallel reasoning.

Limitations

While OPE demonstrates significant improvements, we acknowledge two main limitations. First, the number of parallel outlines (N) is fixed during training. This rigid configuration prevents the model from dynamically adjusting its exploration budget based on query complexity, leading to redundant computation on simple tasks and potentially insufficient diversity on highly complex ones. Second, the Outline Planning RL phase incurs substantial computational overhead. Since outlines cannot be directly verified, their rewards must be estimated via downstream path success rates. This requires sampling K paths for each of the N outlines, resulting in a multiplicative rollout complexity ($N \times K$) per query. This significantly increases training costs, highlighting the need for future research into dynamic budget allocation or lightweight outline reward models. This paper presents work whose goal is to advance the field of Large Reasoning Models. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Acknowledgement

This work was supported by PKU-CMCC(Hubei) Joint Research Lab for LLM Industrial Applications.

References

- Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. 2025. [Matharena: Evaluating llms on uncontaminated math competitions](#).
- ByteDance-Seed. 2025. Beyondaime: Advancing math reasoning evaluation beyond high school olympiads. [<https://huggingface.co/datasets/ByteDance-Seed/BeyondAIME>] (<https://huggingface.co/datasets/ByteDance-Seed/BeyondAIME>).
- Zhoujun Cheng, Shibo Hao, Tianyang Liu, Fan Zhou, Yutao Xie, Feng Yao, Yuexin Bian, Yonghao Zhuang, Nilabjo Dey, Yuheng Zha, and 1 others. 2025. Revisiting reinforcement learning for llm reasoning from a cross-domain perspective. *arXiv preprint arXiv:2506.14965*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.
- Jujie He, Jiakai Liu, Chris Yuhao Liu, Rui Yan, Chaojie Wang, Peng Cheng, Xiaoyu Zhang, Fuxiang Zhang, Jiacheng Xu, Wei Shen, and 1 others. 2025. Skywork open reasoner 1 technical report. *arXiv preprint arXiv:2505.22312*.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.
- Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. *arXiv preprint arXiv:2306.02561*.
- Hynek Kydlicek. 2025. Math-verify: Math verification library. URL <https://github.com/huggingface/math-verify>.
- Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2023. Making language models better reasoners with step-aware verifier. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5315–5333.
- Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let’s verify step by step. In *The Twelfth International Conference on Learning Representations*.
- Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Li Erran Li, and 1 others. 2025a. Deepscaler: Surpassing o1-preview with a 1.5 b model by scaling rl. *Notion Blog*.
- Tongxu Luo, Wenyu Du, Jiayi Bi, Stephen Chung, Zhengyang Tang, Hao Yang, Min Zhang, and Benyou Wang. 2025b. Learning from peers in reasoning models. *arXiv preprint arXiv:2505.07787*.
- Thang Luong, Edward Lockhart, and 1 others. 2025. Advanced version of gemini with deep think officially achieves gold-medal standard at the international mathematical olympiad. *Google DeepMind Blog*, 1.
- Xuefei Ning, Zinan Lin, Zixuan Zhou, Zifu Wang, Huazhong Yang, and Yu Wang. 2023. Skeleton-of-thought: Large language models can do parallel decoding. *Proceedings ENLSP-III*.
- OpenAI. 2025. [gpt-oss-120b gpt-oss-20b model card](#). Preprint, arXiv:2508.10925.

- Jiayi Pan, Xiuyu Li, Long Lian, Charlie Snell, Yifei Zhou, Adam Yala, Trevor Darrell, Kurt Keutzer, and Alane Suhr. 2025. Learning adaptive parallel reasoning with language models. *arXiv preprint arXiv:2504.15466*.
- Fazlollah M Reza. 1994. *An introduction to information theory*. Courier Corporation.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.
- 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*, pages 1279–1297.
- Iliia Shumailov, Zakhar Shumaylov, Yiren Zhao, Yarin Gal, Nicolas Papernot, and Ross Anderson. 2023. The curse of recursion: Training on generated data makes models forget. *arXiv preprint arXiv:2305.17493*.
- Meituan LongCat Team, Bei Li, Bingye Lei, Bo Wang, Bolin Rong, Chao Wang, Chao Zhang, Chen Gao, Chen Zhang, Cheng Sun, and 1 others. 2025. Longcat-flash technical report. *arXiv preprint arXiv:2509.01322*.
- Qwen Team and 1 others. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2(3).
- Trieu H Trinh, Yuhuai Wu, Quoc V Le, He He, and Thang Luong. 2024. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022a. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022b. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.
- Ziqi Wang, Boye Niu, Zipeng Gao, Zhi Zheng, Tong Xu, Linghui Meng, Zhongli Li, Jing Liu, Yilong Chen, Chen Zhu, and 1 others. 2025. A survey on parallel reasoning. *arXiv preprint arXiv:2510.12164*.
- Hao Wen, Yifan Su, Feifei Zhang, Yunxin Liu, Yunhao Liu, Ya-Qin Zhang, and Yuanchun Li. 2025. Parathinker: Native parallel thinking as a new paradigm to scale llm test-time compute. *arXiv preprint arXiv:2509.04475*.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.
- Xinyu Yang, Yuwei An, Hongyi Liu, Tianqi Chen, and Beidi Chen. 2025b. Multiverse: Your language models secretly decide how to parallelize and merge generation. *arXiv preprint arXiv:2506.09991*.
- Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.
- Yu Yue, Yufeng Yuan, Qiyang Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Chengyi Wang, Tiantian Fan, Zhengyin Du, and 1 others. 2025. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks. *arXiv preprint arXiv:2504.05118*.
- Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, and 1 others. 2025a. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*.
- Tong Zheng, Hongming Zhang, Wenhao Yu, Xiaoyang Wang, Runpeng Dai, Rui Liu, Huiwen Bao, Chengsong Huang, Heng Huang, and Dong Yu. 2025b. Parallel-r1: Towards parallel thinking via reinforcement learning. *arXiv preprint arXiv:2509.07980*.

Algorithm 1 OPE Training

```

1: Input: Dataset  $\mathcal{D}$ , Initial Model  $\pi_\theta$ , Iterations  $M = 2$ ,
   Steps  $S_{\text{plan}} = 70$ , Steps  $S_{\text{reason}} = 65$ 
2: Hyperparameters: Outline Count  $N$ , Path Samples for
   Reward Estimation  $K$ 
3: Output: Optimized Model  $\pi_\theta$ 
4: for  $m = 1$  to  $M$  do
5:   {Phase 1: Outline Planning RL (Optimize  $\pi(\mathcal{O}|Q)$ )}
6:   for  $s = 1$  to  $S_{\text{plan}}$  do
7:     Sample batch of queries  $Q \sim \mathcal{D}$ 
8:     Generate outline sets  $\mathcal{O} = \{O_1, \dots, O_N\} \sim \pi_\theta(\cdot|Q)$ 
9:     for each outline  $O_i \in \mathcal{O}$  do
10:      Append token <path_i> and sample  $K$  paths:
         $\{P_{i,k}\}_{k=1}^K \sim \pi_\theta(\cdot|Q, O_i)$ 
11:    end for
12:    Compute Reward:  $R_{\text{plan}}(O_i) = \frac{1}{K} \sum_{k=1}^K \mathbb{I}(P_{i,k} \rightarrow Y)$ 
13:    Update  $\theta$  via GRPO to maximize  $R_{\text{plan}}$ 
14:  end for
15:  {Transition: Construct Dataset for Phase 2}
16:  Initialize  $\mathcal{D}_{\text{reason}} \leftarrow \emptyset$ 
17:  for each query  $Q \in \mathcal{D}$  do
18:    Generate outlines  $\mathcal{O} \sim \pi_\theta(\cdot|Q)$  using current policy
19:     $\mathcal{D}_{\text{reason}} \leftarrow \mathcal{D}_{\text{reason}} \cup \{(Q, O_i) \mid O_i \in \mathcal{O}\}$ 
20:  end for
21:  {Phase 2: Path Reasoning RL (Optimize  $\pi(P|\mathcal{O}, Q)$ )}
22:  for  $s = 1$  to  $S_{\text{reason}}$  do
23:    Sample batch  $(Q, O_i) \sim \mathcal{D}_{\text{reason}}$ 
24:    Construct Inputs:  $X = [Q, O_i, \text{<path_i>}]$ 
25:    Sample paths:  $P \sim \pi_\theta(\cdot|X)$ 
26:    Compute Reward:  $R_{\text{reason}}(P) = \mathbb{I}(P \rightarrow Y)$ 
27:    Update  $\theta$  via GRPO to maximize  $R_{\text{reason}}$ 
28:  end for
29: end for
30: return  $\pi_\theta$ 

```

	OPE Correct	OPE Incorrect
Naive Correct	1595	232
Naive Incorrect	402	2551

Table 4: Cross-matrix statistics of query success: Naive (rows) vs. OPE (columns).

A Implementation Details

A.1 Data Construction

To construct data for the cold-start phase, we leveraged the metadata provided in the GURU (Cheng et al., 2025) dataset, which includes pass rate statistics evaluated by Qwen2.5-7B-Math. We specifically selected the subset of queries with a pass rate greater than 0.5, ensuring that the problems are solvable. Using these queries, we prompted gpt-oss-20b to generate data following the OPE format. The prompt used for synthesis is illustrated in Appendix B.1. To ensure the quality of the synthesized trajectories, we employed rejection sampling: for each query, we generated multiple candidate trajectories and retained only those that yielded the correct final answer.

A.2 Training

All training experiments, including both the Cold Start SFT and the Iterative RL phases, were conducted on a cluster of 32 NVIDIA H800 GPUs (configured as 4 nodes $\times$ 8 GPUs). The detailed hyperparameters for both stages are listed in Table 6.

A.3 Evaluation

During the evaluation phase, we adhered to the recommended settings for the Qwen3 model family, setting the sampling temperature to 0.6 and top-p to 0.95. To ensure consistency with the training phase and accommodate complex reasoning chains, we set the maximum output token limit for each path to 8K tokens. We extract the final answer from the model’s output by parsing the content within the last `\boxed{\}` command. The extracted answer is then compared against the ground truth using the `math-verify` (Kydlicek, 2025) library to determine correctness.

B Prompts

B.1 System Prompt For Cold Start Data Synthesis & OPE Training

B.2 System Prompt LRMs-Based Summary

C Case Study

Table 5: Distribution of correct path occurrences per query, split into “Naive correct, OPE wrong” and “OPE correct, Naive wrong” cases.

Frequency of correct paths (Total N=4 paths)	Naive correct, OPE wrong	OPE correct, Naive wrong
1	167 / 232 (71.98%)	243 / 402 (60.45%)
2	38 / 232 (16.38%)	93 / 402 (23.13%)
3	17 / 232 (7.33%)	40 / 402 (9.95%)
4	10 / 232 (4.31%)	26 / 402 (6.47%)

Table 6: Training Hyperparameters

(a) Cold Start

Hyperparameter	Value
Epochs	2
Batch Size	256
Max Length	8,192
Learning Rate (LR)	1×10^{-5}
Minimum Learning Rate	1×10^{-7}
LR Decay Style	Cosine
LR Warmup Fraction	0.1
Adam Beta1	0.9
Adam Beta2	0.95
Weight Decay	1×10^{-2}

(b) RL configs using VERL.

Hyperparameter	Value
Epochs	1
data.train_batch_size	256
data.max_prompt_length	2,048
data.max_response_length	8,192
actor_rollout_ref.actor.optim.lr	1×10^{-6}
actor_rollout_ref.actor.ppo_mini_batch_size	32
actor_rollout_ref.actor.ppo_micro_batch_size_per_gpu	4
actor_rollout_ref.rollout.n	8
actor_rollout_ref.rollout.name	vllm

System Prompt For Cold Start Data Synthesis & OPE Training

You are an advanced reasoning assistant capable of Outline-Guided Parallel Thinking.

Instructions:

1. Analysis & Planning: Analyze the query and generate several distinct and diverse outlines representing different solving strategies. Each outline explores a different logical direction.
2. Independent Parallel Exploration: Generate reasoning paths strictly following each corresponding outline. Path P_i must execute the strategy prescribed in Outline O_i without deviation.

System Prompt For LRMs-Based Summary

You are a great reasoner. Here is a problem, and multiple thinkers attempt to give their thought processes independently. Each thinker has written its own thought process towards the final answer. Each thinker is encouraged to take the other thinkers' progress into account to reach the final answer.

===== Problem =====

{problem}

===== Thinkers Thought Process =====

Thinker 1: {path_1}

Thinker 2: {path_2}

Thinker 3: {path_3}

Thinker 4: {path_4}

Look at the above problem and thought process from each thinker, summarize from these thought processes and finally give your answer within \boxed. Summarize their thinking on the problem and try to summarize the thinking of these thinkers. Analyze the differences in thinking between these thinkers and try to analyze which thought process is correct.

Note:

- It is generally believed that when most thinkers get the same answer, the answer may be correct. But you can't do it so superficially, because the correct answer may come from very few thinkers, or even no thinker gives the correct answer. For this reason, when you summarizing, you NEED adhere to the principles of professionalism and critical thinking, carefully identify these thought processes, and give a summary and final answer.
- If you realize that none of these thinkers have answered correctly, you can even learn from the wrong experiences in the thought process of these thinkers and re-think the given problem to give the answer you think is most correct.
- Please DO NOT just solve the given problem independently like other thinkers, but summarize the thought process of all thinkers. In other words, you need to give the summary first, and then give the final answer within \boxed, you can re-think this problem only if you realize that none of these thinkers have answered correctly.

Case Study

The problem asks for the sum of all positive divisors of $9!$ that end with the digit 1. To solve it, we need to understand the structure of $9!$, identify which of its divisors satisfy the units-digit condition, and then sum those divisors. Key considerations include the prime factorization of $9!$, the properties of divisors, and the behavior of units digits under multiplication. The solution space is finite but large, so an efficient strategy involves leveraging modular arithmetic and combinatorial counting of divisor pairs. The challenge lies in filtering divisors by their last digit while keeping track of their sum.

<outline_1>**Prime Factorization Lens** Examine the prime factorization of $9!$ to understand the building blocks of its divisors and how their units digits are determined.</outline_1>

<outline_2>**Modular Arithmetic Filter** Use congruence modulo 10 to identify which divisor combinations yield a units digit of 1, narrowing the set of candidates.</outline_2>

<outline_3>**Divisor Pair Symmetry** Exploit the fact that divisors come in pairs whose product is $9!$ to simplify the counting and summing process.</outline_3>

<outline_4>**Combinatorial Counting Strategy** Count the number of divisors with the desired units digit by analyzing the exponents in the prime factorization and applying combinatorial principles.</outline_4>

<path_1>

We need to compute sum of positive divisors of $9!$ that have units digit 1. We must use prime factorization lens. So we need to factor $9!$, find divisors, filter by units digit 1, sum. Let's do step by ...

$$= 4 \Rightarrow 2^0 3^4 = 81$$

Both 1 and 81 are divisors of $9!$ (since $9!$ contains 3^4).

Step 7 – Compute the sum

$$1 + 81 = 82$$

82

</path_1>

<path_2>

We need to compute sum of positive divisors of $9!$ that have units digit 1. We must use modular arithmetic filter. Let's proceed.

First compute $9! = 9 \cdot 8 \cdot 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$. Calculate: $9! = 362880$. But we considered all a,b,e combinations. The only ones that gave units digit 1 were the ones we found. Thus sum is 106.

Thus final answer: 106. The final answer in this path:

106

</path_2>

<path_3>

We need to compute sum of positive divisors of $9!$ that have units digit 1. We must use the outline: Divisor Pair Symmetry. So we should use the fact that divisors come in pairs whose product is $9!$. We ...uct $42 \rightarrow 2$. Not 1. $a=8, b=1, d=1$: $u_2=6, u_3=3, u_7=7 \Rightarrow$ product $126 \rightarrow 6$. Not 1.

Thus only 1 and 21.

Thus sum = 22.

We should output 22. The final answer in this path:

22

</path_3>

<path_4>

We need to compute sum of positive divisors of $9!$ that have units digit 1. We must use the outline: Combinatorial Counting Strategy. So we need to analyze prime factorization of $9!$, determine which $d \mid 9!, d \equiv 1 \pmod{10} \rightarrow d = 1, 21, 81$

Thus the divisors of $9!$ with units digit 1 are 1, 21, 81.

3. Sum of these divisors

$$1 + 21 + 81 = 103.$$

103

</path_4>



Figure 5: Comparison of the performance curves of Naive and OPE approaches across different datasets as training steps progress.