POPE: Learning to Reason on Hard Problems via Privileged On-Policy Exploration

Anonymous authors
Paper under double-blind review

Abstract

Reinforcement learning (RL) has improved the reasoning abilities of large language models (LLMs), yet state-of-the-art methods still fail to learn on many training problems. On *hard* problems, on-policy RL rarely *explores* even a single correct rollout, yielding zero reward and no learning signal. We find that natural solutions to remedy this exploration problem from classical RL, such as entropy bonuses, more permissive clipping of the importance ratio, or direct optimization of pass@k objectives, do not resolve this issue and often destabilize optimization without improving solvability. A natural alternative is to leverage transfer from easier problems. However, we show that mixing easy and hard problems during RL training is counterproductive due to *ray interference*, where optimization focuses on already-solvable problems in a way that actively inhibits progress on harder ones. To address this challenge, we introduce **Privileged On-Policy Exploration** (POPE), an approach that leverages human- or other oracle solutions as *privileged* information to guide exploration on hard problems, unlike methods that use oracle solutions as training targets (e.g., off-policy RL methods or warmstarting from SFT). POPE augments hard problems with prefixes of oracle solutions, enabling RL to obtain non-zero rewards during guided rollouts. Crucially, the resulting behaviors transfer back to the original, unguided problems through a synergy between instruction-following and reasoning. Empirically, POPE expands the set of solvable problems and substantially improves performance on challenging reasoning benchmarks.

1 Introduction

Reinforcement learning (RL) has significantly improved the reasoning abilities of large language models (LLMs) in domains such as math and coding. In particular, relatively small models trained with RL to better exploit test-time compute via longer chains of thought (CoT) can outperform much larger models trained without RL (Setlur et al., 2025b; Liu et al., 2025). While some works argue that RL post-training primarily amplifies existing capabilities (Yue et al., 2025; Zhao et al., 2025), others show that careful design choices can mitigate these effects (Setlur et al., 2025b; Liu et al., 2025). Across various RL recipes, a shared limitation is that on-policy RL fails to train on a large fraction of available problems, leaving substantial gains untapped. On-policy RL often cannot sample any non-zero–reward rollout on *hard* problems relative to the base model, yielding no learning signal; for instance, when running `Qwen3-4B-Instruct` on DAPO-MATH-17K (Yu et al., 2025), fewer than 50% of problems produce a correct rollout even with $K = 32$ attempts and a 16k token budget. Common throughput-oriented heuristics, such as dynamic sampling and zero-variance filtering, further discard these problems explicitly (Yu et al., 2025; Wang et al., 2025b; Khatri et al., 2025).

How can we make progress on hard problems? In a typical RL framing, this would require improving the "exploration" (i.e., rollout generation) mechanism used during learning. While standard on-policy RL relies on inherent stochasticity of the base model's distribution to guide exploration, on hard problems this naïve exploration strategy is insufficient. A natural attempt would be to employ token-level exploration bonuses from classical deep RL to incentivize exploration. We empirically analyze two representative methods from this category, and find that neither approach improves "solvability" (i.e., obtaining at least *one* correct rollout when sampling multiple) without destabilizing optimization.

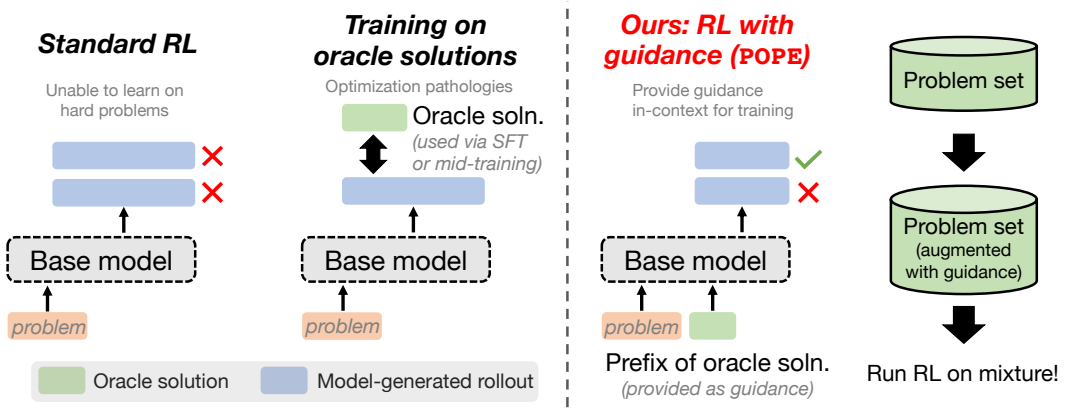

Figure 1: **A schematic of our approach, Privileged On-Policy Exploration (POPE) compared with other approaches for training on hard problems,** where standard on-policy RL largely fails to produce successful rollouts. POPE uses oracle (e.g., human-written) solutions to solely guide on-policy exploration during RL, without ever training on the oracle information as targets (for e.g., via supervised fine-tuning or mid-training). We show that modifying standard RL training objectives to incentivize token-level exploration frequently introduce optimization pathologies. Our training approach (POPE) sidesteps such pathologies, while enabling learning from oracle information by "instructing" the model to build upon it.

An alternative is to leverage *transfer* to guide exploration. Skills learned on easy problem can inform exploration on harder ones. To test whether such transfer enables exploration on hard problems, we train on mixtures of easy and hard problems. Through controlled experiments, we find that even when mixing in the most closely related easy problem, on-policy RL makes slow progress on a hard problem: it first "sharpens" the base model on the easy subset before improving on the hard one. We explain this behavior via ***ray interference*** (Schaul et al., 2019) (Figure 2): an implicit bias in on-policy RL towards further optimizing reward on states where reward is already attained rather than finding reward on new states. Consequently, learning on hard problems requires first obtaining non-zero reward by explicitly encouraging exploration.

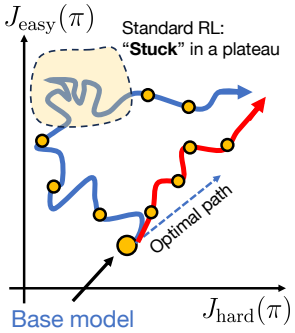

Figure 2: ***Interference.*** In on-policy RL, training on a mixture of easy and hard problems preferentially accelerates progress on easy problems, often stalling or degrading performance on hard ones. This imbalance leads to plateaus during training; an ideal approach would induce more "uniform" progress across all problems.

If the base model cannot sample correct rollouts on a hard problem, how can we obtain non-zero reward? A natural approach is to collect "expert" traces from humans/oracle and either distill them into the base model (Sessa et al., 2024; Agarwal et al., 2024a) or use them in RL as off-policy data (Yan et al., 2025b). However, the type of reasoning traces that LLMs are trained to produce are prohibitively expensive to obtain, and prior work finds limited gains from available human-written data. Empirically, we find that distillation often caps gains from RL and off-policy training destabilizes RL. We therefore seek a more effective source of ***exploratory signal*** on hard problems.

Our key insight is that oracle solutions can effectively guide an LLM's *on-policy exploration* on hard problems, even when they are ineffective as training targets. Consider a hard problem where the LLM repeatedly follows incorrect approaches and fails within the training budget: conditioning on even a short prefix of a "privileged" human-written or oracle-provided solution can substantially increase the probability of reaching the correct answer. This effect is particularly pronounced when the base model has strong instruction-following capabilities, allowing us to steer it into building upon privileged content. **Privileged On-Policy Exploration** (POPE) leverages this principle to guide exploration in RL and this exploration is performed fully on-policy, providing an alternative to distillation or off-policy RL (Figure 1).

Concretely, for a set of hard problems, POPE collects a human- or oracle-provided solution and uses a short prefix of this solution as privileged guidance during training. We train the base LLM with RL on a mixture of the original hard prompts and guided variants augmented with this fixed prefix (optionally together with easier prompts). Although these partial solutions are poor training targets, conditioning on them and "instructing" the policy to utilize them, reliably steers on-policy rollouts into regions where at least one correct attempt can be sampled. Behaviors learned under guidance through RL then transfer back to the original, unguided problems, greatly reducing difficulty of solving the hard problem from scratch. This transfer is

enabled by **(a)** strong instruction-following, which allows the model to build on the prefix despite being unable to generate it itself, and **(b)** by backtracking and reflection behaviors that revisit and reinterpret the guidance during reasoning. When viewed through an RL lens, instruction-following and backtracking improves the overlap between the distribution of underlying states with and without any privileged guidance, which in turn enables transfer. Finally, from a classical RL perspective, `POPE` mirrors a key principle from off-policy RL: learning from on-policy actions from off-policy states (problems + guidance) can be much more effective than learning from both off-policy states and actions (Park et al., 2024).

Results. `POPE` enables models to solve hard problems that remain unsolvable with standard RL, using either human-provided solutions. On a hard training subset, `POPE` solves 10% more problems measured via pass@16 with 64 rollouts and a 32k token budget. These gains persist even when training on mixtures of easy and hard problems, where guided exploration outperforms naïve mixtures and avoids collapse into sharpening on already-solvable problems. On standardized benchmarks such as AIME 2025 and HMMT 2025, `POPE` consistently improves both pass@1 and pass@k, achieving up to 58% pass@1 and 83% pass@16 (vs. 48% and 77% for the base model), demonstrating robust population-level improvements.

2 Preliminaries and Notation

We study RL post-training of a base large language model (LLM), denoted by π_{base} with parameters θ. For any given input problem $\mathbf{x} \sim \rho$ and a rollout $\mathbf{y} \sim \pi(\cdot \mid \mathbf{x})$ attempting to solve this problem, we define a *binary outcome reward* $r(\mathbf{x}, \mathbf{y}) \in \{0, 1\}$ indicating correctness of the answer in these rollouts. Analogous to most work on RL with LLMs, we assume that the rollout $\mathbf{y}$ represents the final answer in a `\boxed{}` block. We study several measures of performance, including the pass@k metric, given by

$$[\text{pass@}k](\mathbf{x}) = \Pr\big[\exists\, \mathbf{y}_1, ..., \mathbf{y}_k \overset{\text{i.i.d.}}{\sim} \pi(\cdot \mid \mathbf{x}) \;\text{ s.t. }\; \max_{j=1}^{k} \; r(\mathbf{x}, \mathbf{y}_j) = 1\big], \tag{1}$$

which measures the probability that at least one of k independent attempts from a model π at the problem $\mathbf{x}$ succeeds. This metric captures the role of parallel exploration during training and measures whether a batch can yield any positive signal for policy gradient algorithms that do not train an explicit value function (e.g., GRPO (Shao et al., 2024)) and rely on Monte-Carlo rollouts for estimating the policy gradient. We use the empirical pass@k value, denoted by $[\widehat{\text{pass@k}}](\mathbf{x})$ as a metric to quantify "solvability" of a problem $\mathbf{x}$ during training. We estimate pass@k by drawing n independent samples from the model, with $n \gg k$.

Outcome-reward on-policy RL. Most RL algorithms train the base model π with a policy gradient, which reinforces rollouts that end in a correct final answer, and reduces probability of rollouts that end up in the wrong answer (i.e., the negative gradient (Setlur et al., 2025b)). This process is also called *outcome-reward RL*. In practice, some of the most-commonly used RL algorithms such as GRPO, uses a reference policy π_{old} for sampling, and normalize rewards into *advantages* before utilizing them in the policy gradient: $A_i(\mathbf{x}, \mathbf{y}_i) = r(\mathbf{x}, \mathbf{y}_i) - \frac{1}{n}\sum_{j=1}^{n} r(\mathbf{x}, \mathbf{y}_j)$, so that updates depend on deviations of reward from the batch mean. This normalized structure makes RL brittle on hard problems. If all n rollouts fail on a given problem $\mathbf{x}$ ($r(\mathbf{x}, \mathbf{y}_i) = 0$), then the advantage for all samples vanishes, $A_i = 0$, and the gradient update is exactly zero on $\mathbf{x}$. Thus when $[\widehat{\text{pass@}k}](\mathbf{x}; \pi_\theta) \approx 0$, training stalls: advantages cannot generate signal, even with large batch sizes over problems or running training for longer. This creates a pathological feedback loop where the model sharpens on "easy" problems but halts learning on "hard" ones.

Definition 2.1 (Hard and easy problems)**.** A problem $\mathbf{x}$ is called *hard* for a given base model π_{base} if for a sufficiently large value of k, $[\widehat{\text{pass@k}}](\mathbf{x}; \pi_{\text{base}}) \approx 0$. A problem is called easy if it is not hard.

An important consideration when applying Definition 2.1 in practice is the output length used to evaluate this definition. Naïve empirical evaluations of the pass@k metric could underestimate its true value due to truncation of long model rollouts under low length budgets. This can make easy problems appear artificially harder. We find that training on such problems often does not pose a challenge, since models are able to "compress" their reasoning traces without any complex exploration problem. For our experiments, we therefore run all rollouts used to estimate $\widehat{\text{pass@k}}(\mathbf{x})$ until completion, up to 32k tokens for our base model `Qwen3-4B-Instruct`, and evaluate pass@k for values of k up to 128.

RL training loss. Our approach and most of our analysis are both agnostic to the choice of the underlying training loss. But some of our analysis in Section 3 does utilize details of the RL training objective. We run a streaming, asynchronous implementation (Piché et al., 2025) of GRPO (DeepSeek-AI et al., 2025; Shao et al., 2024) as our RL training algorithm, without any entropy and KL divergence terms as default. The GRPO loss uses a clipped surrogate similar to PPO (Schulman et al., 2017), averaged over groups of trajectories. A typical loss function we optimize is:

$$\mathcal{L}_{\mathrm{RL}}(\theta) = \mathbb{E}_{\mathbf{x},\mathbf{y}\sim\pi_{\mathrm{old}}}\left[\min\left(\frac{\pi_\theta(\mathbf{y}\mid\mathbf{x})}{\pi_{\mathrm{old}}(\mathbf{y}\mid\mathbf{x})}A(\mathbf{x},\mathbf{y}),\ \mathrm{clip}\left(\frac{\pi_\theta(\mathbf{y}\mid\mathbf{x})}{\pi_{\mathrm{old}}(\mathbf{y}\mid\mathbf{x})},\ 1-\epsilon_{\mathrm{low}},\ 1+\epsilon_{\mathrm{high}}\right)A(\mathbf{x},\mathbf{y})\right)\right], \tag{2}$$

where $A(\mathbf{x},\mathbf{y})$ denotes the advantage estimate discussed above, ϵ_{low} and ϵ_{high} are the low and high clipping thresholds. DAPO (Yu et al., 2025) sets $\epsilon_{\mathrm{high}} > \epsilon_{\mathrm{low}}$ to enable less conservative updates on positives that might be less likely under the sampling distribution π_{old}.

3 Why is Naïve Exploration Insufficient on Hard Problems?

To motivate the design of our approach in the next section, we first perform a systematic analysis to understand the efficacy and training dynamics of several exploration strategies when training on hard problems, where experiencing non-zero reward is challenging. The analogy to exploration in classical RL naturally suggests that seemingly straightforward techniques for encouraging exploration might help.

We therefore study a representative subset of these techniques. The first type of methods perform "token-level" exploration by modifying the RL training objective or incorporating a bonus. The second type relies on transfer across problems by training on a mixture of easy and hard problems (Sun et al., 2025). In both cases, we observe characteristic failure modes pertaining to poor optimization or an amplification of "interference" where any strategies learned on easy problems do not guide learning on hard problems.

3.1 Token-Level Exploration on Hard Problems

We first study the behavior of methods that incentivize token-level exploration on hard problems by training a `Qwen3-4B-Instruct` model on our hard problem set. Although this model cannot solve most of these problems initially, training for sufficiently many steps can yield non-zero reward on a small subset (approximately 6%). To incentivize exploration, we experiment with two variants.

First, we add an entropy bonus together with a KL penalty to the policy objective. As shown in Figure 3, this modification does not make hard problems solvable: the fraction of problems with no correct solution among eight rollouts remains close to that of naïve RL throughout training. More problematically, the entropy of the model's next-token distribution increases sharply, leading to an uncontrolled explosion early in training from which the model fails to recover. Seeing this, we next attempt to increase exploration *without* using an explicit entropy bonus. Specifically, we increase the high clip ratio, ϵ_{high} (Equation 2), in the update following DAPO (Yu et al., 2025), with the goal of updating the model on rare positive traces that would otherwise be clipped. As shown in Figure 3, this approach also increases entropy, somewhat unexpectedly, but does not meaningfully improve solvability and performs no better than the entropy-based approach.

As discussed in Appendix A, there is a systematic reason why the next-token entropy increases with a higher clip ratio. Briefly, when we use an importance-sampled policy gradient (Eq. 2) to train on rare positive traces, it attempts to shift probability mass toward these tokens using only a single gradient step. This both reduces confidence in tokens favored by the base model and fails to properly fit the positive trace, resulting in increased uncertainty and effectively random exploration. This increase in entropy snowballs over training resulting in entropy explosion. We detail this phenomenon in Appendix A.

Takeaways: Token-level exploration is insufficient on hard problems

- Entropy bonuses cause uncontrolled entropy growth, inhibiting learning on hard problems.
- Higher clip ratios can help address the above but utilizing higher values results in amplification of entropy that ultimately results in random and meaningless exploration.

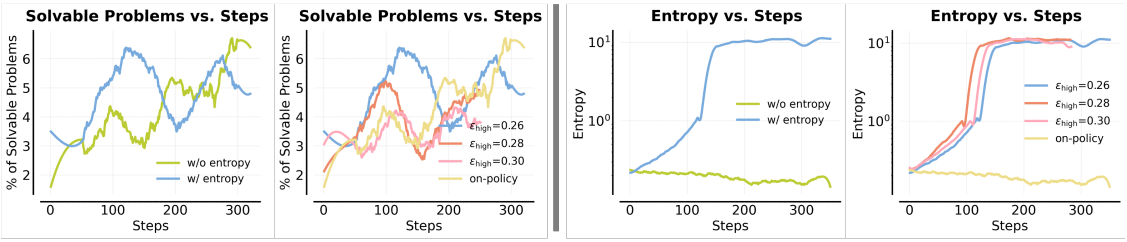

Figure 3: **Left: Evolution of the fraction of solvable problems** (measured via the pass@8 at 16k output length). **Right: average token-level entropy statistics over the course of RL training**. Observe that all of these representative classical exploration methods make similar amounts of (few) problems solvable, while creating pathologies in optimization in the sense that entropy blows up. We do notice large sensitivity to the clip threshold ϵ_{high} in our runs.

3.2 Ray Interference Inhibits Exploration via Transfer

An alternative to token-level exploration is to leverage reasoning behaviors learned on easier problems as building blocks that can be composed to solve harder ones when given a larger token budget. This idea is referred to as *extrapolation* (Setlur et al., 2025b): if training on easier problems produces a model that can use additional test-time compute to chain together multiple strategies, then on-policy RL may amplify this effect without needing specialized exploration mechanisms.

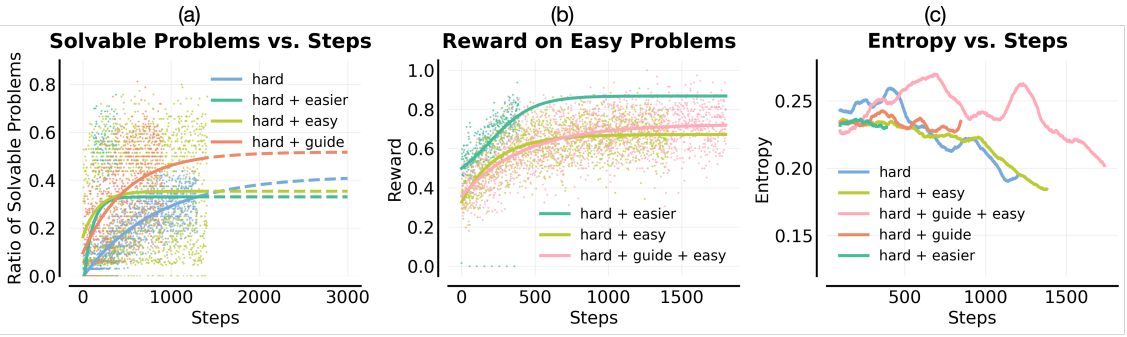

Figure 4: **No meaningful transfer from learning easy problems to hard problems. (a)** evolution of the fraction of solvable problems (measured via pass@8 at 16k response length). **(b)** average training reward on easy problems mixed in training. **(c)** average token-level entropy over the course of RL training. Since we do not use an entropy bonus, entropy generally remains stable (or slightly decreases) throughout training. Observe that the fraction of solvable problems increases the most when using our guidance-based approach, "hard + guide". In contrast, incorporating easy prompts does not improve solvability of hard problems, providing a negative result for the transfer hypothesis for improving exploration on hard problems.

To stress test whether transfer can guide exploration, we co-train on a mixture of easy and hard problems, with each subset containing 256 problems. Here, we define easy problems as those on which the base model achieves approximately 30% success rate, evaluated with a 32k token budget and 128 rollouts, while easier problems correspond to those with roughly 60% success rate under the same evaluation protocol. The motivation is that progress on easier problems during training might transfer to improved exploration and solvability on hard ones. As shown in Figure 4, ***we observe no meaningful improvement in solvability (pass@32) of hard problems***. While mixing in easy problems ("hard + easy") accelerates early gains in pass@32 on the hard set, the pass@32 performance quickly plateaus and converges to a lower asymptote than training on the hard problems alone. This indicates that learning on arbitrary easy problems does not transfer the exploration capabilities required to solve hard ones. A similar effect occurs when easier problems are mixed in ("hard + easier" in Figure 4), which in fact results in even fewer hard problems being solved during training. In contrast, our approach (that we discuss in the next section) yields higher solvability rates, improving pass@8 by approximately 13% relative to all mixture-based baselines. These results show that transfer from easy problems is insufficient for exploration.

Didactic experiment with only one easy and one hard problem. To understand why training on a mixture of easy and hard problems does not help, we run RL training in a didactic setting with only *two* problems: one easy and one hard and show results in Figure 5. As expected, training on only the hard problem ("hard" in Figure 5c and 5d) often yields zero reward until the model succeeds once due to randomness, after which learning picks up and reinforces this success quickly. Mixing in a very related easy problem, measured by cosine similarity between the textual embeddings of the hard and easy problems under the base model,

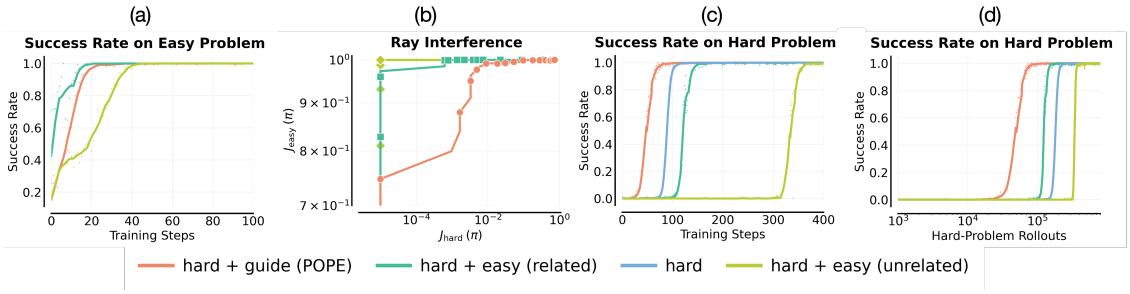

Figure 5: **Didactic two-problem experiment illustrating ray interference.** We train on a setting consisting of one easy and one hard problem. **(a)** Success rate on the easy problem versus training steps. All methods rapidly solve the easy problem. **(b)** Optimization trajectories visualized by plotting $J(\pi_\theta; \text{easy})$ and $J(\pi_\theta; \text{hard})$ jointly over training. Mixing in an unrelated easy problem leads to rapid improvement on the easy problem at the cost of stagnation on the hard problem, illustrating negative transfer due to ray interference. **(c)** Using a related easy problem partially mitigates this effect, but remains inefficient and requires many more training steps to solve the hard problem compared to training on the hard problem alone. Our approach ("hard + guide") is the only one that improves convergence speed on the hard problem of all methods. **(d)** Success rate on the hard problem vs. the number of rollouts allocated to it. Beyond interference, POPE improves sample efficiency by reducing the number of rollouts required to learn the hard problem, indicating an acceleration in solvability of the hard problem.

slightly accelerates training; see "hard + easy (related)" in Figure 5. In contrast, mixing in an easy but unrelated problem that exhibits the lowest cosine similarity with the hard problem ("hard + easy (unrelated)" in Figure 5) substantially slows convergence on the hard problem to a point where RL learns to solve the hard problem much slower than simply training on the hard problem alone. This is a form of ***interference*** between learning on different problems. Crucially, the related and unrelated easy problems (as well as the guided variant) are *matched in base difficulty*: under the base model, exhibiting similar success rates.

We further visualize the optimization trajectory by plotting rewards on the easy and hard problems, $J(\pi_\theta; \text{easy})$ and $J(\pi_\theta; \text{hard})$, against each other over training in Figure 5b. Across all settings, the easy problem begins accumulating reward early. When the easy problem is unrelated, RL preferentially optimizes $J(\pi_\theta; \text{easy})$ while progress on $J(\pi_\theta; \text{hard})$ stagnates, a form of negative interference consistent with ***ray interference*** (Schaul et al., 2019). Ray interference is fundamentally a function-approximation effect in on-policy RL: the same mechanisms that enable transfer across related tasks can hinder learning when problems are semantically disjoint or exhibit large performance skew. Although related easy problems partially mitigate this effect, optimization on the hard problem remains slow. In contrast, applying POPE enables smoother optimization of $J(\pi_\theta; \text{hard})$ ("hard + guide (POPE)" in Figure 5b), yielding a more favorable trajectory that reduces interference from the easy prompt and improves exploration. This experiment isolates ray interference in a minimal setting and explains why naïve transfer from easy to hard is insufficient.

Can we solve this issue by optimizing the empirical pass@k metric directly? A natural next question is whether directly optimizing the empirical pass@k objective can address the lack of progress on hard problems. Prior work has proposed optimizing pass@k-style rewards to encourage population-level diversity and reduce sharpening (Chow et al., 2024; Walder & Karkhanis, 2025). While this approach can mitigate distribution collapse on problems where the model already attains occasional successes, it does not resolve the core difficulty on hard problems where the pass@1 score attained by the base model are quite low.

As shown in Figure 6, increasing k consistently degrades performancein both pass@k (solvability) and average pass@1 reward. While a drop in pass@1 is expected when optimizing pass@k due to an objective shift, pass@k optimization also fails to improve solvability. Why? Consider a setting in which hard problems are independent, so reward obtained on one problem does not transfer to others. In this regime, pass@k optimization can improve solvability only if the model achieves non-zero pass@1 on each problem (since pass@k is a monotonic function of per-problem pass@1), which does not hold for hard problems. Moreover, even when correct rollouts occasionally exist, pass@k optimization redistributes reward toward incorrect traces to encourage diversity, shrinking the reward gap between positive and negative samples. On hard problems, where correct rollouts are already in both pass@k (solvability) and average pass@1 reward, this inhibits learning. As a result, pass@k optimization may mitigate over-sharpening but is ineffective at driving exploration and can slow convergence. We provide a detailed analysis of this behavior, including the pass@k objective and its policy-gradient estimator, in Appendix B.

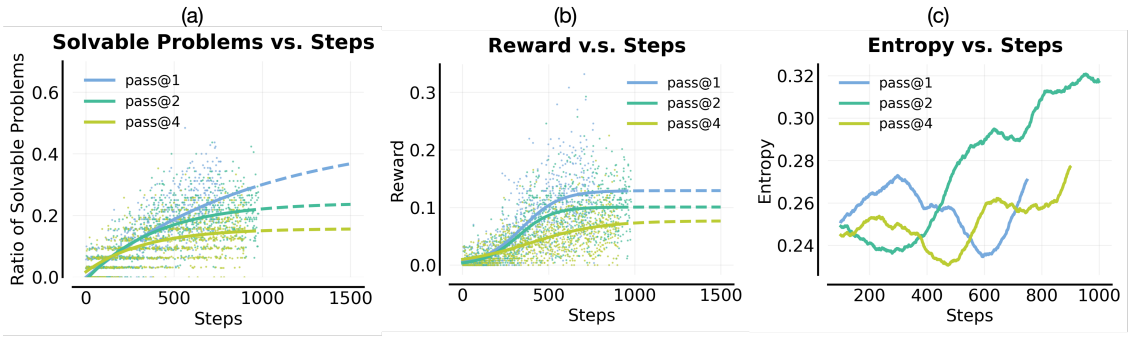

Figure 6: **Directly optimizing pass@k fails to improve exploration on hard problems and primarily prevents over-sharpening on already-solvable ones. (a)** Evolution of the fraction of solvable hard problems under different pass@k objectives (measured at pass@8). **(b)** Average training reward when optimizing pass@k compared to standard on-policy RL (pass@1). **(c)** Average token-level entropy during training. Although pass@k optimization is intended to promote population-level diversity, it does not improve hard-problem solvability. Instead, increasing k consistently degrades performance relative to pass@1. These results indicate that pass@k optimization cannot bootstrap learning when the initial success probability is near zero: it primarily redistributes reward to incorrect traces to reduce over-sharpening, reinforcing already-solvable problems rather than enabling exploration on previously unsolved ones.

Takeaways: Ray interference hurts exploration on hard problems

- As RL starts solving some easy problems, its ability to solve other hard problems reduces.
- This phenomenon can be explained via ray interference from multi-task RL. Ray interference leads to stagnation and inefficient performance improvement on hard problems.

4 POPE: Privileged On-Policy Exploration

In this section, our goal is to develop an exploration approach that enables the model to learn how to solve new, hard problems. To address the limitations of pure on-policy exploration, we leverage oracle solutions, such as human-written solutions, during training. A natural approach would be to train directly on these oracle solutions, either by imitating them via supervised fine-tuning (SFT) before running standard on-policy RL, or by incorporating them directly as additional rollouts during RL. However, we find that both approaches distort the base model's reasoning patterns and lead to optimization instabilities.

Limitations of training on oracle solutions. Concretely, running SFT on human-written solutions over multiple epochs to closely fit the oracle data causes the model to memorize these solutions, resulting in a low-entropy initialization that inhibits meaningful exploration during subsequent RL. Conversely, early stopping SFT to avoid memorization yields a high-entropy initialization that cannot reliably produce rollouts in the style of either the base model or the oracle solutions. In both cases, the resulting policy is not effective enough for further improvement or generalization. We compare against improved variants of SFT and off-policy RL in our experiments in Section 6. Next we develop our approach.

Our approach. Rather than using oracle solutions as training targets, **our key idea** is to use them solely to *steer* on-policy rollouts. We augment each hard problem with guidance in the form of a short prefix of an oracle solution and instruct the model to follow and build upon this guidance (see the system instruction at the end of this section). Although the model cannot generate such sequences on its own, conditioning on the partial solution moves it into more favorable regions of the response space from which non-zero reward becomes attainable. From an RL perspective, this corresponds to initializing rollouts from off-policy "states" informed by human-written solutions, while learning remains fully on-policy. We train on a mixture of unguided hard problems and their guided variants, optionally including easy problems to broaden coverage. We find that this mixture enables behavior learned under guidance to transfer to unguided hard problems (Section 5), often mitigating ray interference and improving overall success. We refer to this approach as **privileged on-policy exploration** (POPE; Figure 7).

Formal description. Formally, given an oracle solution $\mathbf{z}$ to a hard training problem $\mathbf{x} \sim \mathcal{D}_{\text{hard}}$, we generate rollouts conditioned on a prefix $\mathbf{z}^{0:i}$ of $\mathbf{z}$ and a system instruction I that "instructs" the model to build upon $\mathbf{z}^{0:i}$, i.e., $\mathbf{y} \sim \pi(\cdot|\mathbf{x}, \mathbf{z}^{0:i}, I)$. In principle, any prefix $\mathbf{z}^{0:i}$ could be used as guidance. However, an overly long prefix that solves a substantial portion of the hard problem is not useful for learning reasoning strategies that

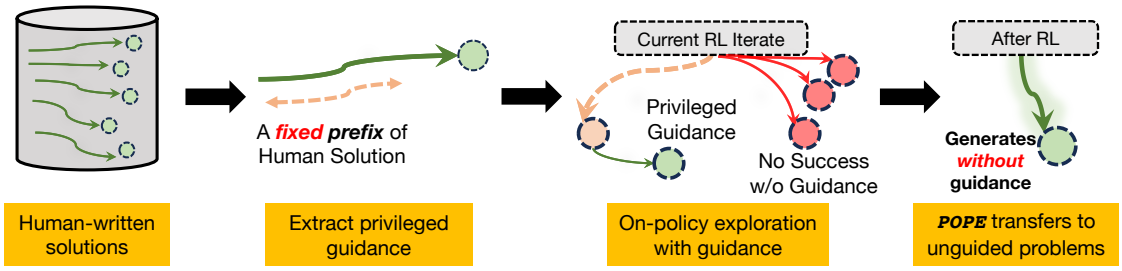

Figure 7: **Illustration of our approach POPE.** POPE trains the model by using privileged guidance from human solutions to condition on-policy generations. We show that training on a mixture of guided and unguided problems then allows transfer of the learned reasoning strategies to the unguided problem.

can transfer to the unguided setting. We therefore restrict ourselves to a short prefix that is sufficient to enable on-policy rollouts to obtain *some* non-zero reward on $\mathbf{x}$. To identify such a prefix, we evaluate the base model's ability to produce at least one successful rollout when conditioned on a set of coarsely chosen, uniformly spaced prefixes, and select the shortest prefix that yields a successful trace under the base model. Let's denote $i^*(\mathbf{x})$ as the length of this short prefix for a problem $\mathbf{x}$. On problems where no prefix leads to a successful rollout, we simply utilize a randomly-chosen prefix that is smaller than $1/4^{\text{rd}}$ of the oracle solution. Using this, we construct a guided set of hard problems:

$$\mathcal{D}_{\text{hard}}^{\text{guided}} := \left\{ \text{concat}(\mathbf{x}, \mathbf{z}^{0:i^*(\mathbf{x})}, I) \mid \mathbf{x} \in \mathcal{D}_{\text{hard}} \right\}. \tag{3}$$

POPE then trains on a dataset consisting of a 1:1 mixture of (unguided) hard problems $\mathcal{D}_{\text{hard}}$ and their guided versions $\mathcal{D}_{\text{hard}}^{\text{guided}}$. Finally, we emphasize that POPE operates fully on-policy: although privileged information guides exploration, the exploration itself is carried out by the model via on-policy rollouts.

POPE System Instruction

You are given a problem and a partial solution. Your task is to **study the partial response, identify what reasoning or steps are already provided, and then complete the solution from where it left off**. Ensure your response is logically consistent and leads to a complete and correct final answer.

Important: *Show your reasoning step-by-step, and present the final answer using LaTeX-style* $\boxed{\cdot}$.
Problem: <Problem>
Partial Response: <Partial Response>
Continue solving the problem, starting from where the partial response ends. Make sure your final answer is written as: $\boxed{\text{your answer here}}$

We find that POPE enables models to gradually learn to solve unguided versions of hard problems that standard RL on the base model fails to solve, resulting in a form of transfer that we analyze next.

Summary: Privileged On-Policy Exploration (POPE)

- POPE conditions on partial solutions from an oracle as privileged information to guide on-policy rollouts during RL training, instead of directly using oracle data as training targets.
- We identify a short prefix of the oracle solution that enables the base model to succeed once to augment a hard problem. We train on a 1:1 mixture of guided and unguided problems.

5 Why Does POPE Work?

We now conceptually and empirically study why learning on guided versions of hard problems transfers to improving performance on their unguided counterparts when training with POPE. Since the model is never trained to imitate the guidance tokens themselves, the source of this transfer is not immediately obvious. Our explanation is based on a simple mental model in which *stitching* plays a central role.

5.1 A Mental Model

To build intuition for why POPE works, we consider a simple mental model of exploration in a Markov decision process (MDP). Suppose that obtaining reward from the initial state requires extensive exploration, but that there exists an intermediate subset of states, denoted $\mathcal{S}_{\text{good}}$, from which reward can be obtained reliably via standard on-policy sampling. Early in training, the agent is unaware of these states, as it has not yet experienced any reward. Guidance acts as a roll-in policy that steers the agent into $\mathcal{S}_{\text{good}}$, where learning signal becomes available and RL can proceed. On-policy RL from these states then learns an effective continuation policy in a region of the state space where reward is attainable.

Once such continuations are learned, the unguided policy no longer requires guidance to succeed from $\mathcal{S}_{\text{good}}$; it only needs to reach these states through its own behavior. Crucially, identifying whether a state belongs to $\mathcal{S}_{\text{good}}$ is itself difficult without evidence of success from that state. Training with guidance creates this evidence by learning successful completions conditioned on reaching $\mathcal{S}_{\text{good}}$. As a result, obtaining positive-reward traces from the initial state reduces to reaching some $s' \in \mathcal{S}_{\text{good}}$, after which the learned policy can already succeed. Once such traces are available, training further reinforces the behavior that leads to s' from the initial state. In contrast, unguided RL must discover both $\mathcal{S}_{\text{good}}$ and the successful behaviors from scratch, making exploration significantly more challenging.

Applying this mental model to LLMs. We now apply this mental model to LLMs. In an autoregressive MDP, a natural notion of state is the entire sequence of tokens produced so far. However, reasoning traces often exhibit substantial redundancy, suggesting that a more accurate notion of state for reasoning is the internal representation induced by a partial sequence, where newly generated segments can overwrite or revise earlier computation or attempts, resulting in revisiting similar states multiple times during a rollout. Guidance steers the model into internal states from which successful completions are more likely. The efficacy of this steering depends on whether the base model can follow the system instruction to build upon the guidance and comprehend the information it contains, even when the guidance itself consists of tokens that are unlikely under the base model. Models with strong ***instruction-following*** capabilities can benefit from this mechanism, and obtain non-trivial reward signal on guided versions of hard problems.

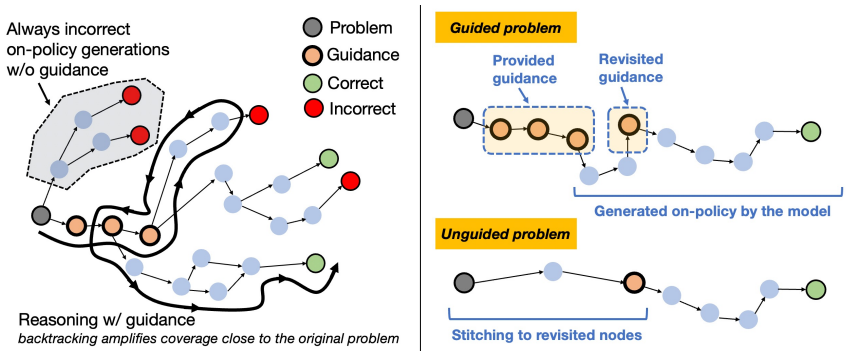

Figure 8: **Illustration of how reasoning structure and instruction following enable performance improvements.** Instruction-following capabilities of the base model enable the policy to pursue reasoning paths (shown as a thick black line) that reach regions of the solution space where reward can be attained. Self-verification and backtracking then allow the LLM to revisit states close to the initialization and construct successful continuations from there, amplifying coverage over states near the initial problem from which success is possible. By doing so, POPE reduces the challenge of attaining reward on the original unguided problem to reaching a nearby state from which successful rollouts have already been experienced on the guided problem.

Once the model has learned to solve the problem from states reached under guidance, the remaining challenge is to stitch these behaviors with those from the initial state. In general, it is unclear whether the base model would ever sample traces that perform computations similar to the provided guidance, especially when the guidance required to obtain a successful completion is long. In the MDP terminology above, the set of states $\mathcal{S}_{\text{good}}$ may itself be difficult to reach. In this regime, the structure of ***reasoning*** traces in long chain-of-thought models plays a central role in reducing the effective difficulty of reaching $\mathcal{S}_{\text{good}}$. Such models often self-verify, revisit earlier steps, and backtrack during generation (Figure 8, left). When these behaviors occur in guided rollouts, they expand the model's coverage over states closer to the problem that are otherwise unlikely to be sampled under base model rollouts without guidance. As a result, RL training begins to observe reward not only from the guided states induced by the oracle prefix, but also from a neighborhood of states that the model can plausibly reach without guidance.

As a result, learning on the unguided problem becomes plausible. Rather than discovering reward from scratch, the policy only needs to explore to reach nearby states that were already visited during guided rollouts and the structure of reasoning amplifies ***overlap*** under function approximation (Figure 8, right). *This explains why* `POPE` *enables transfer from guided to unguided problems.* See Appendix D for an extended discussion and further intuition about the overlap mechanism. Notably, we find that this transfer occurs even when ***a fixed segment of guidance*** is used for training, despite conventional wisdom of coverage Chang et al. (2024) demanding more segments for a transfer of performance to the unguided version.

5.2 Empirically Validating the Stitching and Overlap Hypothesis

As discussed above, despite using a fixed guidance segment, `POPE` enables transfer because backtracking and revision behaviors expand coverage over nearby states that an unguided rollout can plausibly reach. We now test our model above via an intervention that selectively reduces overlap between guided and unguided rollouts. If this overlap is important, then discouraging the model from revisiting earlier parts of the guided solution should weaken transfer from guided to unguided problems.

Experimental setup. We modify the system instruction (shown below) in `POPE` instructing the model to continue solving the problem in a guided rollout, *without* restating, paraphrasing, or recomputing any part of the guidance. This instruction encourages the model to treat the guidance as a silent scaffold and to avoid backtracking to intermediate steps that would otherwise be revisited.

Modified `POPE` System Instruction

You are given a problem and a partial solution. Your task is to infer what reasoning has already been completed and continue solving the problem **without repeating, paraphrasing, or referencing any part of the partial response**. You must **not** restate earlier steps, summarize them, or quote them in any form. Begin directly from the next logical step that has not yet been completed.

Important: *Use the information from the partial response silently. Do not copy, rephrase, or explicitly mention anything from it. Your continuation must be logically consistent with what has already been done. Show your reasoning step by step (only the new steps), and present the final answer using* $\boxed{\cdot}$ *notation.*

Results. As shown in Figure 9, this modification to the system instruction shifts performance in a manner consistent with the stitching/overlap mental model. The modified instruction improves performance on the *guided* version of the hard problems, consistent with making the RL problem easier conditional on guidance. However, it reduces transfer to the *unguided* problems, yielding a lower pass@32 score compared to the default instruction used in `POPE`. In effect, the intervention biases learning toward behaviors that succeed only when guidance is present, rather than behaviors that transfer to the unguided setting. This provides evidence that overlap between guided and unguided state visitation, mediated by backtracking and revisiting intermediate steps, is an important component of `POPE`'s efficacy.

Qualitative evidence. We also compare model outputs produced by models trained with the default and modified instructions. As shown in Table 1, under the default instruction, the unguided solution learned by `POPE` often reflects concepts and intermediate steps that appear in guided traces (see Appendix F for the full problem, partial oracle solution, and representative guided/unguided rollouts), suggesting that the model

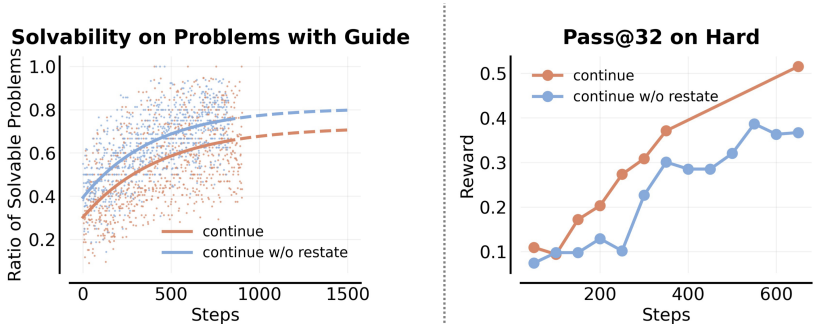

Figure 9: **Left: solvability (pass@8)** and **Right: pass@32 scores on the guided and unguided versions of the hard prompt**. Forcing the model to continue without restating or revisiting information in the guidance solves more problems with guidance, presumably because it simplifies the RL problem conditioned on the guidance. However, this system instruction also achieves a worse pass@32 score on the unguided version of the hard problem, indicating reduced transfer from guided to unguided settings, supporting our mental model.

stitches together reasoning learned under guidance. In contrast, with the modified instruction, the unguided solution no longer resembles the guided trace and instead follows a distinct solution path, exhibiting minimal reuse of concepts present in the guidance. This pattern is consistent with the intervention reducing overlap and thereby weakening the transfer mechanism.

Criterion	Training w/ Default Instruction	Training w/ Modified Instruction
Uses inequality structure	Yes	Yes
Uses cyclic indexing meaningfully	Conceptual	Nominal only
Uses $\lambda = \max S$ idea	Yes	Weak
Follows partial response direction	Partial (extremal patterns)	No
Uses geometric sequence ($a_i = x^{i-1}$)	No	No
Explores extremal constructions	Yes (patterns, ratios)	No
Uses ($n \geq 4k$) meaningfully	Partial	Mention only
Depth of mathematical reasoning	Medium	Low
True continuation of the partial response	Partial	No

Table 1: **Comparison of unguided solutions produced by models trained with the POPE system instruction and the modified instruction on unguided and guided augmentations on hard problems**. Rollouts with the POPE system instruction replicate several aspects of the guidance, indicating successful transfer. In contrast, rollouts from the modified system instruction show far lower resemblance to the guidance, suggesting that this instruction suppresses the stitching effect.

6 Experimental Evaluation

The goal of our experiments is to evaluate the effectiveness of POPE in solving hard problems during training and its impact on downstream performance. To this end, we address three main questions in this section: **(1)** Does POPE improve the solvability of hard problems during training? **(2)** Does solving hard problems via POPE improve performance on (potentially) out-of-distribution evaluation benchmarks? **(3)** How does POPE compare to approaches that use oracle solutions as training targets, such as supervised fine-tuning on oracle solutions? We have already presented several diagnostic analyses in earlier sections, including how POPE mitigates ray interference (Figure 5) and the role played by the system instruction in enabling transfer (Section 5.2). We therefore focus mainly on performance results in this section.

Experimental setup. We run all experiments using the `Qwen3-4B-Instruct-2507` base model. We train using GRPO with a maximum output length of 16384 tokens (recommended for this base model (Yang et al., 2025)) and use a sampling temperature of 0.8. For most of our experiments, we use pipeline-rl (Piché et al., 2025), an asynchronous, streaming RL framework in our experiments, where we set the clip ratio of token-level importance weights to be 5.0 on the higher end, and 0.0 on the lower end. We also implemented POPE on verl (Sheng et al., 2025), where we found similar preliminary results with 1 off-policy update step, and clip ratios of 0.2 and 0.28 on the lower and higher side respectively. Additional implementation details, hyperparameters, and details of our datasets are provided in Appendix E. To construct the hard problem set, we select problems from (Yu et al., 2025), OmniMath (levels 5–8) (Gao et al., 2024), and AceReason (Chen et al., 2025b). A problem is included only if the base model fails to produce any correct rollout under aggressive evaluation, using $k = 128$ parallel samples and a 32k-token budget, ensuring that all selected problems lie in a near-zero–reward regime. Some examples are in Appendix G.

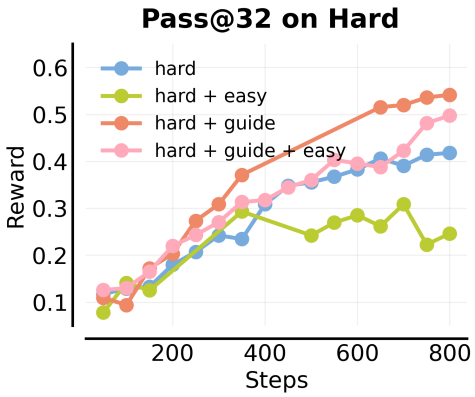

Figure 10: **Pass@32 on the hard problem set evaluated with a 32k token budget.** Mixing in easy problems (green) causes a plateau in pass@32 over training, even though pass@32 continues to improve when training only on the hard set. This drop reflects ray interference caused by the easy data. In contrast, incorporating guidance in the form of a human-written prefix improves pass@32 consistently throughout training (red/pink), indicating that POPE mitigates ray interference.

Result 1: POPE enables solving more hard problems. We first evaluate the efficacy of POPE during training in Figure 10, where we evaluate the pass@32 performance on the training set under a much larger token budget of 32,768 tokens. Note that this evaluation configuration differs from training (which uses 8 rollouts at 16,384 token length), and hence it stress tests if POPE actually makes more progress on the training problems. Observe that POPE ("hard + guide") solves more problems from the hard set compared to any

Approach	Hard problems		AIME 2025		HMMT 2025	
	pass@1	pass@16	pass@1	pass@16	pass@1	pass@16
Base model						
Qwen3-4B-Instruct	0.57	7.42	48.13	77.29	29.06	52.99
Hard problems only						
+ hard	13.55	32.89	49.58	81.43	31.04	63.79
+ hard + guide (POPE)	15.50 +14%	42.53 +29%	53.12 +7%	82.61 +1%	37.81 +22%	67.49 +6%
+ hard (Full-oracle SFT)	2.00 −85%	12.37 −62%	33.89 −32%	64.12 −21%	24.50 −21%	48.09 −25%
+ hard (Prefix + rejection-sampled SFT)	5.14 −62%	24.50 −26%	38.12 −23%	77.62 −5%	30.08 −3%	50.91 −20%
Hard (256) + Easy (256)						
+ hard + easy	8.22	23.81	57.19	82.50	37.19	62.81
+ hard + guide + easy (POPE)	14.32 +74%	38.93 +63%	58.75 +3%	83.87 +2%	38.12 +3%	67.15 +7%
Hard (256) + Easy (1K)						
+ hard + 1K easy problems	2.24	25.12	61.88	83.79	37.03	60.07
+ hard + guide + 1K easy problems (POPE)	13.98 +524%	36.42 +45%	62.01 +0.2%	84.31 +0.6%	40.45 +9%	70.38 +17%

Table 2: **Pass@1 and pass@16 scores** on the hard set and standardized benchmarks (AIME2025 and HMMT2025), with relative gains highlighted in red and relative losses highlighted in yellow. Incorporating guidance via POPE substantially improves performance on the hard problems while also improving performance on standardized benchmarks. The performance gains on hard problems are still preserved when a larger number of easy problems are mixed in. Mixing in easy problems improves performance on the easier AIME2025 benchmark, but training via POPE on hard problems enables improvement on the harder HMMT2025 benchmark.

other configuration. While mixing in easy problems ("hard + easy") results in a performance plateau on hard problems due to interference and this approach saturates at a lower pass@32 performance compared to training on hard problems alone ("hard"), no such performance plateau is observed for "hard + guide", which continues to improve as more steps of RL training are done.

Result 2: Training on broad problem mixtures with POPE. We further evaluate POPE in a more practically relevant setting that mixes hard problems with varying amounts of easy problems, mimicking the broad training mixtures commonly used in practice. We report results in Table 2. Concretely, we train on mixtures of "hard + guide" and the easy problem set, and compare against corresponding mixtures without guidance. As shown in Figure 10 and Table 2, mixing in an equal number of easy problems without guidance significantly degrades performance on the hard set (e.g., "hard + easy" vs. "hard") due to the interference issue discussed earlier. In contrast, *introducing guidance via POPE substantially mitigates this problem.* For instance, even when easy problems are present, "hard + guide + easy" achieves a pass@1 of 14.3% and pass@16 of 38.9% on the hard set, which closely matches "hard + guide" alone (15.5% pass@1 and 42.5% pass@16), and does better than having no guidance.

This trend persists even when we scale up the amount of easy problems in the overall prompt set used for RL training. For instance, adding 1K easy problems without guidance severely harms hard-set performance (2.2% pass@1), the corresponding mixture with guidance ("hard + guide + 1K easy") recovers strong performance (14.0% pass@1, 36.4% pass@16), demonstrating that POPE enables robust learning on hard problems with diverse training mixtures.

Result 3: Performance on standardized benchmarks. Table 2 reports performance on standardized benchmarks (AIME 2025 and HMMT 2025). Although guidance in POPE is designed to target learning on hard training problems, it consistently improves downstream benchmark performance as well. In particular, "hard + guide + 1K easy" achieves the strongest overall results, obtaining the best pass@1 and pass@16 on both AIME 2025 and HMMT 2025 under a 32,768-token evaluation budget. These gains arise because the easy dataset overlaps in difficulty with standardized benchmarks, while POPE enables learning on harder problems despite a substantial ratio of easy problems in the mixture. Most notably, the improvement from adding guided hard problems is larger on HMMT 2025, a benchmark typically considered harder than AIME 2025 (e.g., "hard + guide + 1K easy" vs. "hard + 1K easy"; +3.4% in pass@1 and +10.0% on pass@16). This comparison highlights the benefit of maintaining effective learning on hard problems, even though there are sufficiently many easy problems in the training mixture. Overall, these results demonstrate that POPE not only improves optimization on difficult training instances, but also scales robustly to large, heterogeneous data mixtures commonly used in practice.

Result 4: Comparison with methods that use oracle solutions as training targets. Finally, we compare the performance and optimization behavior of POPE with prior approaches that use oracle solutions directly as training targets. Specifically, we compare against two methods that apply supervised fine-tuning (SFT) on privileged information followed by standard RL. We also attempted to compare to LUFFY (Yan et al., 2025a), which incorporates the oracle solution directly as a rollout during RL but were unable to make it train stably on our hard problems with human reference solutions; hence we skip this comparison for now. The two SFT baselines are: **(a)** SFT directly on the oracle solution ("Full-oracle SFT"), and **(b)** SFT on the prefix of the oracle solution followed by a successful on-policy completion obtained via rejection sampling from the base model (called "Prefix + rejection-sampled SFT"). Note that the oracle prefixes used by these baselines are identical to those employed by POPE.

As shown in Table 2, SFT on full oracle solutions severely degrades performance across all evaluations. On the hard problem set, this approach reduces pass@1 from 13.6% ("+ hard") to 2.0% and pass@16 from 32.9% to 12.4%. This performance degeneration also manifests on standardized benchmarks, with AIME 2025 pass@1 decreasing from 49.6% to 33.9% and HMMT 2025 pass@16 falling from 63.8% to 48.1%. This is perhaps expected since oracle solutions exhibit fundamentally different reasoning styles and cloning such off-policy data disrupt the model's own reasoning capabilities (Yang et al., 2026).

The rejection-sampled SFT variant avoids catastrophic collapse but still underperforms RL training on hard problems. Specifically, it achieves only 5.1% pass@1 and 24.5% pass@16 on the hard set, substantially below both "+ hard" (13.6% / 32.9%) and "+ hard + guide" (15.5% / 42.5%). While its performance on easier benchmarks such as AIME 2025 remains close to the base model, it even underperforms naïve RL training on hard problems starting from the base model ("hard"). In Appendix C, we further show that applying RL on top of the SFT warm start does not improve exploration, yielding virtually no gains in hard-problem solvability compared to standard RL ("hard").

Takeaways: POPE enables robust learning on hard problems

- POPE consistently improves the solvability of hard problems, avoiding ray interference.
- POPE preserves strong performance on hard problems even when training with easy problems.
- Training on hard problems via POPE also improves performance on standardized benchmarks.

7 Related Work

We tackle exploration on hard problems in regimes where naïvely scaling on-policy RL compute yields little progress. At its core, this is an *exploration* challenge, since algorithmic interventions are required to discover high-reward trajectories. We therefore briefly discuss related approaches for improving exploration, including methods that add explicit exploration bonuses and methods that learn from off-policy traces that cover high-reward regions. We also discuss connections with ideas from RL theory.

Exploration methods in RL. Recent work has shown that reinforcement learning can substantially improve LLM reasoning by reinforcing long-horizon behaviors such as self-correction and reflection (Liu et al., 2025; Luo et al., 2025; Qu et al., 2024; Gandhi et al., 2025). However, multiple studies observe that on-policy RL tends to over-optimize already-solvable problems, leaving harder problems unsolved (Yue et al., 2025; Zhao et al., 2025). At the population level, this often manifests as declining pass@k despite increasing training reward. As we show in this work, this behavior can be explained by ray interference (Schaul et al., 2019), which biases optimization toward states where reward is already attainable, creating a structural barrier to learning on hard problems. To address ray interference, our approach POPE makes it possible to make more "uniform" updates on all problems by incorporating guidance derived from a human-written solution (available in most datasets).

Several prior approaches attempt to mitigate over-sharpening using exploration bonuses (Gao et al., 2025; Wang et al., 2025b; Hamid et al., 2025; Song et al., 2025), objectives that directly optimize pass@k (Chow et al., 2024; Balashankar et al., 2025), or curricula and prompt mixtures that rely on transfer from easier problems (Setlur et al., 2025b; Sun et al., 2025; Liu et al., 2025; Hu et al., 2025). However, these methods fundamentally depend on sampling at least one correct rollout. When pass@1 is near zero, token-level

exploration provides no useful signal, pass@k optimization reduces to a monotonic transformation of pass@1, and transfer from easier problems fails due to interference. Our experiments in Section 3 showcase failure modes of a representative subset of these approaches in addressing the challenge of learning on hard problems.

Our approach is conceptually related to classical RL results showing that access to intermediate states or resets can significantly reduce the sample complexity of exploration (Jaksch et al., 2010; Azar et al., 2017; Kakade & Langford, 2002; Agarwal et al., 2021), as well as modern methods such as Go-Explore (Ecoffet et al., 2019; 2020) that revisit previously discovered states. Unlike these methods, POPE does not perform hard resets or rely on explicit state visitation. Instead, we leverage the instruction-following capabilities of LLMs to steer on-policy rollouts into analogous internal states that enable learning signal, and crucially, allow behaviors learned under guidance to transfer back to unguided problems. To our knowledge, prior work does not systematically study this guided-to-unguided transfer mechanism or explain why guided training can improve performance when guidance is absent at test time.

Although we do not establish formal theoretical guarantees for LLMs in this work, we note that POPE may violate several standard assumptions underlying these results, including realizability of oracle demonstrations, uniform coverage via random prefix sampling at every learning step, and the absence of update interference, which plays a central role in our analysis. This suggests that new abstractions may be required to theoretically study exploration in LLMs, an important direction for future work.

Training LLMs on off-policy traces. Motivated by the limitations of on-policy RL, several works propose updating LLM policies using human- or oracle-provided reasoning traces (Lightman et al., 2023; Corrado et al., 2024). While effective in some settings, methods that rely on supervision from a teacher model are inherently bounded by the teacher's capacity (Agarwal et al., 2024b). Moreover, stable learning from off-policy traces often requires additional mechanisms such as reward shaping (Yan et al., 2025b), entropy control (Wang et al., 2025a), and careful hyperparameter tuning (Zhang et al., 2025a).

A more fundamental limitation is that suitable off-policy reasoning traces are not readily available for many hard problems. Although human-written solutions exist for most training prompts and can often be rephrased into more effective formats, producing long chains of thought that align with how models actually reason remains challenging (Zelikman et al., 2022). This mismatch between off-policy traces and the model's native reasoning behavior can lead to unstable learning dynamics, including entropy explosion or entropy collapse depending on the SFT configuration, as discussed conceptually in Section 4. These limitations motivate approaches that avoid using off-policy traces as direct training targets.

Most related prior works. The most closely related prior and concurrent works that address learning on hard problems leverage human or oracle data to extract subgoals, plans, or abstractions, which are then used to inform rollout generation in on-policy RL (Hong et al., 2025; Qu et al., 2025; Li et al., 2025; Chen et al., 2025a). Our work shares a similar high-level philosophy, but shows that simply conditioning on prefixes of past solutions is sufficient to enable learning on hard problems. Prior work that also directly utilizes partial solutions (Amani et al., 2025; Zhang et al., 2025b) primarily studies non-reasoning models that produce short responses and focuses on problems that are not too hard. In particular, with non-reasoning models, Amani et al. (2025) requires adaptively tuning the length of the partial solution for on-policy generation, whereas we find that POPE does not require such curricula, since backtracking and recovery behaviors naturally provide coverage over states close to initialization.

Moreover, to the best of our knowledge, no prior work systematically studies why unguided training is difficult on hard problems, identifies the limitations of existing approaches, and establishes the role of guided training in enabling transfer to settings where guidance is absent. It is in fact unclear many times from prior work, why a guided approach is needed in the first place and simple adjustments to learning configurations of existing algorithms are insufficient. In contrast, we identify the ray interference problem, show that it cannot be solved by several token-level exploration or pass@k optimization approaches, and develop a mental model under which training on augmented problems enables transfer to unguided problems (Section 5.2). We validate all these insights through targeted empirical interventions.

8 Discussion and Perspectives on Future Work

In this paper, we study a fundamental limitation of on-policy RL for LLMs: the inability to learn from hard problems when no correct rollouts are sampled. We show that standard remedies for exploration, including entropy bonuses, optimistic updates, pass@k optimization, and curricula over easy problems, fail to address this challenge due to sharpening and ray interference. To overcome this limitation, we introduce **Privileged On-Policy Exploration** (POPE), a framework that leverages privileged information in the form of partial oracle solutions to guide on-policy exploration without using these solutions as training targets. By conditioning rollouts on solution prefixes along with a system instruction to build on this prefix, and training on a mixture of guided and unguided problems, POPE enables the model to obtain learning signal on hard problems and acquire reasoning behaviors that transfer back to unguided settings. We provide empirical results showing that this transfer is enabled by a synergy between instruction-following and reasoning behaviors, and demonstrate that POPE substantially expands the set of solvable hard problems where existing RL approaches fail.

There are several directions for future work. **First**, formalizing the mechanism by which POPE improves exploration on hard problems is an important open question. Our experiments suggest that POPE improves performance by leveraging the instruction-following capabilities of the underlying LLM to follow and build upon oracle solutions. How can this notion be quantified theoretically? From a practical perspective, how can these instruction-following capabilities be further amplified and systematically leveraged to improve reasoning? **Second**, there exists a class of even harder problems for which models fundamentally lack the knowledge required to solve the task. In such cases, conditioning on an oracle solution and relying on instruction following alone may be insufficient to improve performance, and deriving explicit training targets from the oracle may be necessary. How should such training targets be constructed? How can we mitigate challenges associated with memorization and pathological optimization in this regime? We believe that methods from off-policy RL, for example, training explicit value functions (Setlur et al., 2024; 2025a)) or implicitly modeling them via interventions (Yang et al., 2026) likely provide a natural starting point for answering this question. **Third**, our work highlights the role of ray interference in inhibiting learning on heterogeneous prompt mixtures. Ray interference is not unique to our prompt sets, is a more general phenomenon that is likely present a bigger prompt sets as well. What factors determine the severity of this interference? How does it depend on the model's pre-training or mid-training procedures? Can we predict when ray interference will arise before running RL training? Addressing these questions will lead to more robust and predictable RL recipes that continue to make progress without prematurely plateauing on heterogeneous dataset mixtures.

References

Alekh Agarwal, Sham M Kakade, Jason D Lee, and Gaurav Mahajan. On the theory of policy gradient methods: Optimality, approximation, and distribution shift. *Journal of Machine Learning Research*, 22 (98):1–76, 2021.

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=3zKtaqxLhW.

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes, 2024b. URL https://arxiv.org/abs/2306.13649.

Mohammad Hossein Amani, Aryo Lotfi, Nicolas Mario Baldwin, Samy Bengio, Mehrdad Farajtabar, Emmanuel Abbe, and Robert West. Rl for reasoning by adaptively revealing rationales, 2025. URL https://arxiv.org/abs/2506.18110.

Mohammad Gheshlaghi Azar, Ian Osband, and Rémi Munos. Minimax regret bounds for reinforcement learning. In *International Conference on Machine Learning*, pp. 263–272. PMLR, 2017.

Ananth Balashankar, Ziteng Sun, Jonathan Berant, Jacob Eisenstein, Michael Collins, Adrian Hutter, Jong Lee, Chirag Nagpal, Flavien Prost, Aradhana Sinha, Ananda Theertha Suresh, and Ahmad Beirami. Infalign: Inference-aware language model alignment, 2025. URL https://arxiv.org/abs/2412.19792.

Jonathan D Chang, Wenhao Shan, Owen Oertell, Kianté Brantley, Dipendra Misra, Jason D Lee, and Wen Sun. Dataset reset policy optimization for rlhf. *arXiv preprint arXiv:2404.08495*, 2024.

Justin Chih-Yao Chen, Becky Xiangyu Peng, Prafulla Kumar Choubey, Kung-Hsiang Huang, Jiaxin Zhang, Mohit Bansal, and Chien-Sheng Wu. Nudging the boundaries of llm reasoning. *arXiv preprint arXiv:2509.25666*, 2025a.

Yang Chen, Zhuolin Yang, Zihan Liu, Chankyu Lee, Peng Xu, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Acereason-nemotron: Advancing math and code reasoning through reinforcement learning. *arXiv preprint arXiv:2505.16400*, 2025b.

Yinlam Chow, Guy Tennenholtz, Izzeddin Gur, Vincent Zhuang, Bo Dai, Sridhar Thiagarajan, Craig Boutilier, Rishabh Agarwal, Aviral Kumar, and Aleksandra Faust. Inference-aware fine-tuning for best-of-n sampling in large language models. *arXiv preprint arXiv:2412.15287*, 2024.

Nicholas E. Corrado, Yuxiao Qu, John U. Balis, Adam Labiosa, and Josiah P. Hanna. Guided data augmentation for offline reinforcement learning and imitation learning, 2024. URL https://arxiv.org/abs/2310.18247.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*, 2019.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. First return then explore, 2020.

Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D. Goodman. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars, 2025. URL https://arxiv.org/abs/2503.01307.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Daoguang Zan, Shanghaoran Quan, Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. Omni-math: A universal olympiad level mathematic benchmark for large language models, 2024. URL https://arxiv.org/abs/2410.07985.

Jingtong Gao, Ling Pan, Yejing Wang, Rui Zhong, Chi Lu, Qingpeng Cai, Peng Jiang, and Xiangyu Zhao. Navigate the unknown: Enhancing llm reasoning with intrinsic motivation guided exploration, 2025. URL https://arxiv.org/abs/2505.17621.

Jubayer Ibn Hamid, Ifdita Hasan Orney, Ellen Xu, Chelsea Finn, and Dorsa Sadigh. Polychromic objectives for reinforcement learning. *arXiv preprint arXiv:2509.25424*, 2025.

Joey Hong, Anca Dragan, and Sergey Levine. Planning without search: Refining frontier llms with offline goal-conditioned rl. *arXiv preprint arXiv:2505.18098*, 2025.

Jian Hu, Mingjie Liu, Ximing Lu, Fang Wu, Zaid Harchaoui, Shizhe Diao, Yejin Choi, Pavlo Molchanov, Jun Yang, Jan Kautz, et al. Brorl: Scaling reinforcement learning via broadened exploration. *arXiv preprint arXiv:2510.01180*, 2025.

Thomas Jaksch, Ronald Ortner, and Peter Auer. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11(Apr):1563–1600, 2010.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *International Conference on Machine Learning (ICML)*, volume 2, 2002.

Devvrit Khatri, Lovish Madaan, Rishabh Tiwari, Rachit Bansal, Sai Surya Duvvuri, Manzil Zaheer, Inderjit S Dhillon, David Brandfonbrener, and Rishabh Agarwal. The art of scaling reinforcement learning compute for llms. *arXiv preprint arXiv:2510.13786*, 2025.

Jiazheng Li, Hongzhou Lin, Hong Lu, Kaiyue Wen, Zaiwen Yang, Jiaxuan Gao, Yi Wu, and Jingzhao Zhang. Questa: Expanding reasoning capacity in llms via question augmentation. *arXiv preprint arXiv:2507.13266*, 2025.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023.

Mingjie Liu, Shizhe Diao, Ximing Lu, Jian Hu, Xin Dong, Yejin Choi, Jan Kautz, and Yi Dong. Prorl: Prolonged reinforcement learning expands reasoning boundaries in large language models, 2025. URL https://arxiv.org/abs/2505.24864.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl, 2025. Notion Blog.

Seohong Park, Kevin Frans, Sergey Levine, and Aviral Kumar. Is value learning really the main bottleneck in offline rl? *Advances in Neural Information Processing Systems*, 37:79029–79056, 2024.

Alexandre Piché, Ehsan Kamalloo, Rafael Pardinas, Xiaoyin Chen, and Dzmitry Bahdanau. Pipelinerl: Faster on-policy reinforcement learning for long sequence generation. *arXiv preprint arXiv:2509.19128*, 2025.

Yuxiao Qu, Tianjun Zhang, Naman Garg, and Aviral Kumar. Recursive introspection: Teaching language model agents how to self-improve. *arXiv preprint arXiv:2407.18219*, 2024.

Yuxiao Qu, Anikait Singh, Yoonho Lee, Amrith Setlur, Ruslan Salakhutdinov, Chelsea Finn, and Aviral Kumar. Learning to discover abstractions for LLM reasoning. In *ICML 2025 Workshop on Programmatic Representations for Agent Learning*, 2025. URL https://openreview.net/forum?id=zwEUOOKT8G.

Tom Schaul, Diana Borsa, Joseph Modayil, and Razvan Pascanu. Ray interference: a source of plateaus in deep reinforcement learning. *CoRR*, abs/1904.11455, 2019. URL http://arxiv.org/abs/1904.11455.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Pier Giuseppe Sessa, Robert Dadashi, Léonard Hussenot, Johan Ferret, Nino Vieillard, Alexandre Ramé, Bobak Shariari, Sarah Perrin, Abe Friesen, Geoffrey Cideron, Sertan Girgin, Piotr Stanczyk, Andrea Michi, Danila Sinopalnikov, Sabela Ramos, Amélie Héliou, Aliaksei Severyn, Matt Hoffman, Nikola Momchev, and Olivier Bachem. Bond: Aligning llms with best-of-n distillation, 2024. URL https://arxiv.org/abs/2407.14622.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for llm reasoning. *arXiv preprint arXiv:2410.08146*, 2024.

Amrith Setlur, Yuxiao Qu, Matthew Yang, Lunjun Zhang, Virginia Smith, and Aviral Kumar. Optimizing llm test-time compute involves solving a meta-rl problem. https://blog.ml.cmu.edu/2025/01/08/optimizing-llm-test-time-compute-involves-solving-a-meta-rl-problem/, 2025a. CMU MLD Blog.

Amrith Setlur, Matthew Y. R. Yang, Charlie Snell, Jeremy Greer, Ian Wu, Virginia Smith, Max Simchowitz, and Aviral Kumar. e3: Learning to explore enables extrapolation of test-time compute for llms, 2025b. URL https://arxiv.org/abs/2506.09026.

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

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 2025.

Yuda Song, Julia Kempe, and Remi Munos. Outcome-based exploration for llm reasoning, 2025. URL https://arxiv.org/abs/2509.06941.

Yiyou Sun, Yuhan Cao, Pohao Huang, Haoyue Bai, Hannaneh Hajishirzi, Nouha Dziri, and Dawn Song. Rl grokking recipe: How does rl unlock and transfer new algorithms in llms? *arXiv preprint arXiv:2509.21016*, 2025.

Christian Walder and Deep Karkhanis. Pass@ k policy optimization: Solving harder reinforcement learning problems. *arXiv preprint arXiv:2505.15201*, 2025.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, et al. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.01939*, 2025a.

Yiping Wang, Qing Yang, Zhiyuan Zeng, Liliang Ren, Liyuan Liu, Baolin Peng, Hao Cheng, Xuehai He, Kuan Wang, Jianfeng Gao, Weizhu Chen, Shuohang Wang, Simon Shaolei Du, and Yelong Shen. Reinforcement learning for reasoning in large language models with one training example, 2025b. URL https://arxiv.org/abs/2504.20571.

Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. Learning to reason under off-policy guidance. *arXiv preprint arXiv:2504.14945*, 2025a.

Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. Learning to reason under off-policy guidance, 2025b. URL https://arxiv.org/abs/2504.14945.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Matthew Y. R. Yang, Hao Bai, Ian Wu, Gene Yang, Amrith Setlur, and Aviral Kumar. Int: Self-proposed interventions enable credit assignment in llm reasoning, 2026. URL https://arxiv.org/abs/2601.14209.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model?, 2025. URL https://arxiv.org/abs/2504.13837.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

Wenhao Zhang, Yuexiang Xie, Yuchang Sun, Yanxi Chen, Guoyin Wang, Yaliang Li, Bolin Ding, and Jingren Zhou. On-policy rl meets off-policy experts: Harmonizing supervised fine-tuning and reinforcement learning via dynamic weighting, 2025a. URL https://arxiv.org/abs/2508.11408.

Xuechen Zhang, Zijian Huang, Yingcong Li, Chenshun Ni, Jiasi Chen, and Samet Oymak. Bread: Branched rollouts from expert anchors bridge sft & rl for reasoning. *arXiv preprint arXiv:2506.17211*, 2025b.

Rosie Zhao, Alexandru Meterez, Sham Kakade, Cengiz Pehlevan, Samy Jelassi, and Eran Malach. Echo chamber: Rl post-training amplifies behaviors learned in pretraining, 2025. URL https://arxiv.org/abs/2504.07912.

Appendices

A Why Does Entropy Increase with a Higher Clip Ratio?

We now briefly attempt to understand the mechanism behind our finding that increasing the clip ratio, as in DAPO (Yu et al., 2025), can lead to higher next-token entropy even without an explicit entropy regularizer. Off-policy negative samples generally push the model toward higher token entropy on average, as shown theoretically and empirically by Setlur et al. (2025b). Increasing the positive clip ratio amplifies this effect on hard problems by allowing more optimistic policy updates on low-likelihood positive traces. By definition, the base model is unlikely to sample a successful trace on a hard problem, so positive trajectories are rare and assigned very low probability under the current policy. A larger clip ratio permits updates that attempt to increase the likelihood of tokens appearing in rare traces; however, with only one or a few gradient steps, the model cannot fully reallocate probability mass onto them. As a result, some tokens in rare positive traces receive a disproportionate increase in probability mass. In reasoning models, these are often tokens that signal a shift in the reasoning trajectory (e.g., "Wait", "maybe"), which are known to induce high-entropy next-token distributions (Wang et al., 2025a). Instead, it reduces confidence in previously high-probability tokens without successfully fitting the positive trace, resulting in a flatter next-token distribution and increased entropy. Repeating this process across training steps leads to an entropy explosion. Had the update been able to fully concentrate mass on the positive trace, or been suppressed entirely, this entropy amplification would not occur.

B Details of Pass@k Policy Optimization

In Section 3.2, we experimented with pass@k optimization to see if it can solve the ray interference problem by not over-optimizing pass@1. Recall that ray interference is a direct consequence of the competition between optimizing reward on problems where rewards can already be attained and optimizing reward on new problems. Naturally, one might except that if only optimize pass@k for a higher value of k (e.g., $k = 8$), then we may no longer run into the issue of over-optimizing pass@1 on some problems at the cost of performance and increase the hardness of sampling a correct rollout on the others. Here, we detail the objective we use for pass@k optimization from prior work (Walder & Karkhanis, 2025).

Pass@k estimator. Given a prompt $\mathbf{x}$, we sample $n \geq k$ i.i.d. rollouts $\mathbf{y}_1, \ldots, \mathbf{y}_n \sim \pi_\theta(\cdot \mid \mathbf{x})$ and evaluate their correctness $f_i \triangleq R(\mathbf{x}, \mathbf{y}_i) \in \{0, 1\}$. Let $c \triangleq \sum_{i=1}^{n} f_i$ denote the number of correct rollouts in the batch. An unbiased estimator of pass@k objective is given by:

$$\rho(n, c, k) \;\triangleq\; 1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}, \tag{4}$$

which estimates the probability that at least one of k uniformly sampled rollouts (without replacement) is correct. Intuitively, $\rho(n, c, k)$ increases monotonically with the number of observed successes c, and reduces to pass@1 when $k = 1$.

In our setting, this estimator is applied at the level of individual prompts, and the overall training objective is to maximize the expected pass@k score across the training distribution:

$$\mathcal{J}_k(\theta) \;\triangleq\; \mathbb{E}_{\mathbf{x}\sim\rho}\big[\mathbb{E}_{\mathbf{y}_1,\ldots,\mathbf{y}_n\sim\pi_\theta(\cdot|\mathbf{x})}\big[\rho(n, c(\mathbf{x}), k)\big]\big]\,, \tag{5}$$

where ρ denotes the empirical distribution over training prompts and $c(\mathbf{x})$ is the number of correct rollouts for prompt $\mathbf{x}$.

Unbiased pass@k gradient estimator. With this definition, we now present the policy gradient term that we use from Walder & Karkhanis (2025). Given a prompt $\mathbf{x}$, sample n i.i.d. rollouts $\mathbf{y}_1, \ldots, \mathbf{y}_n \sim \pi_\theta(\cdot \mid \mathbf{x})$ with correctness labels $f_i \triangleq R(\mathbf{x}, \mathbf{y}_i) \in \{0, 1\}$ and let $c \triangleq \sum_{i=1}^{n} f_i$ be the number of correct samples. An unbiased estimator of the gradient of the (per-prompt) pass@k objective can be written as a weighted policy-gradient update:

$$\widehat{\nabla_\theta} \;=\; \sum_{i=1}^{n} r_i \, \nabla_\theta \log \pi_\theta(\mathbf{y}_i \mid \mathbf{x}), \tag{6}$$

where the weights are

$$r_i = \begin{cases} \frac{k}{n}, & \text{if } f_i = 1, \\ \frac{k}{n}\,\rho(n-1, c, k-1), & \text{if } f_i = 0, \end{cases} \tag{7}$$

and $\rho(\cdot)$ is the unbiased pass@k estimator from equation 4:

$$\rho(n, c, k) = 1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}. \tag{8}$$

We can now treat each of the values in Equation 7 as "reward" and run standard RL to optimize it. We chose to use this instantiation of the pass@k policy optimization objective over the variant of Chow et al. (2024) because this version is simpler in terms of implementation.

C Why does SFT + RL Not Improve Solvability on Hard Problems?

Although warmstarting with SFT is often effective when high-quality expert traces are available, it fundamentally alters the reasoning behaviors of the base model, leading to poor performance in Table 2. Even when SFT is restricted to a short prefix of the oracle solution (as used for POPE), followed by a correct on-policy reasoning trace obtained via rejection sampling, SFT concentrates probability mass onto a narrow set of token-level distributions. As shown in Figure 11, initializing RL from such an SFT-trained checkpoint leads to a collapse in token entropy and substantially worse solvability compared to our approach (POPE; "hard + guide").

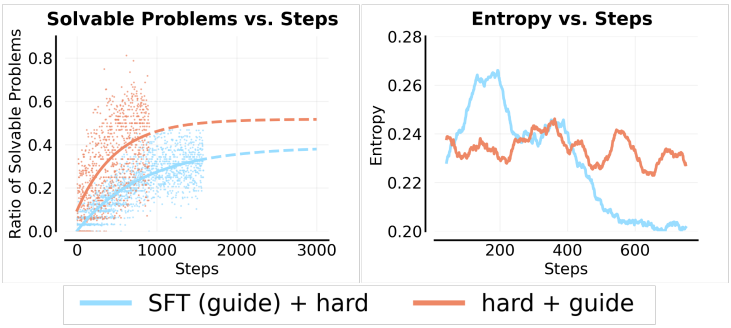

Figure 11: **Effect of SFT warmstarts on solvability and entropy. Left:** Fraction of solvable hard problems during training. Warm-starting RL from an SFT model trained on synthetically generated, rejection-sampled traces results in consistently worse solvability than our approach. **Right:** SFT warmstarts induce a persistent entropy collapse, leading to reduced exploration and suboptimal on-policy learning.

More broadly, low-entropy initialization is especially harmful in sparse-reward regimes. On hard problems, successful trajectories are rare and lie in the tail of the policy's distribution; once entropy collapses, on-policy sampling rarely explores alternative reasoning paths, and policy-gradient updates become dominated by near-duplicate prefixes. As a result, the policy becomes trapped in a locally consistent but globally suboptimal mode, preventing progress on previously unsolved problems.

D Extended Discussion of the Overlap Hypothesis

The core intuition is that POPE converts a sparse-reward exploration problem into a two-stage problem with a much easier first stage. In the MDP picture (Figure 8), the bottleneck is not improving behavior *within* the reward-bearing region, but rather *reaching* any state from which reward is attainable. The guidance (or prefix) functions as a roll-in distribution that reliably lands the learner in $\mathcal{S}_{\text{good}}$, so that even early in training the algorithm can observe non-zero reward and fit effective continuations. Once the continuation policy is learned, the role of guidance is largely complete: it is no longer needed to succeed *from* $\mathcal{S}_{\text{good}}$, but it has created a bank of successful trajectories that certify which parts of the state space admit reward and what actions to take there. This "certification" is crucial because membership in $\mathcal{S}_{\text{good}}$ is only revealed by downstream success; without guided roll-ins, standard on-policy RL must simultaneously discover $\mathcal{S}_{\text{good}}$ and learn to exploit it, which is exponentially harder when reward is rare.

Why can this transfer work for LLM reasoning? In autoregressive LLMs, a state can be viewed as the generated prefix, but for long chain-of-thought reasoning the more relevant notion is the model's internal representation induced by that prefix. Due to self-correction and backtracking, many distinct token sequences can correspond to similar latent "problem-solving states". Guidance therefore helps not only by

increasing the likelihood of a successful continuation, but by steering the model into internal states that are both *reachable* and *stable* under subsequent on-policy sampling. When guided rollouts exhibit behaviors such as self-verification, restarting, or revisiting earlier steps, they induce overlap between guided states and a neighborhood of states that the unguided policy can plausibly reach on its own. **Under *function approximation***, this overlap allows learning signal from guided successes to generalize to unguided prefixes, effectively reducing the remaining exploration problem to reaching any nearby state rather than reproducing the full guidance string. This perspective also clarifies why transfer can occur even with a fixed segment of guidance: the structure of reasoning traces induces many revisitations and near-collisions in latent state space, so a single guided roll-in can populate a broad set of useful states from which the learned continuation policy can succeed.

E Training Hyperparameters

This section summarizes the training hyperparameters and system configuration used in this paper. We additionally describe key components of the `Pipeline-RL` framework that orchestrates distributed rollout, preprocessing, and optimization.

Hyperparameter	Values
learning rate	1.0×10^{-5}
num. train epochs	3
global batch size	128
gradient checkpointing	True
max sequence length	16384
precision	bf16
num. GPUs	8
warmup ratio	0.1

Table 3: Hyperparameters used for SFT.

Hyperparameter	Values
max prompt length	2048
max response length	16384
sampling temperature	0.8
clip ratio (low / high)	0 / 5.0
train batch size	32
learning rate	1.0×10^{-6}
total training steps	1000
num. GPUs	8
rollouts per prompt	8
ring buffer size	128

Table 4: Hyperparameters used for RL training with GRPO under the Pipeline-RL framework.

E.1 Hyperparameters for SFT

For supervised fine-tuning (SFT), we use the TRL codebase. All models are initialized from Qwen3-4B-Instruct. Training is performed with full-parameter fine-tuning using bfloat16 precision and gradient checkpointing.

E.2 Hyperparameters for RL

For reinforcement learning, we adopt the `Pipeline-RL` framework with GRPO as the underlying optimization algorithm. At a high level, the training pipeline consists of (i) actor workers that generate rollouts, (ii) preprocessing workers that chunk, filter, and buffer samples, and (iii) learner workers that perform policy optimization. Actors generate up to 8 rollouts per prompt, and samples are stored in a ring buffer with capacity 128 to replace stale data when training lags behind generation. Same as SFT, we use Qwen3-4B-Instruct as the base policy model.

F Qualitative Example

In this section, we present a representative example used in the analysis of Section 5.2. We first show the original problem, and the partial human-written solution provided as guidance during training.

Problem

Given positive integers n, k such that $n \geq 4k$, find the minimal value $\lambda = \lambda(n, k)$ such that for any positive reals $a_1, a_2, \ldots, a_n$, we have

$$\sum_{i=1}^{n} \frac{a_i}{\sqrt{a_i^2 + a_{i+1}^2 + \cdots + a_{i+k}^2}} \leq \lambda$$

Where $a_{n+i} = a_i, i = 1, 2, \ldots, k$

Partial Oracle Solution (Used as Guidance during Training)

Let S be the sum in question:

$$S = \sum_{i=1}^{n} \frac{a_i}{\sqrt{a_i^2 + a_{i+1}^2 + \cdots + a_{i+k}^2}}$$

Let $D_i = a_i^2 + a_{i+1}^2 + \cdots + a_{i+k}^2$. So the sum is $S = \sum_{i=1}^{n} \frac{a_i}{\sqrt{D_i}}$. The value $\lambda(n, k)$ is the maximum possible value of S.
Step 1: Find a lower bound for λ. We will construct a sequence (a_i) for which the sum S approaches a certain value. This value will be a lower bound for λ. Let $a_i = x^{i-1}$ for $i = 1, 2, \ldots, n$ for some small positive real number x. The indices are cyclic, so $a_{n+j} = a_j = x^{j-1}$.

We next show representative unguided solutions generated by models trained with different system instructions. During evaluation, no guidance or partial oracle solution is provided to the model.

Unguided Solution (Model Trained with Default Instruction)

We are given positive integers n, k with the condition $n \geq 4k$, and we are to find the minimal value $\lambda = \lambda(n, k)$ such that for any positive real numbers $a_1, a_2, \ldots, a_n$ (with indices modulo n, so $a_{n+i} = a_i$), the following inequality holds:

$$\sum_{i=1}^{n} \frac{a_i}{\sqrt{a_i^2 + a_{i+1}^2 + \cdots + a_{i+k}^2}} \leq \lambda.$$

Our goal is to find the minimal such λ that works for all such sequences, under the constraint $n \geq 4k$.
...
Each term in the sum is of the form

$$\frac{a_i}{\sqrt{a_i^2 + a_{i+1}^2 + \cdots + a_{i+k}^2}} = \frac{1}{\sqrt{1 + \sum_{j=1}^{k} \left(\frac{a_{i+j}}{a_i}\right)^2}}.$$

...
Suppose we try to make $a_{i+1}, a_{i+2}, \ldots, a_{i+k}$ very small compared to a_i. Then

$$\frac{a_i}{\sqrt{a_i^2 + \sum_{j=1}^{k} a_{i+j}^2}} \approx 1.$$

But due to the cyclic structure, each a_j appears in multiple denominators, so increasing one term necessarily affects others.
...
So there is a trade-off between making individual terms large and maintaining consistency across the cycle. This suggests that extremal or periodic constructions may be relevant in maximizing the sum.
...

Unguided Solution (Model Trained with Modified Instruction)

We are given positive integers n, k with $n \geq 4k$, and we are to find the minimal value $\lambda = \lambda(n,k)$ such that for any positive real numbers $a_1, a_2, \ldots, a_n$ (with indices modulo n), the following inequality holds:

$$\sum_{i=1}^{n} \frac{a_i}{\sqrt{a_i^2 + a_{i+1}^2 + \cdots + a_{i+k}^2}} \leq \lambda.$$

...

Suppose all a_i are equal. Let $a_i = 1$ for all i. Then each denominator is $\sqrt{k+1}$, and each term equals $1/\sqrt{k+1}$. The total sum is therefore

$$\frac{n}{\sqrt{k+1}}.$$

...

Each term is at most 1, since the denominator includes a_i^2. Thus the entire sum is bounded above by n.

...

We can attempt to apply Cauchy-Schwarz or symmetry arguments to bound the sum, but these approaches do not immediately yield a tighter characterization of the extremal behavior.

...

G More Examples

Here we provide an example from the Omni-MATH dataset with a human solution.

Question from Omni-MATH

Let $k \geq 2$ be an integer. Find the smallest integer $n \geq k+1$ with the property that there exists a set of n distinct real numbers such that each of its elements can be written as a sum of k other distinct elements of the set.

Human Solution

Let $k \geq 2$ be an integer. We need to find the smallest integer $n \geq k+1$ such that there exists a set S of n distinct real numbers, where each element of S can be expressed as a sum of k other distinct elements of S.

To solve this problem, we consider the construction of such a set S.

1. **Understanding the Problem:**

- For each element $s \in S$, we need k distinct elements from $S \setminus \{s\}$ that sum up to s.

2. **Minimum Size Construction:**

- We start by proving that with $n = k+4$, such a set can indeed be constructed.
- Consider a construction where:
- Choose $k+1$ elements as the base set: $\{a_1, a_2, \ldots, a_{k+1}\}$.
- Introduce an additional four elements: $\{b_1, b_2, b_3, b_4\}$.
- We construct our set S as:

$$S = \{a_1, a_2, \ldots, a_{k+1}, b_1, b_2, b_3, b_4\}$$

3. **Illustrating the Construction:**

- Arrange the elements such that:
- Each a_i is expressed as the sum of any k of the other a_j's and some b's if necessary.
- Each b_i can be expressed using a combination of a's and other b's.

4. **Verification:**

- By choosing specific numbers for each b_i, we ensure that each number in the constructed set can indeed be expressed as a sum of k distinct others.
- For example, by choosing values and testing that the sum condition holds, we verify that each possibility works, fulfilling the problem's conditions.

5. **Conclusion:**

- Testing smaller n for valid configurations will fail due to insufficient numbers to formulate each possible sum using k distinct numbers.
- Therefore, the smallest n for which such a configuration is possible indeed turns out to be $n = k + 4$.

Thus, the smallest integer n such that a set S with the given conditions can be constructed is: $\boxed{k+4}$

Here we provide an example from the DAPO dataset with a solution generated by gemini-2.5-pro.

Question from DAPO

Find the smallest natural number n such that the expression $n^2 - n + 11$ has exactly four prime factors, which may not be distinct.

Solution generated by Gemini

Let the given expression be $f(n) = n^2 - n + 11$. We are looking for the smallest natural number n such that $f(n)$ has exactly four prime factors, including multiplicity. This means $\Omega(f(n)) = 4$, where $\Omega(k)$ is the total number of prime factors of k.

Let's analyze the expression $f(n) = n(n-1) + 11$. Since $n(n-1)$ is always an even number, $f(n)$ is always odd. So, 2 cannot be a prime factor of $f(n)$. Let's check for other small prime factors: - Modulo 3: $n(n-1)$ can be $0 \times (-1) \equiv 0$, $1 \times 0 \equiv 0$, or $2 \times 1 \equiv 2$. So $n^2 - n + 11 \equiv n(n-1) + 2 \pmod 3$ can be $0 + 2 = 2$, $0 + 2 = 2$, or $2 + 2 = 4 \equiv 1$. So $f(n)$ is never divisible by 3. - Modulo 5: $n(n-1) + 11 \equiv n(n-1) + 1 \pmod 5$. $n(n-1)$ can be $0, 2, 6 \equiv 1, 12 \equiv 2, 20 \equiv 0$. So $n(n-1) + 1 \pmod 5$ can be $1, 2, 3$. So $f(n)$ is never divisible by 5. - Modulo 7: $n(n-1) + 11 \equiv n(n-1) + 4 \pmod 7$. $n(n-1)$ can be $0, 2, 6, 12 \equiv 5, 20 \equiv 6, 30 \equiv 2, 42 \equiv 0$. So $n(n-1) + 4 \pmod 7$ can be $4, 6, 10 \equiv 3, 9 \equiv 2$. So $f(n)$ is never divisible by 7. - Modulo 11: $f(n) = n^2 - n + 11 \equiv n(n-1) \pmod{11}$. $f(n)$ is divisible by 11 if and only if $n(n-1) \equiv 0 \pmod{11}$. Since 11 is prime, this occurs when $n \equiv 0 \pmod{11}$ or $n \equiv 1 \pmod{11}$.

So, the smallest possible prime factor of $f(n)$ is 11. We want $f(n)$ to be a product of four primes, $p_1 p_2 p_3 p_4$. To find the smallest n, we should look for the smallest possible value of $f(n)$. The function $f(n) = n^2 - n + 11$ is increasing for $n \geq 1$. So smaller values of n give smaller values of $f(n)$.

The smallest number that is a product of four primes, where each prime is at least 11, is 11^4. Let's see if $f(n)$ can be equal to 11^4. $11^4 = 14641$. We need to solve $n^2 - n + 11 = 14641$ for a natural number n. $n^2 - n - 14630 = 0$. Using the quadratic formula, $n = \frac{-(-1) \pm \sqrt{(-1)^2 - 4(1)(-14630)}}{2(1)} = \frac{1 \pm \sqrt{1 + 58520}}{2} = \frac{1 \pm \sqrt{58521}}{2}$. For n to be an integer, 58521 must be a perfect square. Let's estimate $\sqrt{58521}$. $200^2 = 40000$, $250^2 = 62500$. $240^2 = 57600$. $241^2 = (240+1)^2 = 57600 + 480 + 1 = 58081$. $242^2 = (240+2)^2 = 57600 + 960 + 4 = 58564$. Since $241^2 < 58521 < 242^2$, 58521 is not a perfect square. Thus, there is no integer n for which $f(n) = 11^4$.

Let's find the next smallest number composed of four prime factors, each at least 11. The candidates for the smallest such numbers are $11^4 = 14641$, $11^3 \times 13 = 1331 \times 13 = 17303$, $11^2 \times 13^2 = 121 \times 169 = 20449$, etc. The smallest is 11^4, which we've ruled out. The next smallest is $11^3 \times 13$. Let's check if $f(n)$ can be equal to $11^3 \times 13 = 17303$. We need to solve $n^2 - n + 11 = 17303$. $n^2 - n - 17292 = 0$. Using the quadratic formula, $n = \frac{1 \pm \sqrt{1 - 4(1)(-17292)}}{2} = \frac{1 \pm \sqrt{1 + 69168}}{2} = \frac{1 \pm \sqrt{69169}}{2}$. For n to be an integer, 69169 must be a perfect square. Let's estimate $\sqrt{69169}$. $200^2 = 40000$, $300^2 = 90000$. $260^2 = 67600$. The last digit is 9, so the root could end in 3 or 7.

Let's try $263^2 = (260+3)^2 = 67600 + 2 \times 3 \times 260 + 9 = 67600 + 1560 + 9 = 69169$. So, 69169 is a perfect square, and its square root is 263. Then $n = \frac{1 \pm 263}{2}$. This gives two possible integer values for n: $n_1 = \frac{1+263}{2} = \frac{264}{2} = 132$. $n_2 = \frac{1-263}{2} = \frac{-262}{2} = -131$. Since we are looking for a natural number n, we have $n = 132$. For $n = 132$, $f(132) = 132^2 - 132 + 11 = 17424 - 132 + 11 = 17292 + 11 = 17303$. And $17303 = 11^3 \times 13$. The prime factors are 11, 11, 11, 13. There are exactly four prime factors. So, $n = 132$ is a candidate.

To be sure that $n = 132$ is the smallest such natural number, we need to confirm that no smaller n results in a four-prime-factor number. The function $f(n)$ is increasing for $n \geq 1$. So for any $n < 132$, $f(n) < f(132) = 17303$. If there were a smaller $n_0 < 132$ such that $f(n_0)$ has four prime factors, then $f(n_0)$ would have to be a number smaller than 17303 with four prime factors, each being at least 11. The smallest such number is $11^4 = 14641$. We have already shown that $f(n)$ cannot be 14641. The next smallest such number is $11^3 \times 13 = 17303$, but this value is not smaller than 17303. Any other combination of four primes (e.g., $11^2 \times 13^2 = 20449$) will be larger. Therefore, there is no value of $f(n)$ with four prime factors smaller than 17303. This means that no $n < 132$ can produce a value of $f(n)$ with four prime factors. Thus, the smallest natural number n is 132.
Final check: For $n = 132$, $f(132) = 132^2 - 132 + 11 = 17303$. Prime factorization of 17303: $17303 = 11 \times 1573$ $1573 = 11 \times 143$ $143 = 11 \times 13$ So $f(132) = 11 \times 11 \times 11 \times 13 = 11^3 \times 13$. This number has $3 + 1 = 4$ prime factors.