PLANNED TEST-TIME SCALING WITH COORDINATED REASONING PATHS

Anonymous authors
Paper under double-blind review

ABSTRACT

Test-time scaling with parallel branches is widely adopted to improve performance on challenging reasoning tasks. The predominant approach, repeated sampling, draws branches independently from a single policy, which can produce redundant attempts and thereby limit the gains from additional inference compute. To address this limitation, we propose **Planned Test-Time Scaling** (PTTS), which replaces independent sampling with a coordinated joint policy: a *planner* generates a solution outline for each branch, steering the branches toward distinct reasoning paths, and an *executor* produces a full solution conditioned on each outline. Formally, we show that PTTS strictly generalizes repeated sampling and, in a stylized setting, provably promotes coverage of complementary reasoning modes and yields better pass@k scaling. We instantiate PTTS on top of strong reasoning models, keeping them fixed as executors while replacing repeated sampling with PTTS inference to further enhance test-time scaling. Concretely, we develop two variants: **PTTS-ZS** prompts a model to jointly generate outlines for all branches in a single autoregressive pass, while **PTTS-RL** directly optimizes the planner against the pass@k reward using truncated execution rollouts for efficient training and a sharper reward signal. Across five mathematical reasoning benchmarks with Qwen3-1.7B and 4B, PTTS-ZS improves pass@64 over repeated sampling by up to 6.7 points, while PTTS-RL further increases the gain to up to 13.4 points. Further analysis indicates that broader coverage of distinct reasoning paths contributes to these gains. Overall, PTTS provides a general framework for improving test-time scaling by coordinating reasoning branches, with zero-shot and trainable instantiations that yield substantial performance gains.

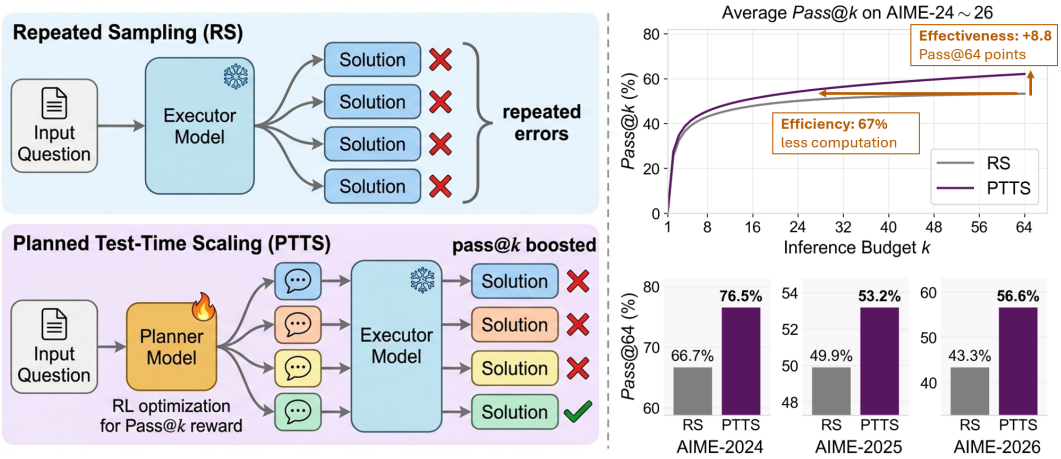


Figure 1: **Left: overview of planned test-time scaling (PTTS).** Unlike repeated sampling (RS) that draws k independent solutions from a frozen executor and often repeats the same errors, PTTS uses a planner to generate k diverse outlines and guide the executor, where the planner can be further trained via reinforcement learning to directly optimize pass@k.

Right: PTTS performance with Qwen3-1.7B on AIME-2024–2026. PTTS matches the pass@64 of RS with 67% less computation (efficiency) and exceeds it by +8.8 points at $k = 64$ (effectiveness).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 INTRODUCTION

As large language models (LLMs) are applied to increasingly challenging reasoning tasks, even strong models often fail to solve a problem reliably in a single attempt. Test-time scaling addresses this limitation by allocating additional inference compute (Wu et al., 2024; Snell et al., 2024; Brown et al., 2024), most commonly through *repeated sampling* (Brown et al., 2024): given a budget of k attempts, the model independently samples k candidate solutions from a single policy, and performance is measured by $\mathrm{pass@}k$, i.e., whether at least one attempt is correct. The benefit of this additional compute therefore depends critically on **coverage**: additional attempts are useful only insofar as they explore complementary reasoning paths rather than repeat existing ones. Under repeated sampling, however, independent samples from the same policy tend to concentrate around high-probability reasoning modes, producing redundant attempts. Reinforcement learning (RL) can further amplify this effect, improving $\mathrm{pass@}1$ while narrowing the policy distribution and degrading $\mathrm{pass@}k$ (Yue et al.; Cui et al., 2025; Wu et al., 2026). Unlocking the full value of parallel compute therefore requires explicit coordination that actively steers branches toward distinct reasoning paths rather than relying on stochasticity to provide coverage.

Recent work explicitly coordinates parallel attempts by first constructing diverse high-level reasoning plans through handcrafted pipelines—for example, iterative concept elicitation from an LLM (Handa et al., 2025) or over-generation followed by clustering of candidate plans (Yang et al., 2025b)—and then using these plans to steer different reasoning branches. Their empirical gains suggest that explicit coordination via a planning stage is a promising direction. We formalize this shared structure by treating planning as a joint policy across branches, enabling both theoretical analysis and direct optimization.

We introduce **Planned Test-Time Scaling** (PTTS), which factorizes the joint policy over k reasoning branches into a planner-executor decomposition. As shown in Figure 1, a *planner* jointly produces k solution outlines, one for each branch, and an *executor* then generates a full solution for each branch conditioned on its outline. By construction, PTTS strictly generalizes repeated sampling and thus guarantees no worse achievable $\mathrm{pass@}k$. In a stylized setting of reasoning-mode selection, we further show that a marginal policy optimized for $\mathrm{pass@}1$ collapses onto a single dominant mode, causing repeated sampling to replay similar failures and bottlenecking test-time scaling on problems that benefit from complementary reasoning modes. In contrast, PTTS can be optimized directly for $\mathrm{pass@}k$, incentivizing the planner to coordinate branches across complementary reasoning modes and yielding strictly better $\mathrm{pass@}k$ scaling.

We instantiate PTTS in two complementary forms, PTTS-ZS and PTTS-RL. The zero-shot variant, **PTTS-ZS**, uses a strong reasoning model as the executor and a *base* LLM as the planner, leveraging the base model's broader generation diversity. Unlike prior approaches that rely on multi-stage pipelines to construct diverse plans, PTTS-ZS generates all k outlines jointly in a single autoregressive pass, yielding a simple yet effective design that naturally enables end-to-end optimization. Building on this formulation, **PTTS-RL** freezes the executor and directly optimizes the planner with the $\mathrm{pass@}k$ of the k executor solutions as the reward. To reduce rollout cost and sharpen credit assignment to the planner, we truncate the executor's reasoning budget during training, preventing the executor from rescuing poor outlines through extended rethinking.

Evaluation on five mathematical reasoning benchmarks using Qwen3 models (Yang et al., 2025a) demonstrates substantial gains in both effectiveness and efficiency. PTTS-ZS consistently improves over repeated sampling, yielding up to a 6.7-point gain in $\mathrm{pass@}64$, while optimizing the planner with PTTS-RL further increases this gain to as much as 13.4 points. Notably, PTTS-RL surpasses repeated sampling using only half the sampling budget, making it more than $2\times$ as compute-efficient. Further analysis points to improved coverage of the reasoning space: PTTS-ZS substantially increases overall reasoning diversity, while PTTS-RL further enhances diversity among branches that reach correct answers. Finally, PTTS-RL planners can serve as flexible add-ons that transfer to executors unseen during training, including larger reasoning models (+4.4 $\mathrm{pass@}64$) and executors trained with diversity-oriented RL (+4.4 $\mathrm{pass@}64$), demonstrating that PTTS is complementary to executor-side improvements.

To summarize, our contributions are as follows: (1) We introduce Planned Test-Time Scaling (PTTS), a planner-executor framework that replaces independent repeated sampling with a coordinated joint policy over multiple reasoning branches. (2) We analyze PTTS theoretically, showing

that it strictly generalizes repeated sampling and that directly optimizing for $\mathrm{pass@}k$ promotes coverage of complementary reasoning modes under fixed inference budgets. (3) We instantiate PTTS in zero-shot and RL-based variants, achieving gains of up to 13.4 points over repeated sampling across five mathematical reasoning benchmarks.

2 Related Work

Mode collapse in test-time scaling. Recent work increasingly uses test-time scaling to enhance off-the-shelf models by allocating additional inference compute (Snell et al., 2024; Wu et al., 2024; Zhang et al., 2025; Brown et al., 2024). A particularly simple and effective approach is repeated sampling, which independently draws multiple solutions from the same policy (Brown et al., 2024). However, repeated sampling relies on independently sampled branches from the same policy, so reduced diversity makes additional samples redundant, limiting coverage and performance. This issue is especially pronounced after reinforcement learning with verifiable rewards (RLVR) (Shao et al., 2024; Guo et al., 2025), which can concentrate the model on a narrow distribution (Yue et al.; Cui et al., 2025; Wu et al., 2026), thereby harming $\mathrm{pass@}k$.

Diversity-aware test-time scaling. One line of work addresses this at inference time by coordinating the k branches with a planning stage prior to problem solving. As our primary baseline, Guided Sampling (Handa et al., 2025) iteratively generates distinct solution concepts to guide the final generation; others sample and cluster high-level plans (Yang et al., 2025b) or perturb the query before parallel attempts (Wang et al., 2025). While effective, these methods lack a unified formal framework that can be optimized end-to-end, which we aim to provide. Closely related to our approach, Kang & Zhang (2025) and Li et al. (2026) train planners to propose multiple strategies before solving, but mostly focus on synthetic or coding tasks and lack a joint formulation or theoretical perspective.

Diversity-aware RL training. Another line of work redesigns RL training to promote exploration behavior and diversity in the marginal policy itself. Prior work directly optimizes estimates of $\mathrm{pass@}k$ for each branch (Tang et al., 2025; Chen et al., 2025; Walder & Karkhanis, 2025), maintains entropy to prevent mode collapse (Cui et al., 2025; Wang et al., 2026), promotes semantic diversity (Li et al., 2025; Yao et al., 2026), adopts risk-aware training objectives (Ren et al.; Jiang et al., 2025), or designs structured curricula to encourage exploration (Setlur et al., 2026). These methods focus on improving the marginal policy and are therefore complementary to our proposed test-time scaling algorithm. As shown in §4.3, our proposed PTTS can be combined with this line of work to further improve performance.

3 Planned Test-Time Scaling

This section presents our planned test-time scaling (PTTS) framework and instantiations. We begin with the problem formulation (§3.1), then provide two theoretical perspectives on why PTTS improves $\mathrm{pass@}k$ (§3.2), and finally describe the concrete algorithm (§3.3).

3.1 From Repeated Sampling to PTTS

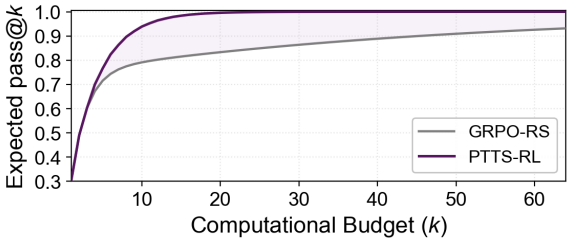
Consider a verifiable reasoning problem $\mathbf{x}$, and let $r(\mathbf{y}; \mathbf{x}) \in \{0, 1\}$ indicate whether a candidate solution $\mathbf{y}$ is correct. Given a test-time budget of k attempts, we measure success using $\mathrm{pass@}k$:

$$\mathrm{pass@}k\left(\mathbf{y}_1, \ldots, \mathbf{y}_k; \mathbf{x}\right) \triangleq \max_{i \in \{1,\ldots,k\}} r(\mathbf{y}_i; \mathbf{x}).$$

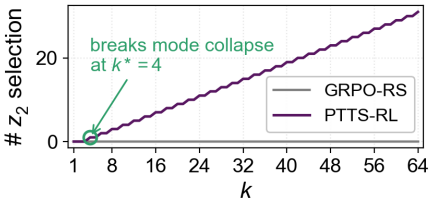
Repeated sampling (RS). As the predominant test-time scaling approach, RS draws the k attempts independently from a single base policy π, inducing the joint distribution:

$$(\mathbf{y}_1, \ldots, \mathbf{y}_k) \sim \pi_{\mathrm{rs}}^{(k)}(\cdot \mid \mathbf{x}), \qquad \pi_{\mathrm{rs}}^{(k)}(\mathbf{y}_1, \ldots, \mathbf{y}_k \mid \mathbf{x}) \triangleq \prod_{i=1}^{k} \pi(\mathbf{y}_i \mid \mathbf{x}).$$

RS is simple and broadly applicable, but it does not coordinate branches or allocate the budget across complementary reasoning paths. As a result, additional samples may cluster around similar high-probability solutions and repeat the same mistakes.



(a) Comparing the pass@k of PTTS-RL against repeated sampling based on GRPO (GRPO-RS).



(b) Inference budgets (out of k) allocated to the non-dominant mode z_2, with larger value representing higher diversity.

Figure 2: Pass@k **performance and policy diversity in a synthetic setup** with two data categories and two reasoning modes (z_1 being dominant and z_2 being non-dominant), as detailed in Appendix A.1.

Planned test-time scaling (PTTS). PTTS instead treats the k attempts as a structured test-time computation rather than independent samples. Ideally, one would model a joint policy directly over $(\mathbf{y}_1, \ldots, \mathbf{y}_k) \sim \pi^{(k)}_{\text{ptts}}(\cdot \mid \mathbf{x})$. However, directly modeling an unrestricted joint distribution over $(\mathbf{y}_1, \ldots, \mathbf{y}_k)$ is difficult: the space of solution tuples grows rapidly with k, and the policy must capture both solution quality and how attempts should differ.

PTTS therefore uses a **planner-executor** factorization. The *planner* first produces a shared plan consisting of k solution outlines, $\mathbf{o} = (\mathbf{o}_1, \ldots, \mathbf{o}_k) \sim \pi_p(\cdot \mid \mathbf{x})$, where each outline steers one attempt toward a distinct reasoning path. The *executor* then generates a full solution conditioned on the problem and the corresponding outline: $\mathbf{y}_i \sim \pi_e(\cdot \mid \mathbf{x}, \mathbf{o}_i)$. Equivalently, PTTS defines the following joint distribution over the generated plan and solutions:

$$\pi^{(k)}_{\text{ptts}}(\mathbf{o}, \mathbf{y}_1, \ldots, \mathbf{y}_k \mid \mathbf{x}) \triangleq \pi_p(\mathbf{o} \mid \mathbf{x}) \prod_{i=1}^{k} \pi_e(\mathbf{y}_i \mid \mathbf{x}, \mathbf{o}_i).$$

This factorization makes PTTS tractable while preserving branch-level coordination: the planner chooses complementary high-level reasoning paths, and the executor realizes each path independently. In §3.2, we analyze why this structure can improve pass@k over RS.

3.2 Why PTTS Improves Pass@k

Expressiveness. From a policy-class perspective, $\pi^{(k)}_{\text{ptts}}$ strictly generalizes $\pi^{(k)}_{\text{rs}}$: when the planner emits an empty or constant outline for every branch, PTTS reduces to RS. Therefore, the optimal pass@k attainable within PTTS is no lower than that attainable within RS.

This gap can be made *strict* under a per-attempt success constraint. Suppose each branch has fixed marginal success probability $p \triangleq \mathbb{E}\left[r(\mathbf{y}_i; \mathbf{x})\right]$. For RS, independence gives $\text{pass@}k = 1 - (1 - p)^k$. In contrast, PTTS can coordinate branches so that their success events are nearly disjoint, for example by assigning complementary strategies. In the ideal disjoint case, $\text{pass@}k = \min(kp, 1)$, which is *strictly larger* than RS for $p \in (0, 1)$ and $k > 1$. Thus, PTTS converts the same average per-attempt competence into a higher joint pass rate by reducing redundancy across attempts.

Mode coverage analysis. We then take a dataset-level optimization perspective to illustrate why standard paradigms such as GRPO suffer from *mode collapse*, and how PTTS-RL naturally mitigates this issue. Optimizing dataset-level pass@1 encourages the policy to concentrate on a *dominant reasoning mode*. While this maximizes average single-shot reward, it inherently sacrifices diversity and can ultimately degrade pass@k (Yue et al.).

To formalize this intuition, we study a simplified setting in which the executor is fixed and imperfect, reducing the reasoning process to a mode-selection problem. Assume each example belongs to a latent category $c \in \mathcal{C}$, and the policy selects a reasoning mode $z \in \mathcal{Z}$, with success rate R_{zc} determined only by the selected mode z and data category c. We consider three realistic assumptions, with the full derivation deferred to Appendix A:

1. *Imperfect policy:* The policy cannot observe the true category c perfectly. Instead, it acts based on a noisy observed category c', making perfect deterministic mode-matching impossible.
2. *Mode dominance:* For a certain observed category c', there exists a single dominant mode $z^*(c')$ that achieves a strictly higher expected success rate than any other mode.

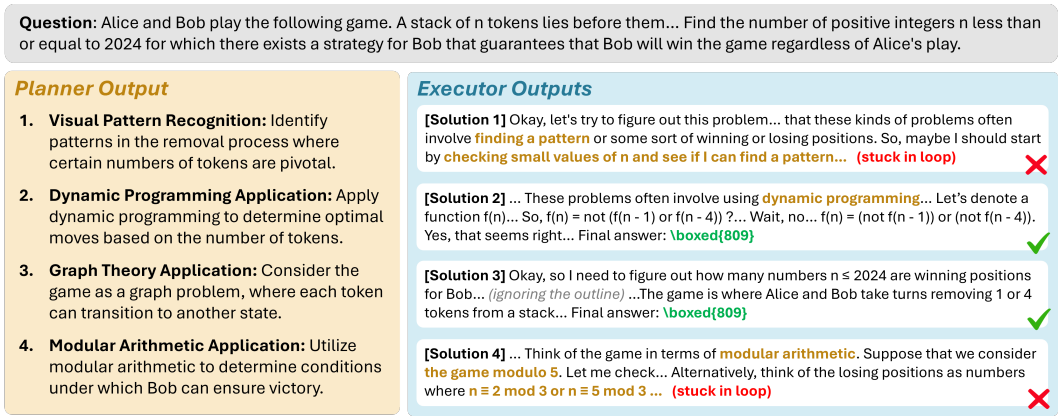


Figure 3: **Example output from PTTS-ZS. Left:** a markdown list of four distinct outlines generated by the planner π_p in a single autoregressive pass. **Right:** solutions generated by the executor π_e conditioned on each outline, with the outline-following portions **highlighted**. Most solutions faithfully follow their outlines, steering the reasoning toward different modes and boosting the diversity.

3. *Complementary modes:* The dominant mode z^* struggles on a "worst-case" category $c^\dagger$ by a strict margin. Crucially, this worst-case category benefits more from a non-dominant mode.

Under these conditions, optimizing the marginal policy for pass@1 via GRPO drives fundamentally different behavior than optimizing the joint policy against $\text{pass@}k$ via PTTS-RL. Since GRPO maximizes expected average return, it assigns all probability mass to the dominant mode $z^*(c')$, leading directly to mode collapse. In contrast, PTTS-RL optimizes directly for $\text{pass@}k$, naturally accounting for the *diminishing marginal returns* of repeatedly sampling z^*. We show that beyond a finite threshold k^*, **PTTS-RL strategically allocates reasoning slots to non-dominant modes**, thereby rescuing performance on the worst-case category $c^\dagger$. Consequently, these policies exhibit distinct scaling dynamics: PTTS-RL drives the residual error $\epsilon = 1 - \text{pass@}k$ to zero exponentially as $k \to \infty$, whereas GRPO's $\text{pass@}k$ plateaus whenever the dominant mode has zero accuracy on the worst-case category. Even when GRPO does not plateau, **PTTS-RL shrinks the residual error at a strictly faster exponential rate** by actively covering blind spots rather than repeatedly resampling the dominant mode.

To illustrate these theoretical mechanics, Figure 2 instantiates this model in a minimal two-category, two-mode environment where z_1 is dominant. The numerical results confirm our intuition: PTTS-RL actively breaks mode collapse, substantially improving coverage of the non-dominant class (Figure 2b) and accelerating $\text{pass@}k$ improvements after the k^* threshold is crossed and before test-time scaling saturates (Figure 2a).

3.3 Instantiation

PTTS-ZS. In the zero-shot setting, we instantiate both the planner π_p and the executor π_e with LLMs. As shown in Figure 3, given a problem $\mathbf{x}$, the planner generates a markdown list of k solution outlines in a single autoregressive pass, which is then deterministically parsed into k outlines $(\mathbf{o}_1, \ldots, \mathbf{o}_k)$. The executor then independently generates a full solution $\mathbf{y}_i \sim \pi_e(\cdot \mid \mathbf{x}, \mathbf{o}_i)$ for each outline. Concretely, we use strong reasoning models such as Qwen3 (Yang et al., 2025a) as executors, making PTTS-ZS an inference-only add-on to further enhance the test-time scaling. However, we find that reasoning models are poorly suited as planners: they tend to commit early to a single strategy and start solving the problem, rather than generating multiple diverse, high-level solution outlines. Therefore, we use the corresponding base models as planners. Despite its simplicity, PTTS-ZS yields surprisingly large improvements over $\text{pass@}k$.

PTTS-RL. We then present PTTS-RL, which trains the planner end-to-end with GRPO (Shao et al., 2024; Yu et al., 2026) to directly optimize $\text{pass@}k$.

In GRPO, for each input $\mathbf{x}$, we sample G outputs $\{\mathbf{y}^{(i)}\}_{i=1}^{G}$ from the model and obtain a reward R_i for each output. We then compute the advantage A_i by normalizing the rewards within the group: $A_i = \Big(R_i - \text{mean}(\{R_j\}_{j=1}^{G})\Big) \,/\, \text{std}(\{R_j\}_{j=1}^{G})$. This same advantage is assigned to every token in

the corresponding output $\mathbf{y}^{(i)}$, i.e., $A_{i,t} = A_i$. The resulting training objective is:

$$\mathcal{J}_{\text{GRPO}} = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|\mathbf{y}^{(i)}|}\sum_{t=1}^{|\mathbf{y}^{(i)}|}\min\left(\rho_{i,t}A_{i,t},\ \text{clip}(\rho_{i,t}, 1-\epsilon_L, 1+\epsilon_H)A_{i,t}\right)\right],$$

where $\rho_{i,t} = \pi_\theta\left(\mathbf{y}_t^{(i)} \mid \mathbf{x}, \mathbf{y}_{<t}^{(i)}\right) \ / \ \pi_{\theta_{old}}\left(\mathbf{y}_t^{(i)} \mid \mathbf{x}, \mathbf{y}_{<t}^{(i)}\right)$ is the importance sampling term.

PTTS-RL applies GRPO to the planner model π_p, where each sampled output is a list of outlines $\mathbf{o} = (\mathbf{o}_1, \ldots, \mathbf{o}_k)$. The reward for a sampled output is computed using a fixed executor π_e, which generates a solution $\mathbf{y}_i$ for each of the k outlines, $i = 1, \ldots, k$. Each generated solution is evaluated for accuracy as $r(\mathbf{y}_i \mid \mathbf{x}) \in \{0, 1\}$, yielding the pass@$k$ reward:

$$R(\mathbf{o}) = \max_{i=1,\ldots,k} r(\mathbf{y}_i \mid \mathbf{x}), \quad \mathbf{y}_i \sim \pi_e(\cdot \mid \mathbf{x}, \mathbf{o}_i).$$

However, generating full-length solutions for all k branches makes training computationally expensive. Moreover, long execution traces allow the executor to drift away from the outline and achieve the correct answer through extensive rethinking, assigning positive reward to a poor outline and weakening the training signal. We therefore adopt a *truncated execution* strategy during training, limiting executor outputs to fewer tokens than at inference ($4k$ vs. $10k$ tokens, as detailed in §4.1) to improve efficiency and sharpen the reward signal.

4 Experiments

We empirically study three questions: (1) How much does the PTTS framework improve pass@k? (2) Which output properties drive these gains, and how do PTTS-ZS and PTTS-RL shape them? (3) Which PTTS design choices and hyperparameters most affect PTTS pass@k performance? In this section, we describe the experimental setup in §4.1, address (1)–(2) through the results and analyses in §4.2, and study (3) through ablations in §4.3.

4.1 Setup

Evaluation settings. We evaluate on five benchmarks: MATH-500 (Hendrycks et al.), AIME 2024, 2025, and 2026 (Art of Problem Solving, n.d.), and HMMT-Feb26 (Dekoninck et al., 2026). We report pass@k for k up to 64. For each problem, we generate $n = 64$ solutions, with a budget of $10k$ tokens per solution. We then analytically compute pass@k as the expected maximum reward among k responses sampled from the set of n generated responses:

$$\text{pass@}k = \mathbb{E}\left[1 - \binom{n-c}{k} \Big/ \binom{n}{k}\right], \quad c = \sum_{i=1}^{n}\mathbf{1}\left(r(\mathbf{y}_i \mid \mathbf{x}) = 1\right).$$

Models and baselines. We compare PTTS-ZS and PTTS-RL against two baselines: (1) repeated sampling, and (2) Guided Sampling (Handa et al., 2025), an inference-time method similar to PTTS-ZS that first generates diverse *concepts* and then uses them to guide reasoning. We conduct experiments with Qwen3 models at 1.7B and 4B scales. For all methods, we use the reasoning model `Qwen3-1.7B/4B` as the executor. For zero-shot methods that involve planning (PTTS-ZS and Guided Sampling), we use the corresponding base model, `Qwen3-1.7B/4B-Base`, as the planner. For Guided Sampling, we follow the recommended setting of generating 5 ideas. For PTTS-RL, we train the planner starting from the base model, as discussed below.

Training settings. We train the planner from the base models using the DAPO training set (Yu et al., 2026), optimizing for the pass@k ($k = 4$) reward. As in §3.3, we use a truncated execution budget of $4k$ tokens for each solution. We train the `Qwen3-1.7B/4B-Base` models with a learning rate of 1e-6 for 240 steps and use the final checkpoint; detailed hyperparameter settings are provided in Appendix B.

Scaling PTTS to larger test-time budgets. In both inference (PTTS-ZS and PTTS-RL) and training (PTTS-RL), we use a branching factor of $k = 4$. To evaluate larger total budgets $K > k$, we independently repeat the PTTS procedure until we collect K executor solutions in total, and then compute pass@K over the aggregated set. For PTTS-RL, this allows us to keep the same planner

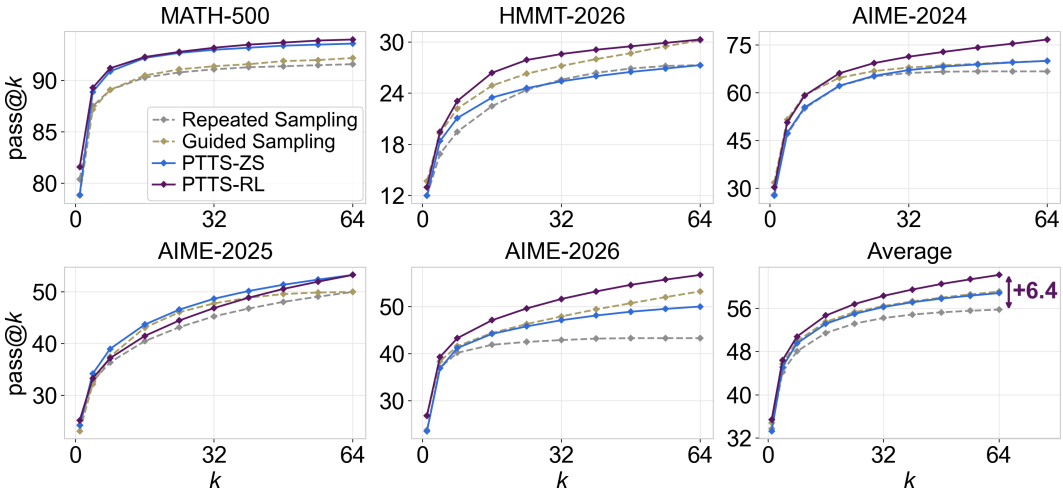


Figure 4: **Main results for Qwen3-1.7B.** PTTS-RL improves pass@64 over repeated sampling by up to 13.4 (6.4 on average), while its pass@32 already exceeds repeated sampling's pass@64 (58.3 vs. 55.8).

while seamlessly scaling to larger inference budgets. As discussed in §4.3, further increasing k yields diminishing returns; therefore, we use $k = 4$ as a computationally efficient setting to demonstrate our method.

4.2 Main Results and Discussions

Main results. Figures 4 and 5 present our main results for the 1.7B and 4B models, respectively. Despite its simple design, PTTS-ZS consistently outperforms repeated sampling, with pass@64 gains of up to 6.7 for 1.7B models and 3.4 for 4B models. PTTS-RL further improves upon PTTS-ZS and consistently outperforms both repeated sampling and Guided Sampling. Specifically, compared to repeated sampling, PTTS-RL improves pass@64 by up to 13.4 for 1.7B models and 6.7 for 4B models. Beyond performance gains, improved compute efficiency is a broader benefit of PTTS. Most notably, at both model scales, PTTS-RL's pass@32 exceeds repeated sampling's pass@64, achieving better performance with only half the sampling budget. Overall, these results demonstrate the benefit of coordinating test-time compute across complementary reasoning paths.

Diversity as an indicator of pass@k. To better understand what drives these gains, we examine whether broader coverage of reasoning strategies is associated with better pass@k performance. Concretely, we prompt GPT-5-mini to group outlines and solutions into clusters and report the number of unique clusters as the diversity metric, as detailed in Appendix C.2. We then use Qwen3-1.7B's PTTS-ZS outputs on AIME-2024–2026 to analyze the rank-averaged correlations among outline diversity, solution diversity, and pass@4 across groups of $k = 4$ reasoning branches. Results show that greater outline diversity is associated with higher solution diversity (Spearman's $\rho = 0.57$) and better resulting pass@4 ($\rho = 0.59$), while solution diversity is also positively correlated with pass@4 ($\rho = 0.32$). Overall, these results support diversity as a meaningful indicator of pass@k performance across both the outline and solution levels.

Diversity induced by PTTS. We next investigate whether PTTS effectively enhances diversity. Using the same diversity measure discussed above, we compare the diversity of repeated sampling, PTTS-ZS, and PTTS-RL on Qwen3-1.7B outputs on the AIME-2024–2026 datasets. As shown in Figure 6, both PTTS variants increase overall response diversity over repeated sampling, with PTTS-ZS achieving the highest diversity across generated solutions. Optimized for the pass@k objective, PTTS-RL does not further increase overall diversity over PTTS-ZS, but instead steers diversity toward branches that reach correct answers. Measuring this *useful diversity* as the normalized count of distinct clusters among correct branches (detailed in Appendix C.3), PTTS-RL achieves the highest *useful diversity* of 15.9%, outperforming both repeated sampling and PTTS-ZS. Figure 10 further shows a qualitative example of the increased diversity under PTTS. Overall, PTTS not only increases reasoning diversity but also steers it toward successful reasoning paths, thereby improving pass@k.

Outline adherence of the executor. Beyond the planner, PTTS also relies on the executor to faithfully develop each proposed outline into a full solution. Reasoning models, however, are not ex-

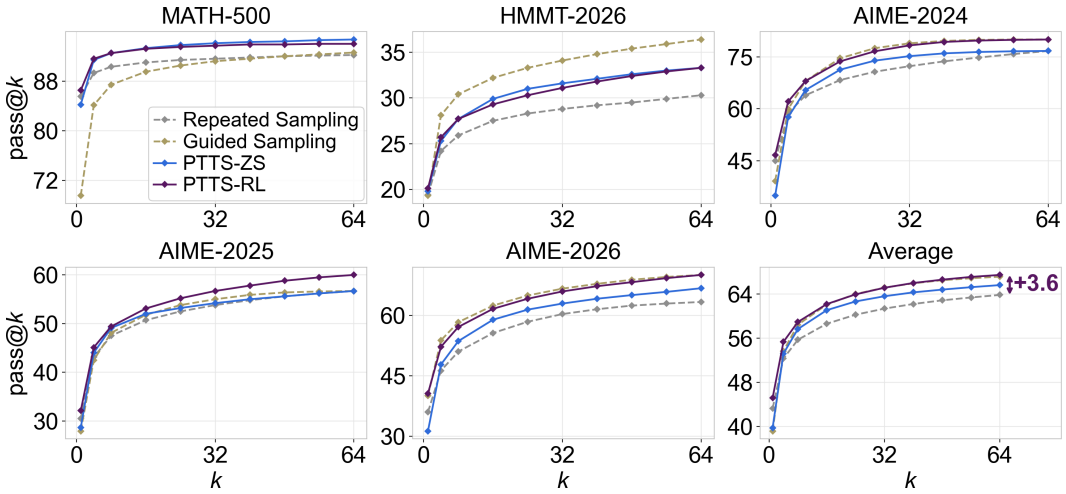


Figure 5: **Main results for Qwen3-4B.** PTTS-RL improves pass@64 over repeated sampling by up to 6.7 (3.6 on average), while its pass@32 already exceeds repeated sampling's pass@64 (65.1 vs. 63.8).

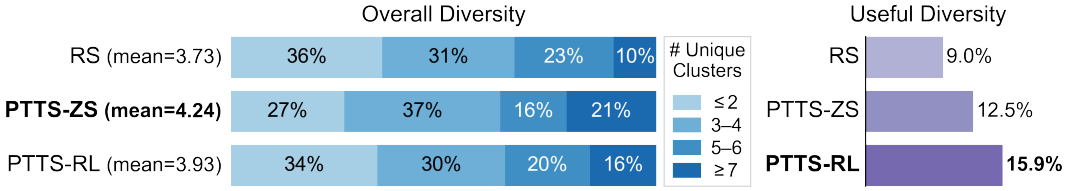


Figure 6: **Solution diversity for repeated sampling (RS), PTTS-ZS, and PTTS-RL**, measured on Qwen3-1.7B outputs for AIME 2024–2026. **Left:** Overall diversity, measured by the number of distinct solution clusters among $k = 64$ branches (Appendix C.2). Bars show the fraction of problems in each diversity range, with parentheses indicating the mean number of distinct clusters. **Right:** Useful diversity, measured as the fraction of correct solutions belonging to distinct solution clusters (Appendix C.3).

plicitly trained for outline-guided generation, making it unclear how closely their solutions adhere to the provided outlines. We evaluate outline adherence via LLM-as-a-judge using GPT-5-mini, as detailed in Appendix C.1. Analysis of PTTS-ZS and PTTS-RL outputs on the AIME-2024–2026 datasets shows that Qwen3-1.7B and Qwen3-4B follow outlines reasonably well, achieving adherence rates of 70% and 71%, respectively. Interestingly, qualitative observations show that while executors typically begin by either strictly following or reiterating the outline verbatim, they may pivot to alternative strategies if the initial approach fails, with an example shown in Figure 11. This may dilute the outline's influence on final performance as reasoning traces become longer, thereby affecting the training signal, as discussed in §4.3.

4.3 Ablation Studies

Ablations on the branching factor k. We study the effect of the PTTS branching factor k while fixing the total inference budget at $K = 64$, following the scaling procedure in §4.1. As shown in Figure 7, moving from repeated sampling ($k = 1$) to modest branching factors substantially improves PTTS-ZS, with performance peaking at $k = 4$ for Qwen3-1.7B and $k = 8$ for Qwen3-4B. Beyond these points, increasing k yields no consistent pass@64 gains, with performance fluctuating at larger values. Since larger k provides no reliable benefit while substantially increasing PTTS-RL training cost, we use $k = 4$ throughout our main experiments.

Ablations on truncated execution. To evaluate the impact of training-time execution limits, we train Qwen3-1.7B with PTTS-RL using execution budgets from $2k$ to $10k$ tokens, while fixing the inference budget at $10k$. As shown in Figure 8, performance peaks at $4k$ and degrades with larger budgets: matching the $10k$ inference budget lowers average pass@64 by 7.8 points and throughput by 27%. Longer executions allow the executor to recover from poor outlines through extended rethinking, causing these outlines to receive positive reward and weakening the training signal. Conversely, an overly strict $2k$ budget prematurely truncates valid reasoning, making the reward

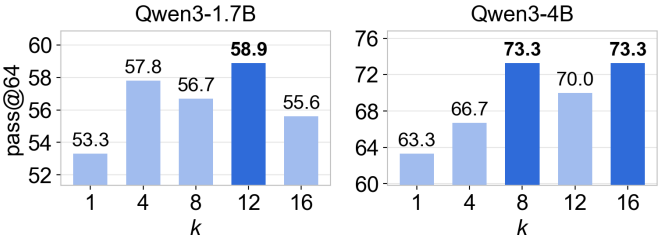


Figure 7: Pass@64 performance of PTTS-ZS across branching factors k at a fixed inference budget of $K = 64$, averaged across AIME 2024–2026; $k = 1$ denotes repeated sampling.

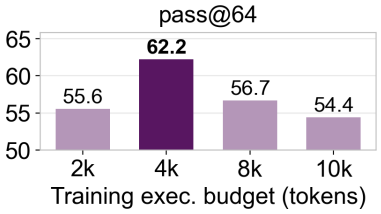


Figure 8: Pass@64 performance of PTTS-RL using Qwen3-1.7B trained with varying execution budgets, averaged across AIME 2024–2026.

uninformative and substantially hurting performance. Overall, a $4k$ budget yields the optimal balance, providing enough length to reliably execute outlines and actively truncating unguided rethinking.

Ablations on planner and executor sizes. While our main experiments use planners and executors of the same sizes, decoupling them reveals distinct scaling behaviors. As shown in Table 1, 1.7B planners effectively guide a larger Qwen3-4B executor, yielding significant gains over repeated sampling and matching the performance of 4B planners, despite a train-test mismatch: the 1.7B PTTS-RL planner is trained with the smaller Qwen3-1.7B executor. In contrast, using 4B planners to guide a Qwen3-1.7B executor provides no additional benefit over 1.7B planners. These results suggest a promising strategy of training small but capable planners alongside small executors, then scaling only the executor at inference time.

Algorithm	Planner	Pass@k		
		k=1	k=4	k=64
Qwen3-1.7B as executor				
RS	-	25.4	38.9	53.3
PTTS-ZS	1.7B	25.2	39.5	57.8
PTTS-ZS	4B	25.6	40.4	60.0
PTTS-RL	1.7B	**27.5**	41.1	**62.2**
PTTS-RL	4B	**27.5**	**41.3**	61.1
Qwen3-4B as executor				
RS	-	37.2	49.4	65.6
PTTS-ZS	1.7B	36.8	52.0	68.9
PTTS-ZS	4B	31.6	49.8	66.7
PTTS-RL	1.7B	**42.1**	**54.7**	**70.0**
PTTS-RL	4B	39.8	53.1	**70.0**

Table 1: Pass@k performance of repeated sampling (RS), PTTS-ZS, and PTTS-RL under different planner and executor configurations. Highlighted rows indicate configurations where the planner and executor differ in size.

Combination with diversity-aware RL. As discussed in §2, PTTS operates at inference time and is complementary to training-side diversity-aware improvements. We demonstrate this by using e3 (Setlur et al., 2026) as the executor, which is trained to improve exploration through a structured curriculum. With `e3-1.7B` as the executor, both the zero-shot 1.7B planner and the RL planner trained with a Qwen3-1.7B executor outperform repeated sampling by 4.4 and 3.3 points, respectively. These gains show that PTTS remains effective on top of a diversity-aware RL executor, providing an additional and complementary source of improvement.

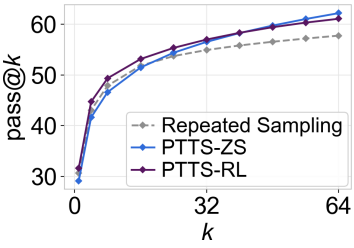


Figure 9: Pass@k of repeated sampling, PTTS-ZS, and PTTS-RL with `e3-1.7B` as the executor.

5 Conclusion

We introduce Planned Test-Time Scaling (PTTS), a framework that addresses the redundant reasoning and repeated errors of standard repeated sampling by explicitly coordinating attempts across k different branches. PTTS decomposes test-time scaling under a budget of k reasoning branches into a planner stage that produces diverse solution outlines and an executor stage that follows each outline. We instantiate PTTS with a zero-shot variant, PTTS-ZS, which plans using an off-the-shelf model, and a trained variant, PTTS-RL, which directly optimizes the planner for the pass@k reward. We show theoretically and empirically that PTTS outperforms independent repeated sampling: across five mathematical datasets, both PTTS-ZS and PTTS-RL consistently yield gains of up to 13.4 points. Overall, PTTS provides a principled paradigm for test-time scaling, highlighting explicit coordination and budget allocation as key to better unlocking its potential.

REFERENCES

Art of Problem Solving. AIME Problems and Solutions. https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions, n.d. AoPS Wiki. Accessed: 2026-05-14.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

Zhipeng Chen, Xiaobo Qin, Youbin Wu, Yue Ling, Qinghao Ye, Wayne Xin Zhao, and Guang Shi. Pass@ k training for adaptively balancing exploration and exploitation of large reasoning models. *arXiv preprint arXiv:2508.10751*, 2025.

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, et al. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint arXiv:2505.22617*, 2025.

Jasper Dekoninck, Nikola Jovanović, Tim Gehrunger, Kári Rögnvaldsson, Ivo Petrov, Chenhao Sun, and Martin Vechev. Beyond benchmarks: Matharena as an evaluation platform for mathematics with llms. 2026. URL https://arxiv.org/abs/2605.00674.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Divij Handa, Mihir Parmar, Aswin RRV, Md Nayem Uddin, Hamid Palangi, and Chitta Baral. Guidedsampling: Steering llms towards diverse candidate solutions at inference-time. *arXiv preprint arXiv:2510.03777*, 2025.

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

Yuhua Jiang, Jiawei Huang, Yufeng Yuan, Xin Mao, Yu Yue, Qianchuan Zhao, and Lin Yan. Risk-sensitive rl for alleviating exploration dilemmas in large language models. *arXiv preprint arXiv:2509.24261*, 2025.

Shijia Kang and Muhan Zhang. The road less traveled: Enhancing exploration in llms via sequential sampling. *arXiv preprint arXiv:2510.15502*, 2025.

Tianjian Li, Yiming Zhang, Ping Yu, Swarnadeep Saha, Daniel Khashabi, Jason Weston, Jack Lanchantin, and Tianlu Wang. Jointly reinforcing diversity and quality in language model generations. *arXiv preprint arXiv:2509.02534*, 2025.

Yilong Li, Suman Banerjee, and Tong Che. Cast a wider net: Coordinated pass@ k policy optimization for code reasoning. *arXiv preprint arXiv:2605.27000*, 2026.

Tao Ren, Jinyang Jiang, Hui Yang, Wan Tian, and Yijie Peng. Riskpo: Risk-based policy optimization with verifiable reward for llm post-training. In *NeurIPS 2025 Workshop MLxOR: Mathematical Foundations and Operational Integration of Machine Learning for Uncertainty-Aware Decision-Making*.

Amrith Setlur, Matthew Yang, Charlie Snell, Jeremiah Greer, Ian Wu, Virginia Smith, Max Simchowitz, and Aviral Kumar. e3: Learning to explore enables extrapolation of test-time compute for llms. In *International Conference on Learning Representations*, volume 2026, pp. 127323–127361, 2026.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

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. *arXiv preprint arXiv: 2409.19256*, 2024.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Yunhao Tang, Kunhao Zheng, Gabriel Synnaeve, and Rémi Munos. Optimizing language models for inference time objectives using reinforcement learning. *arXiv preprint arXiv:2503.19595*, 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, Xiong-Hui Chen, Jianxin Yang, Zhenru Zhang, et al. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *Advances in Neural Information Processing Systems*, 38:115452–115486, 2026.

Tianchun Wang, Zichuan Liu, Yuanzhou Chen, Jonathan Light, Weiyang Liu, Haifeng Chen, Xiang Zhang, and Wei Cheng. On the effect of sampling diversity in scaling llm inference, 2025. URL https://arxiv.org/abs/2502.11027.

Fang Wu, Weihao Xuan, Ximing Lu, Mingjie Liu, Yi Dong, Zaid Harchaoui, and Yejin Choi. The invisible leash: Why rlvr may or may not escape its origin, 2026. URL https://arxiv.org/abs/2507.14843.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models. *arXiv preprint arXiv:2408.00724*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025a. URL https://arxiv.org/abs/2505.09388.

Kaisen Yang, Lixuan He, Rushi Shah, Kaicheng Yang, Qinwei Ma, Dianbo Liu, and Alex Lamb. Explore-execute chain: Towards an efficient structured reasoning paradigm. *arXiv preprint arXiv:2509.23946*, 2025b.

Jian Yao, Ran Cheng, Xingyu Wu, Jibin Wu, and Kay Chen Tan. Diversity-aware policy optimization for large language model reasoning. *Advances in Neural Information Processing Systems*, 38: 94801–94826, 2026.

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. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2026.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Wenyue Hua, Haolun Wu, Zhihan Guo, Yufei Wang, Niklas Muennighoff, et al. A survey on test-time scaling in large language models: What, how, where, and how well? *arXiv preprint arXiv:2503.24235*, 2025.

A PROOFS OF PROPOSITIONS

As discussed in §3.2, we consider a finite space for data category $\mathcal{C}$ and a finite space for reasoning mode $\mathcal{Z}$. For a problem $\mathbf{x}$ with data category $c(\mathbf{x}) = c \in \mathcal{C}$, a policy $\pi(\cdot \mid \mathbf{x})$ chooses a reasoning mode $z \sim \pi(\cdot \mid \mathbf{x}),\ z \in \mathcal{Z}$ for solving this task, and the resulting success rate R_{zc} is determined only by the selected mode z and data category c; when multiple attempts are issued on the same problem, their outcomes are independent conditional on the data category, each succeeding with probability R_{zc} under its selected mode.

Critically, we assume an **imperfect policy** that cannot perfectly observe the true data category $c(\mathbf{x})$ and perform optimal mode selection accordingly. Instead, we assume the policy that chooses mode based on observed data category $c'(\mathbf{x})$ with imperfect data categorization,

$$\pi(\cdot \mid \mathbf{x}) = \pi(\cdot \mid c'(\mathbf{x})),$$

with some emission probability conditioned on the true data category $c(\mathbf{x})$, i.e.

$$\Pr\left[c'(\mathbf{x}) = c' \mid c(\mathbf{x}) = c\right].$$

For notation convenience, we assume this probability is strictly positive for any (c, c') pair; similarly, we assume the marginal probability $\Pr[c(\mathbf{x}) = c]$ is strictly positive for all c. Then we have a posterior distribution of

$$\omega(c \mid c') \triangleq \Pr\left[c(\mathbf{x}) = c \mid c'(\mathbf{x}) = c'\right]$$

that is also non-zero for any (c, c'). Then, for a certain observed data category c', we have the *posterior-averaged success rate* of each mode $z \in \mathcal{Z}$ under the observation c',

$$\bar{R}_{zc'} \ \triangleq\ \mathbb{E}_{c\sim\omega(\cdot|c')}\left[R_{zc}\right] \ =\ \sum_{c\in\mathcal{C}} \omega(c \mid c')\, R_{zc},$$

i.e. the success rate that mode z effectively attains through the imperfect observation: since the true category is unobserved, a single attempt with mode z succeeds with probability exactly $\bar{R}_{zc'}$ given $c'(\mathbf{x}) = c'$ by the tower rule. Based on this formulation, we enforce two critical assumptions that will drive the rest of the proof:

Assumption 1 (Mode dominance) *The observed category $c' \in \mathcal{C}$ is dominated by a single mode $z^* \in \mathcal{Z}$:*

$$\forall z \in \mathcal{Z}, z \neq z^*,\ \bar{R}_{zc'} < \bar{R}_{z^*c'}.$$

Assumption 2 (Complementary modes) *There exists a single worst category $c^\dagger \in \mathcal{C}$,*

$$\exists\, c^\dagger \in \mathcal{C},\ \forall c \in \mathcal{C}, c \neq c^\dagger,\ R_{z^*c} > R_{z^*c^\dagger},$$

and the category benefits more from a mode other than z^*:

$$\exists z \in \mathcal{Z},\ R_{zc^\dagger} > R_{z^*c^\dagger}.$$

We now consider two types of policies, a **GRPO-RS** policy and a **PTTS-RL** policy, both operating on the observed category $c'(\mathbf{x}) = c'$ and issuing k attempts under inference-time budget k. Each attempt will adopt a reasoning mode z, resulting in a solution $\mathbf{y}$ with a binary outcome $r(\mathbf{y}; \mathbf{x}) \in \{0, 1\}$, where $\Pr[r(\mathbf{y}; \mathbf{x}) = 1] = R_{zc}$.

GRPO-RS policy, as the standard approach, optimizes a marginal policy against $\mathrm{pass@1}$, and utilizes repeated sampling (RS) to scale to a larger inference-time budget k. With slight abuse of notation, we write $\mathrm{pass@1}(\pi)$ for the expectation of $\mathrm{pass@1}$ under policy π. Formally, the marginal policy is trained to maximize $\mathrm{pass@1}$, i.e. the single-attempt success rate,

$$\pi_{\mathrm{rs}}(\cdot \mid c') \ \in\ \arg\max_{\pi}\ \mathrm{pass@1}(\pi),$$

where

$$\begin{aligned}\mathrm{pass@1}(\pi) &\triangleq \mathbb{E}_{c\sim\omega(\cdot|c'),\, z\sim\pi(\cdot|c')}\left[\mathbf{1}\left(r(\mathbf{y};\mathbf{x}) = 1\right)\right] = \mathbb{E}_{c\sim\omega(\cdot|c')}\left[\sum_{z\in\mathcal{Z}} \pi(z \mid c')\, R_{zc}\right] \\ &= \sum_{z\in\mathcal{Z}} \pi(z \mid c')\, \bar{R}_{zc'}.\end{aligned}$$

At inference budget k, RS draws modes i.i.d. from this *fixed* marginal, $z_1, \ldots, z_k \overset{\text{i.i.d.}}{\sim} \pi_{\text{rs}}(\cdot \mid c')$, and runs one attempt per draw. The resulting $\text{pass@}k$ has expectation

$$\text{pass@}k\left(\pi_{\text{rs}}^{(k)}\right) = 1 - \mathbb{E}_{c\sim\omega(\cdot|c')}\left[\Big(\sum_{z\in\mathcal{Z}} \pi_{\text{rs}}(z \mid c')\,(1 - R_{zc})\Big)^k\right],$$

with residual error $\epsilon_k^{\text{rs}} \triangleq 1 - \text{pass@}k\left(\pi_{\text{rs}}^{(k)}\right)$.

PTTS-RL policy, on the other hand, optimizes the use of the entire inference budget directly against $\text{pass@}k$. At inference budget k, the policy selects an allocation $\mathbf{k} = (k_z)_{z\in\mathcal{Z}}$ that assigns the k attempt slots across reasoning modes, with $k_z \in \mathbb{Z}_{\geq 0}$ and $\sum_{z\in\mathcal{Z}} k_z = k$. An allocation $\mathbf{k}$ then achieves a $\text{pass@}k$ of:

$$\text{pass@}k(\mathbf{k}) = 1 - \mathbb{E}_{c\sim\omega(\cdot|c')}\left[\prod_{z\in\mathcal{Z}} (1 - R_{zc})^{k_z}\right],$$

and the PTTS-RL policy at budget k selects the optimal allocation

$$\mathbf{k}^{(k)} \in \underset{\mathbf{k}:\,\sum_z k_z = k}{\arg\max}\ \text{pass@}k(\mathbf{k}),$$

with residual error $\epsilon_k^{\text{ptts}} \triangleq 1 - \text{pass@}k\big(\mathbf{k}^{(k)}\big)$ and induced mode-selection distribution $\pi_{\text{ptts}}^{(k)}(z \mid c') \triangleq k_z^{(k)}/k$. In contrast to GRPO-RS whose marginal distribution is optimized once against $\text{pass@}1$, PTTS-RL is optimized against $\text{pass@}k$ specifically for the budget k, and different attempts within a single budget may use different modes.

Result 1: GRPO-RS collapses onto the dominant mode. Optimizing against $\text{pass@}1$ alone leaves no reason to preserve mode diversity: under mode dominance, the optimal marginal policy is unique and deterministic.

Proposition 1 (GRPO-RS selects only z^*) *Under Assumption 1, the unique $\text{pass@}1$-optimal marginal policy for the observed category c' is the point mass on the dominant mode, $\pi_{\text{rs}}(\cdot \mid c') = \delta_{z^*}$. Consequently, at every inference budget k, GRPO-RS issues all k attempts with mode z^*, and its residual error is*

$$\epsilon_k^{\text{rs}} = \sum_{c\in\mathcal{C}} \omega(c \mid c')\,(1 - R_{z^*c})^k.$$

Proof. Since $\text{pass@}1(\pi) = \sum_z \pi(z \mid c')\,\bar{R}_{zc'}$ is linear in π and $\sum_z \pi(z \mid c') = 1$,

$$\text{pass@}1(\delta_{z^*}) - \text{pass@}1(\pi) = \sum_{z\neq z^*} \pi(z \mid c')\left(\bar{R}_{z^*c'} - \bar{R}_{zc'}\right),$$

which is non-negative by Assumption 1 and zero if and only if $\pi(z \mid c') = 0$ for every $z \neq z^*$; hence δ_{z^*} is the unique maximizer. GRPO-RS therefore issues every attempt with z^*, and substituting $\pi_{\text{rs}} = \delta_{z^*}$ into $\text{pass@}k\left(\pi_{\text{rs}}^{(k)}\right)$ yields the closed form. □

Result 2: PTTS-RL diversifies beyond a finite budget. Optimizing against $\text{pass@}k$ directly leads to a qualitatively different behavior: beyond an explicit finite budget, the optimal allocation never concentrates on z^* alone. Let $z^\dagger \in \mathcal{Z}$ denote a complementary mode furnished by Assumption 2, i.e. $R_{z^\dagger c^\dagger} > R_{z^*c^\dagger}$ (note this forces $z^\dagger \neq z^*$), and define the complementarity gap on the worst category, the failure rate of z^* on its worst category, and its largest failure rate on any other category:

$$\Delta \triangleq R_{z^\dagger c^\dagger} - R_{z^*c^\dagger} > 0, \qquad q^\dagger \triangleq 1 - R_{z^*c^\dagger}, \qquad \rho \triangleq \max_{c\neq c^\dagger}(1 - R_{z^*c}).$$

Proposition 2 (PTTS-RL assigns mass outside z^*) *Under Assumptions 1 and 2, define the finite threshold*

$$k^*_{\text{ptts}} \triangleq 2 + \left\lfloor \frac{\ln\left(\omega(c^\dagger \mid c')\,\Delta\right)}{\ln\left(\rho/q^\dagger\right)} \right\rfloor,$$

with the convention $k^*_{\text{ptts}} \triangleq 2$ *when* $\rho = 0$*. Then for every budget* $k \geq k^*_{\text{ptts}}$*, every optimal allocation* $\mathbf{k}^{(k)}$ *satisfies* $\sum_{z \neq z^*} k_z^{(k)} \geq 1$*; equivalently, the induced policy* $\pi^{(k)}_{\text{ptts}}$ *assigns non-zero probability mass to modes other than* z^**.*

Proof. The threshold is well-defined and finite. First, $|\mathcal{C}| \geq 2$: if $\mathcal{C} = \{c^\dagger\}$, mode dominance would give $R_{z^\dagger c^\dagger} = \bar{R}_{z^\dagger c'} < \bar{R}_{z^* c'} = R_{z^* c^\dagger}$, contradicting Assumption 2; hence ρ is a maximum over a non-empty finite set. Next, $q^\dagger \geq \Delta > 0$ (as $R_{z^\dagger c^\dagger} \leq 1$), and $0 \leq \rho < q^\dagger$ since $c^\dagger$ is the single worst category of z^*. Finally, $\omega(c^\dagger \mid c') \in (0, 1)$ because the posterior is strictly positive on at least two categories, so $\omega(c^\dagger \mid c')\,\Delta \in (0, 1)$. For $\rho > 0$, both logarithms are therefore negative and $k^*_{\text{ptts}} \geq 2$ is finite.

Now fix any $k \geq 2$ and compare the pure allocation $\mathbf{k}^{\text{all}}$, placing all k slots on z^*, against the swapped allocation $\mathbf{k}^{\text{swap}}$, placing $k-1$ slots on z^* and one on $z^\dagger$. Since the corresponding failure products differ only in one factor,

$$\begin{aligned}\text{pass@}k(\mathbf{k}^{\text{swap}}) - \text{pass@}k(\mathbf{k}^{\text{all}}) &= \sum_{c \in \mathcal{C}} \omega(c \mid c')\,(1 - R_{z^* c})^{k-1}\,(R_{z^\dagger c} - R_{z^* c}) \\ &\geq\ \omega(c^\dagger \mid c')\,\Delta\,(q^\dagger)^{k-1} - \rho^{k-1},\end{aligned}$$

where the last step keeps the $c^\dagger$ term exactly and bounds every other term via $|R_{z^\dagger c} - R_{z^* c}| \leq 1$, $1 - R_{z^* c} \leq \rho$, and $\sum_{c \neq c^\dagger} \omega(c \mid c') \leq 1$. If $\rho = 0$, the right-hand side is positive for every $k \geq 2 = k^*_{\text{ptts}}$. Otherwise it is positive iff $(\rho/q^\dagger)^{k-1} < \omega(c^\dagger \mid c')\,\Delta$, i.e. iff $k - 1 > \ln(\omega(c^\dagger \mid c')\Delta)/\ln(\rho/q^\dagger)$, since dividing by $\ln(\rho/q^\dagger) < 0$ flips the inequality; this holds for every $k \geq k^*_{\text{ptts}}$ because $k^*_{\text{ptts}} - 1 = 1 + \lfloor \cdot \rfloor$ strictly exceeds the ratio. Hence the pure allocation is strictly suboptimal for every $k \geq k^*_{\text{ptts}}$, and since only finitely many allocations of size k exist, every optimal allocation places at least one slot on a mode other than z^*. □

Note that k^*_{ptts} is a sufficient budget rather than an exact transition point—the proof discards every favorable term outside $c^\dagger$—and it decreases as the worst category becomes more probable (larger $\omega(c^\dagger \mid c')$) or more fixable (larger Δ): hedging pays off sooner when the failure mode of z^* is both likely and repairable.

Result 3: PTTS-RL reduces error exponentially and strictly faster; GRPO-RS may not. Having characterized how each policy allocates its attempts, we now compare their residual errors and the exponential rates at which these decay. The two policies part ways on both counts: PTTS-RL drives its error to zero exponentially fast under Assumptions 1 and 2 alone, and with a strictly better error exponent, whereas GRPO-RS converges only when its dominant mode retains some success probability on the worst category. Throughout, let

$$r \ \triangleq\ \frac{1 - R_{z^\dagger c^\dagger}}{q^\dagger} \ \in\ [0, 1),$$

which is well-defined since $q^\dagger > 0$ and below one since $R_{z^\dagger c^\dagger} > R_{z^* c^\dagger}$ by Assumption 2.

Proposition 3 (Error rates and strict separation) *Under Assumptions 1 and 2:*

(i) $\epsilon_k^{\text{ptts}} \leq \epsilon_k^{\text{rs}}$ *for every budget* $k \geq 1$*.*

(ii) $\omega(c^\dagger \mid c')\,(q^\dagger)^k \leq \epsilon_k^{\text{rs}} \leq (q^\dagger)^k$ *for every* $k \geq 1$*, and hence*

$$\lambda_{\text{rs}} \ \triangleq\ \lim_{k \to \infty} \left(\epsilon_k^{\text{rs}}\right)^{1/k} \ =\ q^\dagger.$$

In particular, $\epsilon_k^{\text{rs}} \to 0$ *if and only if* $R_{z^* c^\dagger} > 0$*; otherwise* ϵ_k^{rs} *converges to the plateau* $\omega(c^\dagger \mid c') > 0$*: GRPO-RS never solves problems from the worst category.*

(iii) For every $\alpha \in (0, 1)$ *satisfying* $\rho^{1-\alpha} < q^\dagger$*, which exists since* $\rho < q^\dagger$*,*

$$\limsup_{k \to \infty} \left(\epsilon_k^{\text{ptts}}\right)^{1/k} \ \leq\ \max\left\{q^\dagger r^\alpha,\ \ \rho^{\,1-\alpha}\right\} \ <\ q^\dagger \ =\ \lambda_{\text{rs}}.$$

In particular, $\epsilon_k^{\text{ptts}} \to 0$ *at least exponentially fast even when* ϵ_k^{rs} *does not vanish, and PTTS-RL strictly improves the error exponent over GRPO-RS.*

Proof. (i) The pure allocation placing all k slots on z^* is feasible for the optimization defining $\mathbf{k}^{(k)}$, and its pass@k coincides with that of GRPO-RS by Proposition 1; optimality of $\mathbf{k}^{(k)}$ yields the claim.

(ii) Start from the closed form of Proposition 1. Since $c^\dagger$ is the single worst category of z^* by Assumption 2, every category satisfies $1 - R_{z^*c} \le q^\dagger$; bounding every term accordingly gives the upper bound, and retaining only the $c^\dagger$ term gives the lower bound, where $\omega(c^\dagger \mid c') > 0$ since the posterior is non-zero for every pair. Both bounds hold for every value of $R_{z^*c^\dagger}$, including $q^\dagger = 1$; taking k-th roots and using $\omega(c^\dagger \mid c')^{1/k} \to 1$ gives $\lambda_{\mathrm{rs}} = q^\dagger$. If $R_{z^*c^\dagger} > 0$, then $q^\dagger < 1$ and the upper bound vanishes. If instead $R_{z^*c^\dagger} = 0$, the $c^\dagger$ term is frozen at $\omega(c^\dagger \mid c')$, while every remaining term vanishes as $k \to \infty$ since $1 - R_{z^*c} \le \rho < q^\dagger = 1$; hence ϵ_k^{rs} converges to $\omega(c^\dagger \mid c')$.

(iii) Fix $\alpha \in (0, 1)$ with $\rho^{1-\alpha} < q^\dagger$; such α exists because $\rho^{1-\alpha} \to \rho < q^\dagger$ as $\alpha \to 0^+$. For every $k > \frac{1}{1-\alpha}$, so that $\lceil \alpha k \rceil < k$, consider the feasible allocation placing $\lceil \alpha k \rceil$ slots on $z^\dagger$ and the remaining $k - \lceil \alpha k \rceil$ slots on z^*. Its $c^\dagger$ term satisfies, using $r < 1$ and $\lceil \alpha k \rceil \ge \alpha k$,

$$\omega(c^\dagger \mid c')\,(q^\dagger)^{\,k-\lceil \alpha k \rceil}\,(1 - R_{z^\dagger c^\dagger})^{\lceil \alpha k \rceil} \;\le\; (q^\dagger)^k\, r^{\lceil \alpha k \rceil} \;\le\; \left(q^\dagger r^\alpha\right)^k .$$

Every other term vanishes if $\rho = 0$; otherwise, using $1 - R_{z^\dagger c} \le 1$, $1 - R_{z^* c} \le \rho < 1$, $\lceil \alpha k \rceil \le \alpha k + 1$, and $\sum_{c \ne c^\dagger} \omega(c \mid c') \le 1$, these terms contribute at most

$$\rho^{\,k - \lceil \alpha k \rceil} \;\le\; \rho^{\,(1-\alpha)k - 1} \;=\; \rho^{-1} \left(\rho^{\,1-\alpha}\right)^k .$$

Writing $\mu \triangleq \max\{q^\dagger r^\alpha,\, \rho^{1-\alpha}\}$, optimality of $\mathbf{k}^{(k)}$ over feasible allocations therefore certifies

$$\epsilon_k^{\mathrm{ptts}} \;\le\; \left(1 + \rho^{-1}\right) \mu^k \quad \text{when } \rho > 0, \qquad \epsilon_k^{\mathrm{ptts}} \;\le\; \mu^k \quad \text{when } \rho = 0.$$

Taking k-th roots gives $\limsup_k (\epsilon_k^{\mathrm{ptts}})^{1/k} \le \mu$. It remains to check $\mu < q^\dagger$: the first branch satisfies $q^\dagger r^\alpha < q^\dagger$ since $r < 1$ and $\alpha > 0$, and the second satisfies $\rho^{1-\alpha} < q^\dagger$ by the choice of α. Since $\mu < q^\dagger \le 1$, the certificate also shows $\epsilon_k^{\mathrm{ptts}} \to 0$ at least exponentially fast. □

The separation quantifies the value of hedging: GRPO-RS decays exactly at the rate at which its single mode clears the worst category, and whenever z^* is hopeless on that category, repeated sampling replays the same failure and the error plateaus at exactly its posterior mass. Devoting even a constant fraction of the budget to a complementary mode strictly improves the error exponent, with the residual error dominated by whichever category the mixed allocation covers worst.

A.1 Closed-Form Analysis of a Synthetic Example

Finally, we apply our framework to a two-category, two-mode example to derive analytical expressions for both pass@k and mode coverage. These results are visualized in Figure 2 of the main text.

Instantiation. Let $\mathcal{C} = \{c_1, c_2\}$ and $\mathcal{Z} = \{z_1, z_2\}$, with a prior probability of $\Pr[c(\mathbf{x}) = c_1] = p = 0.75$. We consider a "blind" policy as an extreme case of an imperfect policy, where the observed category $c'(\mathbf{x})$ is independent of the true category. Consequently, the posterior reduces to the prior: $\omega(c_1 \mid c') = p = 0.75$ and $\omega(c_2 \mid c') = 1 - p = 0.25$. We define the success rates as follows:

$$R_{z_1 c_1} = R_{z_2 c_2} = a = 0.40, \quad R_{z_1 c_2} = R_{z_2 c_1} = b = 0.02.$$

Thus, each mode is highly effective on exactly one category.

This instantiation satisfies the two critical assumptions. The posterior success rates are $\bar{R}_{z_1 c'} = pa + (1-p)b = 0.305$ for z_1, and $\bar{R}_{z_2 c'} = pb + (1-p)a = 0.115$ for z_2; as $\bar{R}_{z_1 c'} > \bar{R}_{z_2 c'}$, Assumption 1 holds with the dominant mode $z^* = z_1$. For z_1, then, $c^\dagger = c_2$ is the single worst category that is better served by $z^\dagger = z_2 \ne z_1$. Thus, Assumption 2 also holds.

GRPO-RS in closed form. By Proposition 1, GRPO-RS always selects the dominant mode z_1 (see Figure 2b), yielding:

$$\begin{aligned} \text{pass@}k\left(\pi_{\mathrm{rs}}^{(k)}\right) &= 1 - p \cdot (1-a)^k - (1-p) \cdot (1-b)^k \\ &= 1 - 0.75 \times 0.6^k - 0.25 \times 0.98^k. \end{aligned}$$

As visualized in Figure 2a, the performance curve quickly saturates on c_1 problems (as the 0.6^k term decays rapidly to near zero) and then only slowly improves on the remaining c_2 problems (due to the slow decay of the 0.98^k term).

PTTS-RL in closed form. By Proposition 2, PTTS-RL splits the budget across both modes once k is large enough. In this symmetric instance, the optimal allocation places $m(k) \approx k/2 - 1.12$ (rounded to an integer) slots on z_2, which becomes non-zero and breaks the mode collapse at $k = 4$, as visualized in Figure 2b. The pass@k is then derived as:

$$\begin{aligned}\text{pass@}k\big(\mathbf{k}^{(k)}\big) &= 1 - p\,(1-a)^{(k-m(k))}(1-b)^{m(k)} - (1-p)\,(1-b)^{(k-m(k))}(1-a)^{m(k)} \\ &= 1 - 0.75 \times 0.6^{(k-m(k))} \times 0.98^{m(k)} - 0.25 \times 0.98^{(k-m(k))} \times 0.6^{m(k)},\end{aligned}$$

which strictly exceeds $\text{pass@}k\left(\pi_{\text{rs}}^{(k)}\right)$ once $m(k) > 0$. Moreover, since PTTS-RL splits the inference budget approximately evenly ($m(k) \sim k/2$), the residual error decays at a rate of approximately $\sqrt{(1-a)(1-b)} = \sqrt{0.6 \times 0.98} \approx 0.77$ per attempt, significantly faster than the 0.98^k term of GRPO-RS, leading to the wide gap in Figure 2a.

B Training Details

For the PTTS-RL training, we initialize from the Qwen3-1.7B/4B-Base models, using the corresponding Qwen3-1.7B or Qwen3-4B models as executors. The execution budget is fixed as $4k$ tokens, and the branching factor k is set as 4. Our implementation is based on `verl` (Sheng et al., 2024). We train on the DAPO-Math-17K dataset (Yu et al., 2026) using 8x 80GB NVIDIA A100 GPUs. Each model is trained for 240 optimization steps, and we report results from the final checkpoint. Additional hyperparameters are shown in Table 2.

Hyperparameter	**Value**
Batch size	16
Responses per prompt	16
PPO clipping ratio (lower)	0.20
PPO clipping ratio (upper)	0.28
Learning rate	1×10^{-6}
Rollout temperature	1.0
Weight decay	0.1
Gradient clipping	1.0

Table 2: Training hyperparameters.

C Analysis Details

C.1 Outline Adherence Analysis

We use GPT-5-mini to evaluate how faithfully the executor follows the outline generated by the planner. Specifically, for each instance, we provide the judge with the problem statement, the planner-generated outline, and the corresponding executor-generated response, and ask it to assess whether the executor's primary reasoning path implements the strategy, decomposition, or perspective specified by the outline. The detailed prompt is shown in Prompt 3. The model assigns a score on a five-point scale, which we normalize to [0, 1] for ease of interpretation and comparison across settings.

C.2 Diversity Analysis by Clustering

We quantify the diversity of both outlines and solutions by clustering them into distinct conceptual groups and computing the metrics reported in Appendix C.3. We apply the same procedure separately to planner outlines and executor solutions.

Concretely, the clustering procedure consists of two steps. (1) **Concept extraction**: we prompt GPT-5-mini to identify the single most important mathematical or logical concept underlying each planner outline o_i and each executor response $\mathbf{y}_i$. The extracted concept may be a theorem, formula, invariant, transformation, or canonical solution strategy. The full prompt is shown in Prompt 4. (2) **Concept clustering**: after extracting the concepts independently, we provide GPT-5-mini with all extracted concepts for each problem and ask it to group concepts that represent the same underlying mathematical mechanism. The full prompt is shown in Prompt 5.

C.3 Diversity Metrics

For a given problem, consider k responses, either k outlines sampled from the planner or k solutions generated by repeated sampling or PTTS. We use the procedure described above in Appendix C.2 to assign each response i to a cluster c_i, and use the following metrics to quantify their diversity:

Overall diversity. We measure overall diversity by the number of distinct clusters represented among the k responses, formally defined as:

$$\mathcal{C} = \{c_i \mid i = 1, \ldots, k\}, \qquad \text{Overall diversity} = |\mathcal{C}|.$$

Intuitively, a larger value indicates that the responses cover a broader range of distinct reasoning approaches.

Useful diversity. Overall diversity can be easily hacked by producing responses that are superficially diverse but ultimately unhelpful for solving the problem. We therefore also measure diversity among responses that eventually lead to a correct answer. Let $r_i \in \{0, 1\}$ indicate whether response i leads to a correct solution, and define

$$\mathcal{C}^+ = \{c_i \mid r_i = 1\}.$$

We normalize the number of distinct clusters among correct responses by the total number of correct responses:

$$\text{Useful diversity} = |\mathcal{C}^+| \,/\, \sum_{i=1}^{k} r_i.$$

Thus, useful diversity is high when correct responses span distinct reasoning approaches, and low when many correct responses concentrate on the same approach.

C.4 Ranked-Averaged Correlation Test

When reporting an aggregated correlation between two metrics, such as diversity and $\mathrm{pass@}k$, we have M groups of responses for each problem ($M = 16$ in §4.2), each with a diversity score and a $\mathrm{pass@}k$ value. Typically, each problem provides too few points to reliably estimate a correlation on its own. We therefore leverage data across multiple problems to identify a consistent trend. However, points from different problems cannot be pooled directly: simply concatenating them introduces problem difficulty as a confounding factor, while directly averaging across problems is not meaningful because there is no natural correspondence between groups from different problems.

We therefore use a rank-averaging approach that aligns points across problems by their *relative position* rather than their raw values. For a variable a (e.g., outline diversity), let $a_{q,i}$, $i = 1, \ldots, M$, denote its values for problem q, and let $b_{q,i}$ denote the corresponding values of a target variable b (e.g., $\mathrm{pass@}k$). For each problem, we sort the M groups by a. Let $a_{q,(j)}$ denote the j-th smallest value of a, and let $b_{q,(j)}$ denote the b-value of the same group. We then average both variables across problems at each rank j:

$$\bar{a}_j = \frac{1}{Q} \sum_{q=1}^{Q} a_{q,(j)}, \qquad \bar{b}_j = \frac{1}{Q} \sum_{q=1}^{Q} b_{q,(j)}.$$

Finally, we compute the Spearman correlation between $\{\bar{a}_j\}_{j=1}^{M}$ and $\{\bar{b}_j\}_{j=1}^{M}$.

We apply this procedure to three pairs in §4.2: outline diversity vs. response diversity, outline diversity vs. $\mathrm{pass@}k$, and response diversity vs. $\mathrm{pass@}k$.

D QUALITATIVE EXAMPLES

This section presents additional qualitative examples, including an example of diversity induced by PTTS in Figure 10 and an example of outline adherence and drift in Figure 11.

Problem: Let $\triangle ABC$ be a right triangle with $\angle A = 90^\circ$ and $BC = 38$. There exist points K and L inside the triangle such that $AK = AL = BK = CL = KL = 14$. The area of the quadrilateral $BKLC$ can be expressed as $n\sqrt{3}$ for some positive integer n. Find n.

Answer: 104

Repeated Sampling: All four branches develop the same line of reasoning: they place A at the origin with B and C on the axes, derive $AB^2 + AC^2 = 1444$, and locate K and L as intersections of equal-radius circles. None resolves the resulting system. Across the full pool of 64 branches, only two distinct solution clusters appear.

Pass@4 = 0

PTTS-RL

Branch 1

Outline: **Symmetry and Equilateral Triangles**: Since $AK = AL = BK = CL = 14$, we can consider that points K and L form two equilateral triangles with sides of length 14. ... we can infer that the quadrilateral $BKLC$ is composed of two congruent triangles ...

Reasoning: First, the problem mentions that $AK = AL = BK = CL = 14$. So, points K and L are each 14 units away from A and from B and C respectively. Also, KL is 14. **The solution outline suggests symmetry and equilateral triangles.** Let me try to visualize this. ... **But the outline says that quadrilateral $BKLC$ is composed of two congruent triangles.** Maybe triangles BKC and LKC? Wait, not sure. Alternatively, maybe triangles BKL and LKC? Hmm. Alternatively, **since AKL is equilateral, and $BK = CL = 14$, maybe there's some symmetry here.** ...

Final answer: none. **Correct:** ×

Branch 2

Outline: **Coordinate Geometry**: We can place the triangle $\triangle ABC$ in a coordinate plane, with point A at the origin $(0, 0)$, point B on the x-axis, and point C on the y-axis. ... calculate the area of quadrilateral $BKLC$ using the Shoelace formula or the determinant method ...

Reasoning: **First, let me recall the solution outline. The problem suggests using coordinate geometry.** So I should place triangle ABC on a coordinate plane. Let me try to visualize that. **Since angle A is 90°, I can put point A at $(0, 0)$, point B on the x-axis, so B is $(b, 0)$, and point C on the y-axis, so C is $(0, c)$.** Then, BC is the hypotenuse, so the distance between B and C is $\sqrt{b^2 + c^2} = 38$. So that gives me the equation $b^2 + c^2 = 38^2 = 1444$. ... **So now, coordinates of K are $(b/2, \sqrt{196 - b^2/4})$ and coordinates of L are $(\sqrt{196 - c^2/4}, c/2)$.** Now, we need to use the KL distance equation ...

Final answer: none. **Correct:** ×

Branch 3

Outline: **Trigonometric Relationships and Angles**: ... if we denote the angle $\angle BAK$ as θ, we can use trigonometric identities to express the lengths of the segments BK and CL in terms of θ. ...

Reasoning: Alternatively, maybe I can consider triangles ABK and ALC. Since $AK = BK = 14$, triangle ABK is isoceles with $AK = BK$. ... **Therefore, angle at A is angle BAK, which I denoted earlier as θ. Then, using the law of cosines:** ... Simplify: $196 = c^2 + 196 - 28c\cos(\theta) \Rightarrow 0 = c^2 - 28c\cos(\theta) \Rightarrow c^2 = 28c\cos(\theta) \Rightarrow c = 28\cos(\theta)$. Therefore, $\cos(\theta) = c/28$. ... **Therefore, coordinates of K are $(c/2, 14\sin(\theta)) = (14\cos(\theta), 14\sin(\theta))$** ...

Final answer: none. **Correct:** ×

Branch 4

Outline: **Geometric Transformations and Similarity**: ... if we rotate point K around point A by 120° (a characteristic angle in equilateral triangles), we can create a new point that might help in understanding the relationship between K and L. ...

Reasoning: **The solution outline suggests using geometric transformations, like rotations. Maybe a rotation by 120 degrees?** Since 120 degrees is related to equilateral triangles, which have angles of 60 degrees, and maybe that could help with the distances here. Let me think. ... **Alternatively, maybe there's a rotation that maps K to L or something like that. ... Suppose we rotate point K around A by 60 degrees. If that rotation maps K to L, then $AK = AL = 14$, and angle between AK and AL is 60 degrees. Then triangle AKL would be equilateral, which matches the given $KL = 14$.** So that seems possible. ... But regardless, the area of $BKLC$ came out to be $104\sqrt{3}$. Given that the steps are correct and the logic holds, I think that is the answer. Therefore, the value of n is 104.

Final answer: 104. **Correct:** ✓

Pass@4 = 1

Figure 10: **Qualitative example of diversity induced by PTTS**, generated by PTTS-RL with Qwen3-4B. Reasoning parts that explicitly follow the given outline are **highlighted**. The four reasoning traces cover distinct directions, with one reaching the correct final answer.

Problem: Find the sum of all real numbers r such that there is at least one point where the circle with radius r centered at $(4, 39)$ is tangent to the parabola with equation $2y = x^2 - 8x + 12$.

Answer: 50

Outline: **Geometric Distance Approach**: Begin by considering the geometric interpretation of tangency between a circle and a parabola. **The distance from the center of the circle $(4, 39)$ to any point on the parabola must equal the radius r** of the circle at the point of tangency. **Define the distance formula** between the center of the circle and a general point (x, y) on the parabola. This distance will be expressed in terms of x and y, with y being determined by the parabola's equation. Since the circle and parabola are tangent, **the distance equation will have exactly one solution for x**. This condition will provide a relationship between x, y, and r. **Use the discriminant of the resulting quadratic equation in x** to ensure that there is exactly one solution, which implies that **the discriminant must be zero**. This will give us a condition on r without needing to solve for x explicitly. Solve the condition on r obtained from the discriminant to find the possible values of r. The sum of these values will be the answer.

Reasoning: First, I need to recall what it means for a circle and a parabola to be tangent. Tangent means they touch at exactly one point. So, the distance from the center of the circle to the parabola must be equal to the radius r, and there should be exactly one such point.

The solution outline suggests using the geometric distance approach. So, I should start by considering a general point (x, y) on the parabola. Since the parabola's equation is given as $2y = x^2 - 8x + 12$, I can express y in terms of x. ... Dividing both sides by 2: $y = (x^2 - 8x + 12)/2$. So, any point on the parabola can be represented as $(x, (x^2 - 8x + 12)/2)$.

Now, the distance from the center of the circle $(4, 39)$ to this point (x, y) should be equal to the radius r. The distance formula is $\sqrt{(x-4)^2 + (y-39)^2} = r$. But since y is expressed in terms of x, I can substitute that into the distance formula. ... To eliminate the square root, I can square both sides: $(x-4)^2 + \left[(x^2 - 8x + 12)/2 - 39\right]^2 = r^2$.

Now, this equation should have exactly one solution for x because the circle and parabola are tangent at that point. Therefore, the equation must have exactly one real root. To find the condition for this, I need to simplify the equation and then find the discriminant of the resulting quadratic (or higher degree) equation in x. ... So, rearranged: $x^4 - 16x^3 - 64x^2 + 1024x + (4420 - 4r^2) = 0$.

Now, this is a quartic equation in x. ... **However, the solution outline says that the equation must have exactly one solution for x, which would correspond to the case where they are tangent.** However, perhaps the equation is a quartic, which can have multiple roots. But for tangency, maybe there is a multiple root? So, the quartic equation should have a multiple root, which would mean that the equation and its derivative both equal zero at that root. **But the solution outline says that the discriminant of the resulting quadratic equation in x must be zero. Wait, but the equation we have is quartic. Hmm, maybe I made a mistake here. Let me check again.** ... But that seems correct. However, the problem mentions that the discriminant of the resulting quadratic equation in x must be zero. But here, we have a quartic equation. So maybe there's a miscalculation here. **Alternatively, perhaps there is a different approach.** ... But how do I find the values of r for which the quartic equation has at least one multiple root? ... Alternatively, maybe the equation we derived (the quartic) can be factored. Let me check if that's possible. ... But this seems complicated. Maybe there's a better way. Let me think again.

Alternatively, perhaps I can use calculus. If the circle and parabola are tangent, then they have a common point and their tangent lines at that point are the same. So, maybe I can set up the system of equations for the circle and the parabola, and then impose that their derivatives are equal at the point of intersection. Let me try that approach. ... $dy/dx = [2x - 8]/2 = x - 4$. Now, find the derivative of the circle implicitly. ... $dy/dx = -(x-4)/(y-39)$. At the point of tangency, the derivatives must be equal, so: $x - 4 = -(x-4)/(y-39)$. Assuming that $x \neq 4$ (if $x = 4$, then the derivative of the parabola is 0, and we can check separately). So, if $x \neq 4$, then we can divide both sides by $(x-4)$: $1 = -1/(y-39)$. Therefore, $y - 39 = -1 \Rightarrow y = 38$. ... But x is $4 \pm 4\sqrt{5}$, so $(x-4)^2$ is $(\pm 4\sqrt{5})^2 = 16 \cdot 5 = 80$. Therefore, the distance is $\sqrt{80 + 1} = \sqrt{81} = 9$. Therefore, $r = 9$. ... If $x = 4$, then from the parabola equation, $y = (16 - 32 + 12)/2 = (-4)/2 = -2$. So, the point is $(4, -2)$. The distance from $(4, 39)$ to $(4, -2)$ is $\sqrt{0 + (39 - (-2))^2} = \sqrt{41^2} = 41$. Therefore, $r = 41$. ... Therefore, there are two values of r: 9 and 41. Their sum is $9 + 41 = 50$. So, the answer should be 50.

But wait, why did the earlier approach with the quartic equation not lead me to this? Because when I derived the quartic equation, I considered that for tangency, the equation has exactly one solution. However, in reality, there are two different radii: one corresponding to the point $(4, -2)$ and another corresponding to the points $(4 \pm 4\sqrt{5}, 38)$. ...

Final answer: 50. **Correct:** ✓

Figure 11: **Qualitative example of outline adherence and drift**, from PTTS-RL with Qwen3-4B. Reasoning parts that explicitly follow the outline are marked in **blue**, while parts that deviate are marked in **orange**. The executor initially follows the outline but deviates when the outlined approach fails, eventually reaching the correct answer. Such recovery can assign positive reward to a poor outline, weakening the training signal, as discussed in §3.3.

E PROMPTS

This section presents the prompts used for planning, execution, and analysis, including the PTTS planner prompt in Prompt 1, the executor prompt in Prompt 2, the outline adherence evaluation prompt in Prompt 3, and the concept extraction and clustering prompts in Prompts 4 and 5.

```
[SYSTEM]

You are an annotator tasked with generating multiple high-level solution outlines for a
math problem.

Your goal is to explore different perspectives, strategies, or conceptual approaches that
could be used to solve the problem. Based on this, produce {num_suboutlines} distinct
outlines that could independently guide a solver from start to finish.
* You must NOT solve the problem.
```

```
* You must NOT compute values, simplify expressions, or use algebra.
* Any calculation makes the output invalid.

Format output as a numbered list (1. - {num_suboutlines}.), where each item is an outline.

[USER]

{question}
```

Prompt 1: Prompt used by the PTTS planner.

```
[SYSTEM]

You are a math problem solver. You are given a math problem and a solution outline. Follow
the outline carefully to solve the problem step by step. Show your work and put your final
answer in \boxed{}.

[USER]

Problem:
{question}

Solution Outline:
{outline}

Now solve the problem by following this outline:
```

Prompt 2: Prompt used by the PTTS executor.

```
[SYSTEM]

You are an expert evaluator for mathematical reasoning and
outline-conditioned generation.

You will be given:
1. A math problem.
2. An outline provided to a model.
3. A solution response generated by the model conditioned on that outline.

Determine whether the solution response follows the provided outline.

Evaluation rules:
- Do not assign a high following score merely because the solution is correct.
- Do not assign a low following score merely because the solution is incorrect.
- A response follows an outline when its primary reasoning path uses the strategy,
decomposition, or perspective described by the outline.
- A response may introduce additional details while still following the outline, provided
that those details are consistent with the outlined approach.
- A response that uses a substantially different method should receive a low following
score, even if it reaches the correct answer.
- If the outline or response is too unclear, malformed, or irrelevant to assess, use
CANNOT_JUDGE.

Outline following rubric:
5 = FOLLOWS_EXACTLY: Clearly follows the intended strategy.
4 = MOSTLY_FOLLOWS: Mainly follows the outline but adds, skips, or changes minor steps.
3 = PARTIALLY_FOLLOWS: Uses some ideas from the outline but substantially deviates.
2 = DOES_NOT_FOLLOW: Uses a different strategy or largely ignores the outline.
1 = CANNOT_JUDGE: The outline or response is too unclear, malformed, or irrelevant to
assess.

Return only valid JSON using the following schema:

{
  "outline_following_score": <integer from 1 to 5>,
  "outline_following_label":
    "<FOLLOWS_EXACTLY | MOSTLY_FOLLOWS | PARTIALLY_FOLLOWS |
      DOES_NOT_FOLLOW | CANNOT_JUDGE>",
  "outline_following_note": "<brief explanation>",
}

[USER]

[QUESTION]
{question}
[/QUESTION]
```

```
[OUTLINE]
{outline}
[/OUTLINE]

[OUTLINE_CONDITIONED_SOLUTION]
{response}
[/OUTLINE_CONDITIONED_SOLUTION]

Evaluate the outline-following behavior according to the system instructions. Return only
valid JSON.
```

Prompt 3: Prompt used to evaluate outline following.

```
You are given a math problem and a full solution response or outline generated by a model.

The response may be correct or incorrect.

Your task is to identify the SINGLE most important mathematical or logical concept, theorem
, canonical formula, invariant, transformation, or strategy that makes the attempted
solution or outline possible.

Important:
- Summarize the concept actually used or attempted in the response.
- Do NOT repair the response into a better concept.
- Do NOT introduce a concept that is not supported by the response.
- If the response is flawed, still identify the key concept the response attempted to use.
- If the response uses multiple concepts, choose the one without which the attempted
solution would not work, usually the first pivotal step.
- Choose the narrowest concept that still covers the attempted solution.
  - Good: "Pythagorean Theorem", "Vieta's Formulas", "Modular Invariant", "Complement
Counting".
  - Bad: "Geometry", "Algebra", "Number Theory", "Counting".

Definition of a valid key concept:
- It should be a specific mathematical concept, theorem, formula, invariant, transformation
, or canonical strategy.
- It should explain the core mechanism of the attempted solution.
- It should not be a full outline.
- It should not include the final answer.
- It should not include detailed computations, equation chains, or numerical derivations.
- It should be specific enough to distinguish this solution path from other possible
approaches.

Formatting rules:
- The concept name should use Title Case and singular form when possible.
- Prefer standard mathematical names when available.
- If no standard theorem applies, use a concise strategy label, such as "Casework on
Remainders", "Symmetry Reduction", "Complement Counting", or "Bounding Argument".
- Do not use overly broad labels such as "Algebra", "Geometry", "Combinatorics", or "Number
 Theory" unless the response is too unclear to identify a narrower concept.

Return ONLY valid JSON in the following format:
{{
  "concept": "<single most important concept name>",
  "evidence": "<one short sentence explaining why this concept is pivotal in the attempted
response>",
  "confidence": <integer from 1 to 5>
}}

Confidence score:
5 = clearly identifiable and specific concept.
4 = mostly clear concept, with minor ambiguity.
3 = plausible concept, but response uses several competing ideas or is partially unclear.
2 = weak guess because the response is flawed, vague, or inconsistent.
1 = cannot meaningfully identify a concept.

[QUESTION]
{question}
[/QUESTION]

[FULL_SOLUTION_RESPONSE]
{response}
[/FULL_SOLUTION_RESPONSE]
```

Prompt 4: Prompt used to extract the pivotal concept from each executor response.

```
You are given a math problem and a list of candidate key concepts extracted from model-
generated solution responses or outlines.

Some source responses may be correct and some may be incorrect.

Your task is to cluster the candidate concepts by the underlying mathematical or logical
concept used in the attempted solution.

Clustering rules:
- Group concepts together if they refer to essentially the same pivotal mathematical idea,
theorem, invariant, transformation, or canonical strategy.
- Merge synonymous or near-synonymous names.
  - Example: "Modulo Invariant", "Invariant Modulo 5", and "Residue Class Invariant" may
belong together if they describe the same attempted idea.
- Separate concepts if they represent meaningfully different solution mechanisms, even if
they are from the same broad field.
  - Example: "Vieta's Formulas" and "Discriminant Condition" should usually be separated.
  - Example: "Complement Counting" and "Inclusion-Exclusion Principle" should be separated
unless the evidence shows they refer to the same pivotal step.
- Cluster by attempted concept, not by whether the source response was correct.
- A cluster may contain concepts extracted from both correct and incorrect source responses
.
- Do NOT repair an incorrect or vague concept into a better one.
- Do NOT introduce a new concept that is not represented by the cluster members.
- Prefer the narrowest canonical concept name that covers the cluster.

For each cluster, write:
- "canonical_concept": the standard or clearest concept name.
- "member_indices": the indices of candidate concepts in this cluster.
- "canonical_description": one short sentence describing the shared concept.

Canonical concept rules:
- Use Title Case and singular form when possible.
- Prefer standard mathematical names when available.
- If no standard theorem applies, use a concise strategy label, such as "Casework on
Remainders", "Symmetry Reduction", "Complement Counting", or "Bounding Argument".
- Do not use overly broad labels such as "Algebra", "Geometry", "Combinatorics", or "Number
 Theory" unless the cluster is genuinely too vague to identify a narrower concept.
- Do not include final answers, computations, equation chains, or problem-specific
numerical values.

Return ONLY valid JSON in this exact schema:
{{
  "clusters": [
    {{
      "cluster_id": 1,
      "canonical_concept": "<Title Case concept name>",
      "member_indices": [1, 3],
      "canonical_description": "<one short sentence describing the shared concept>"
    }}
  ]
}}

Example:

[EXAMPLE_QUESTION]
Find the number of positive integers n satisfying a certain divisibility condition.
[/EXAMPLE_QUESTION]

[EXAMPLE_CANDIDATE_CONCEPTS]
1. Concept: Modular Arithmetic Filtering
   Evidence: The solution restricts possible values of n using congruences modulo small
integers.
2. Concept: Casework on Parity
   Evidence: The solution splits n into odd and even cases and eliminates impossible cases.
3. Concept: Congruence Class Filtering
   Evidence: The solution uses residues modulo small primes to narrow the candidate values.
4. Concept: Exhaustive Enumeration
   Evidence: The solution checks all feasible candidates directly against the condition.
[/EXAMPLE_CANDIDATE_CONCEPTS]

[EXAMPLE_EXPECTED_OUTPUT]
{{
  "clusters": [
    {{
      "cluster_id": 1,
      "canonical_concept": "Modular Arithmetic Filtering",
      "member_indices": [1, 3],
      "canonical_description": "Use congruence constraints to narrow the possible values
before checking the original condition."
    }},
```

```
    {{
      "cluster_id": 2,
      "canonical_concept": "Casework On Parity",
      "member_indices": [2],
      "canonical_description": "Split the variable into parity cases and analyze each case
separately."
    }},
    {{
      "cluster_id": 3,
      "canonical_concept": "Exhaustive Enumeration",
      "member_indices": [4],
      "canonical_description": "Check all feasible candidates directly against the required
 condition."
    }}
  ]
}}
[/EXAMPLE_EXPECTED_OUTPUT]

Now cluster the actual candidate concepts below.

[QUESTION]
{question}
[/QUESTION]

[CANDIDATE_CONCEPTS]
{candidate_concepts}
[/CANDIDATE_CONCEPTS]
```

Prompt 5: Prompt used to cluster response concepts for each problem.

F AI Assistants In Research Or Writing

We used AI assistants solely for stylistic improvements in writing, such as improving clarity, grammar, and phrasing. We did not use AI assistants for coding, brainstorming, research design, data analysis, interpretation of results, or any other critical intellectual contribution.