

Variational Co-Evolution via Reinforcement Learning

Qiahao Zheng^{1*} Hancong Jin^{2*} Junbo Niu^{1*}
Yinjie Wang³ Wentao Zhang^{1†} Ruixun Zhang^{1†}

¹Peking University

²University of Electronic Science and Technology of China

³University of Chicago

Abstract

Despite recent progress, large language models still exhibit limitations and remain brittle on challenging reasoning tasks, as effective reasoning depends not only on the underlying solution policy but also on the capability to construct prompts that reliably elicit correct reasoning. Most existing methods optimize these two components separately, focusing either on prompt rewriting or solely on strengthening the solving policy, which makes it hard to build a persistent, mutually reinforcing loop between prompting and solution generation. To address this limitation, we propose **Variational Co-Evolution (VCE)**, a novel co-evolving framework that jointly improves prompt generation and solution generation with the explicit goal of strengthening both capabilities simultaneously. VCE formulates co-evolution through a reinforcement learning framework with a variational inference, treating prompts as adaptive decisions jointly optimized with solution policies. By enabling coherent updates of prompting and reasoning under a unified objective, VCE provides a stable and scalable paradigm for co-evolving prompts and solvers on challenging reasoning tasks. Experiments across multiple benchmarks demonstrate strong and consistent gains, validating the effectiveness of co-evolution for improving both prompt construction and solution accuracy.

1 Introduction

The large language models (LLMs) have recently shown significant improvement in their capabilities to solve different natural language tasks, including mathematical reasoning and code generation (Shao et al., 2024; Guo et al., 2025; Zhang et al., 2024; Wang et al., 2025). Despite these advances, achieving reliable performance on challenging reasoning problems remains nontrivial, as success often depends not only on the underlying reasoning policy but also on how the task is elicited and guided through prompting. In this work, we aim to improve LLMs’ reasoning performance, which naturally calls for jointly strengthening two tightly coupled capabilities: solution generation and prompt generation for eliciting effective reasoning.

Recent post-training efforts show that reinforcement learning and preference-based optimization can strengthen long-horizon reasoning, supported by stable policy optimization (Schulman et al., 2017) and modern preference-based objectives (Rafailov et al., 2023; Meng et al., 2024). However, these solver-centric approaches typically optimize the solution policy under a largely fixed prompting distribution, treating prompts as externally provided context, which can bottleneck reasoning improvements and induce train-deploy mismatch.

In parallel, appropriate prompting can unlock latent reasoning ability in LLMs without modifying model parameters, from chain-of-thought to more structured elicitation strategies (Wei et al., 2022; Yao et al., 2023a; Zheng et al., 2024). Automated prompt optimization and self-improvement therefore use LLMs to search, refine, or evolve prompts (Yang et al., 2024; Wang et al., 2024; Fernando et al., 2024), but are often applied as an outer loop around a

*Equal contribution.

†Corresponding authors.

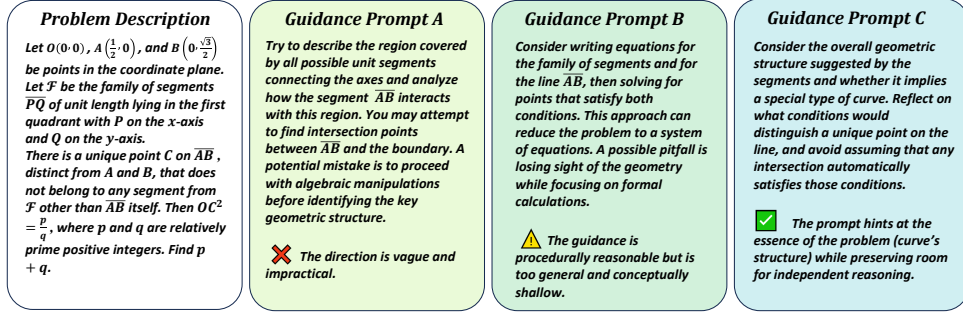


Figure 1: An example of a problem description with three generated guidance prompts. The first guidance prompt is vague and impractical, leading to incorrect solutions. The second guidance is procedurally reasonable but is too general and conceptually shallow, making it difficult to infer the correct solution. The final prompt clearly highlights the key issues and frequently yields correct solutions.

fixed solver (or a black box), making it hard to keep prompts effective as the solver evolves and to translate prompt gains into durable improvements after further optimization.

These two lines of progress therefore advance largely in isolation, which hinders the emergence of a stable feedback loop between prompting and reasoning. Better prompts are not guaranteed to remain optimal as the reasoning capability improves, and stronger solvers may still underperform when elicitation is outdated, mis-specified, or distributionally shifted. This motivates a unified training paradigm that couples prompt generation and solution learning in a closed loop, so that prompts and solvers can continuously reinforce each other rather than being optimized separately.

To enable this closed-loop feedback in a principled and scalable way, we propose **Variational Co-Evolution (VCE)**, a unified framework that jointly optimizes prompt generation and solution generation under a single reinforcement-learning objective.

VCE is grounded in a variational perspective on co-evolution: prompt generation is modeled as a latent decision process that mediates solution quality, which leads to a principled learning objective for optimizing both prompting and solving together. Under this view, the useful prompts are those that are refined in light of observed solution outcomes. We operationalize this principle with self-reflection: when solutions generated under an initial prompt fail, the model reflects on the failures and produces a revised prompt that better elicits correct reasoning. This makes prompt improvement directly target what matters for downstream accuracy, rather than relying on task-agnostic heuristics.

Building on this formulation, VCE performs coupled reinforcement-learning updates for both components. Prompt generation is trained to prefer prompts that induce higher solution quality, while the solver is trained with rewards that are calibrated by the quality of the elicitation, reducing mismatch between training-time prompts and deployment-time self-generated prompts. The overall optimization follows a stable, clipped policy-optimization style and is compatible with modern scalable RL variants used for reasoning models.

Empirically, we focus on mathematical reasoning, where performance is highly sensitive to elicitation and long-horizon solution quality. We evaluate VCE on Qwen3-1.7B, 4B, and 8B using a suite of competitive reasoning benchmarks. Across model scales and benchmarks, VCE consistently improves solution accuracy and Pass@k robustness over base models and solver-only reinforcement learning baselines. Moreover, VCE yields stronger self-generated prompts: prompts produced by the co-evolving model transfer more effectively and produce larger gains when applied to a solver, providing evidence that VCE improves elicitation quality rather than merely tuning the solver under a fixed prompt distribution.

2 Related Work

Prompting The prompt engineering has demonstrated its effectiveness and unique value for large language models. To harness the reasoning capabilities of large language models,

(Wei et al., 2022) introduced the Chain-of-Thought (CoT) prompting method, which attaches a sequence of reasoning steps to the input question before generating the final answer. This straightforward few-shot prompting method substantially improves model performance on tasks that require complex reasoning. To reduce the need for manual annotations, (Kojima et al., 2022) introduced Zero-shot-CoT, which enables large language models to generate reasoning steps without relying on exemplars. At present, there have been numerous related works (Zhou et al., 2023a; Yao et al., 2023a; Besta et al., 2024; Chen et al., 2023) such as Plan-and-Solve Prompting (PS) (Wang et al., 2023a) in the direction of designing more effective prompting strategies. However, due to the limitations of the aforementioned prompt engineering methods, which are hand designed, current research on prompting is gradually focusing on prompt optimization for automated prompt generation. For the diversity of methods and the extension of tuning prompts to more domains, recent works (Yang et al., 2024; Fernando et al., 2024), such as Automatic Prompt Engineer (APE) (Zhou et al., 2023b), have explored leveraging large language models for prompt optimization. On this foundation, the self-feedback mechanism of large language models can be employed to reformulate prompts and yield more effective instructions (Xiang et al., 2025). Moreover, recent works suggest that self-reflection encourages large language models to introspect on their own outputs (Pryzant et al., 2023; Wang et al., 2024; Chen et al., 2024). Yet, in contrast to existing works, VCE refines the associated prompts by self-reflection and then generates more refined solutions. Thus, through reinforcement learning of prompts and solutions, the co-evolution of the two generative capabilities can be achieved.

Reinforcement learning for LLMs Proximal Policy Optimization (PPO) (Schulman et al., 2017) is a widely used reinforcement learning algorithm that stabilizes policy updates by constraining them within a clipped objective, making it particularly effective for fine-tuning large language models. Methods such as Direct Preference Optimization (DPO) (Rafailov et al., 2023) and its variants (Cen et al., 2025; Meng et al., 2024; Liu et al., 2020) serve as an alternative to reinforcement learning by removing the need for a reward model and directly aligning large language model outputs with human preferences, thereby improving efficiency. Recent advances in Group Relative Policy Optimization (GRPO) demonstrate strong scalability for large-scale reinforcement learning (Guo et al., 2025; Shao et al., 2024). Reinforcement learning also has some applications in prompt engineering and optimization (Deng et al., 2022; Batorski et al., 2025). However, our goal is not to directly compete with the above-mentioned reinforcement learning methods. In contrast, different reinforcement learning methods can be integrated into the prompt and solutions in our framework, thereby achieving the co-evolution of our methods.

3 Our Method

In this section, we propose Variational Co-Evolution, an innovative reinforcement learning framework designed based on variational inference, and through self-reflection, interactively enhanced the prompt generation and math solution capabilities of LLMs. Firstly, through variational inference, we demonstrate that better prompt words generated via reflective mechanisms are required during the coevolutionary process. We then theoretically derive the reward function for both prompt generation and math solution development. Finally, we introduce our novel variational collaborative reinforcement learning framework, VCE.

3.1 Variational Inference for Co-Evolution

Given the basic large language model $\mathcal{M}$, its target task $\mathcal{T}$. The policy of LLM for generating prompt under a given mathematical problem q is $\pi_\theta(p | q)$, while the policy for generating the observation D of the solution under the given question q and prompt p is denoted by $\pi_\theta(D | p, q)$. More and more work indicate that a good prompt plays an important role in the output of LLM (Zheng et al., 2024; Madaan et al., 2023; Wang et al., 2023b; Yao et al., 2023b), which has led to the emergence of many methods for optimizing the prompt (Cheng et al., 2024; Hu et al., 2024; Do et al., 2024; Soylu et al., 2024). Therefore, compared with directly focusing on the maximum likelihood of $\pi_\theta(D | p, q)$ after the given question q and the prompt p , it is also necessary to consider the solution in the average sense through the

For the LLM's self-generating capability of prompts, we first expand on $\omega^*(p | q)$ by denoting $Z = \mathbb{E}_{p \sim \pi_{\theta_{\text{old}}}(\cdot | q)}[\pi_{\theta_{\text{old}}}(D | p, q)]$:

$$\omega^*(p | q) = \frac{\pi_{\theta_{\text{old}}}(p | q) \pi_{\theta_{\text{old}}}(D | p, q)}{Z} \propto \pi_{\theta_{\text{old}}}(p | q) \pi_{\theta_{\text{old}}}(D | p, q).$$

Naturally, to increase $\mathbb{E}_{p \sim \omega^*(\cdot | q)}[\log \pi_{\theta}(D | p, q)]$, we want to use reinforcement learning to raise expectations regarding the prompt policy by assigning higher reward to better prompt. To achieve a rigorous design of the reward function, we first calculate the gradient of $\mathbb{E}_{p \sim \omega^*(\cdot | q)}[\log \pi_{\theta}(D, p | q)]$ at $\theta = \theta_{\text{old}}$ based on the split of $\omega^*(p | q)$, which is given by

$$\begin{aligned} & \nabla_{\theta} \mathbb{E}_{p \sim \omega^*(\cdot | q)}[\log \pi_{\theta}(p | q)] \\ &= \mathbb{E}_{p \sim \pi_{\theta_{\text{old}}}(\cdot | q)} \left[\frac{\omega^*(p | q)}{\pi_{\theta_{\text{old}}}(p | q)} \nabla_{\theta} \log \pi_{\theta}(p | q) \right] \\ &= \frac{1}{Z} \mathbb{E}_{p \sim \pi_{\theta_{\text{old}}}(\cdot | q)} \left[\pi_{\theta_{\text{old}}}(D | p, q) \nabla_{\theta} \log \pi_{\theta}(p | q) \right]. \end{aligned} \quad (4)$$

Therefore, when using reinforcement learning, $\pi_{\theta_{\text{old}}}(D | p, q)$ serves as our fundamental reward function. As for the reasoning and problem solution capabilities of LLM about $\mathbb{E}_{p \sim \omega^*(\cdot | q)}[\log \pi_{\theta}(D | p, q)]$, naturally, we tend to generate correct solution observations. Let $a_{\theta}(p, q) = \mathbb{E}_{D \sim \pi_{\theta}(\cdot | p, q)}[\mathbf{1}_{\text{correct}}(D)]$, which is the accuracy rate of the LLM's solution based on the prompt p and the question q , where $\mathbf{1}_{\text{correct}}(D) \in \{0, 1\}$ is the ground truth test. Therefore, we only need to use reinforcement learning to increase $\mathbb{E}_{p \sim \omega^*(\cdot | q)}[\log a_{\theta}(p, q)]$. First of all, through the calculation in the appendix, we can obtain

$$\begin{aligned} \nabla_{\theta} a_{\theta}(p, q) &= \sum \pi_{\theta}(D | p, q) \nabla_{\theta} \log \pi_{\theta}(D | p, q) \mathbf{1}_{\text{correct}}(D) \\ &= \mathbb{E}_{D \sim \pi_{\theta}(\cdot | p, q)} [\mathbf{1}_{\text{correct}}(D) \nabla_{\theta} \log \pi_{\theta}(D | p, q)]. \end{aligned} \quad (5)$$

Next, we calculate the gradient of $\mathbb{E}_{p \sim \omega^*(\cdot | q)}[\log a_{\theta}(p, q)]$ by substituting equation 5. Through the variational proof in the appendix, the gradient at $\theta = \theta_{\text{old}}$ is given by

$$\begin{aligned} & \nabla_{\theta} \mathbb{E}_{p \sim \omega^*(\cdot | q)}[\log a_{\theta}(p, q)] \\ &= \mathbb{E}_{p \sim \omega^*(\cdot | q)} \left[\frac{1}{a_{\theta_{\text{old}}}(p, q)} \nabla_{\theta} a_{\theta}(p, q) \right] \\ &= \mathbb{E}_{p \sim \omega^*(\cdot | q), D \sim \pi_{\theta}(\cdot | p, q)} \left[\frac{1}{a_{\theta_{\text{old}}}(p, q)} (\mathbf{1}_{\text{correct}}(D) - a_{\theta_{\text{old}}}(p, q)) \nabla_{\theta} \log \pi_{\theta}(D | p, q) \right]. \end{aligned} \quad (6)$$

At this point, we can increase the target expectation by assigning more rewards to the correct solution through reinforcement learning. Here, we use $\frac{1}{a_{\theta_{\text{old}}}(p, q)} (\mathbf{1}_{\text{correct}}(D) - a_{\theta_{\text{old}}}(p, q))$ as the reward function for the solution.

3.3 Co-evolving Generative Prompt and solution with RL

For each task $q \in \mathcal{T}$ which is paired with ground truth test, the LLM generates n prompts p_j by the prompt policy $\pi_{\theta}(\cdot | q)$, where $1 \leq j \leq n$. Conditioned on each p_j , the LLM generates m solutions $s_{j,k}$, where $1 \leq k \leq m$. Each solution $s_{j,k}$ is evaluated by the ground-truth test $\mathbf{1}_{\text{correct}}(\cdot)$ for the task q , producing a binary outcome $b_{j,k} = \mathbf{1}_{\text{correct}}(s_{j,k}) \in \{0, 1\}$. For each p_j that exists with k such that $b_{j,k} = 0$, let the LLM performs self-reflection on all incorrect solutions to produce a refined prompt p_j^* . For each p_j^* , the LLM generates l new solutions $s_{j,t'}^*$ and evaluate them by the ground truth test, yielding outcomes $b_{j,t'}^*$, where $1 \leq t' \leq l$.

For each generated prompt p_j , where $1 \leq j \leq n$, we assign higher rewards to prompt with more solutions that pass the ground truth test, which encourages the LLM to generate prompt with higher accuracy. The reward is defined as:

$$R_{p_j}^{\text{prompt}} = \frac{1}{m} \sum_{k=1}^m b_{j,k}. \quad (7)$$

Table 1: Performance of baseline models, GRPO-trained models, and VCE-trained models on four mathematical reasoning benchmarks. The table reports average accuracy (Avg, %) over 16 test runs and Pass@16 accuracy (P@16, %) for direct answering and for answering with self-generated guidance prompts (Avg* and P@16*). Bold indicates the best result.

Model	MATH500				AIME2024				AIME2025				AMC2024			
	Avg	Avg*	P@16	P@16*	Avg	Avg*	P@16	P@16*	Avg	Avg*	P@16	P@16*	Avg	Avg*	P@16	P@16*
Qwen3-8B	91.0	91.2	95.8	95.8	59.8	60.0	80.0	83.3	46.0	47.3	70.0	73.3	68.6	69.0	82.2	80.0
Qwen3-8B + GRPO	92.0	92.0	95.2	95.4	62.3	62.9	83.3	83.3	51.7	50.8	76.7	76.7	71.3	71.9	82.2	84.4
Qwen3-8B + VCE	93.3	94.0	98.0	98.8	64.8	66.0	90.0	96.7	54.4	55.2	83.3	86.7	74.2	74.9	91.1	93.3
Qwen3-4B	88.4	88.7	94.2	94.6	48.8	47.3	66.7	76.7	32.3	35.4	53.3	60.0	58.3	60.6	73.3	75.6
Qwen3-4B + GRPO	90.1	90.0	95.8	94.6	49.2	50.6	70.0	73.3	38.5	39.4	56.7	56.7	60.7	63.5	75.6	77.8
Qwen3-4B + VCE	91.7	92.6	97.4	98.2	51.9	54.0	80.0	86.7	40.8	43.8	63.3	66.7	63.2	66.1	82.2	84.4
Qwen3-1.7B	82.9	84.2	93.0	94.4	29.2	32.1	63.3	70.0	25.4	27.5	40.0	50.0	53.8	57.2	71.1	73.3
Qwen3-1.7B + GRPO	83.6	84.8	93.4	93.8	35.2	35.6	66.7	70.0	27.1	26.7	46.7	53.3	55.6	57.8	77.8	77.8
Qwen3-1.7B + VCE	85.8	87.5	95.6	97.2	37.5	39.4	73.3	80.0	30.6	32.7	53.3	63.3	57.9	59.2	80.0	82.2

Let $\mathcal{S}_q = \{(s_{j,k}, p_j, b_{j,k}) : \text{for } p_j, \text{ exist } b_{j,k} = 1\} \cup \{(s_{j,t}^*, p_j^*, b_{j,t}^*)\}$. For each unique $\tilde{p}$ in $\mathcal{S}_q$, let $\hat{a}(\tilde{p}) = \text{mean}\{b : (s, \tilde{p}, b) \in \mathcal{S}_q\}$. We assign higher rewards to the correct solutions provided by the LLM based on the posterior generated prompt. Then for each $(s, \tilde{p}, b) \in \mathcal{S}_q$, this leads to the following reward for the solutions:

$$R_s^{\text{sol}} = \frac{1}{\hat{a}(\tilde{p}) + \gamma} (b - \hat{a}(\tilde{p})), \quad (8)$$

where γ is the parameter to avoid $a(\tilde{p}) = 0$. After collecting the rollout samples for prompts and solutions in $\mathcal{S}_q$, we optimize the two policies by reinforcement learning, using the following objectives iteratively:

$$\begin{aligned} & J_{\text{prompt}}(\theta, \{p_j\}_{j=1}^n) \\ &= \mathbb{E}_{q, \{p_j\} \sim \pi_{\theta_{\text{old}}}(\cdot|q)} \left[\frac{1}{n} \sum_{j=1}^n \min \left(r_{\theta}(p_j | q) A_{p_j}, \text{clip} \left(r_{\theta}(p_j | q), 1 - \varepsilon, 1 + \varepsilon \right) A_{p_j} \right) \right] \\ &\quad - \beta \mathbb{E}_{q, \{p_j\} \sim \pi_{\theta_{\text{old}}}(\cdot|q)} \left[D_{\text{KL}} [\pi_{\theta}(\cdot | q) \| \pi_{\theta_{\text{ref}}}(\cdot | q)] \right], \end{aligned} \quad (9)$$

where $\text{clip}(x, \varepsilon) := \min(\max(x, 1 - \varepsilon), 1 + \varepsilon)$, $r_{\theta}(p | q) = \frac{\pi_{\theta}(p|q)}{\pi_{\theta_{\text{old}}}(p|q)}$, $\pi_{\theta}(\cdot | q)$ is the prompt policy to be optimized, $\pi_{\theta_{\text{old}}}(\cdot | q)$ is the old prompt policy, $\{p_j\}_{j=1}^n$ is the rollout samples of prompts, and A_{p_j} is the normalized reward corresponding to $R_{p_j}^{\text{prompt}}$.

$$\begin{aligned} & J_{\text{solution}}(\theta, (s, \tilde{p}) \in \mathcal{S}_q) \\ &= \mathbb{E}_{q, \tilde{p} \sim \omega^*(\cdot|q)} \left[\frac{1}{|\mathcal{S}_q|} \sum_{(s, \tilde{p}, b) \in \mathcal{S}_q} \min \left(r_{\theta}(s | \tilde{p}, q) A_s, \text{clip} \left(r_{\theta}(s | \tilde{p}, q), 1 - \varepsilon, 1 + \varepsilon \right) A_s \right) \right] \\ &\quad - \beta \mathbb{E}_{q, \tilde{p} \sim \omega^*(\cdot|q)} \left[D_{\text{KL}} [\pi_{\theta}(\cdot | \tilde{p}, q) \| \pi_{\theta_{\text{ref}}}(\cdot | \tilde{p}, q)] \right], \end{aligned} \quad (10)$$

where $r_{\theta}(s | \tilde{p}, q) = \frac{\pi_{\theta}(s|\tilde{p},q)}{\pi_{\theta_{\text{old}}}(s|\tilde{p},q)}$, $\pi_{\theta}(\cdot | \tilde{p}, q)$ is the solution policy to be optimized, $\pi_{\theta_{\text{old}}}(\cdot | \tilde{p}, q)$ is the old solution policy, and A_s is the normalized reward corresponding to R_s^{sol} . Specifically, we iteratively optimize the policy of LLM's capability to generate prompts on their own with $J_{\text{prompt}}(\theta, \{p_j\}_{j=1}^n)$, and the policy of ability for reasoning and generating solutions based on the prompts (see Figure 2).

4 Experiments

4.1 Experimental Settings

Datasets For training, we use the MATH dataset (Hendrycks et al., 2021) and retain only problems at difficulty levels 3–5, yielding approximately 8K challenging tasks. For evaluation, we select four widely used benchmarks in mathematical reasoning to ensure a comprehensive assessment: MATH500 (Hendrycks et al., 2021), AIME2024 (Mathematical Association of America, 2024a), AIME2025 (Mathematical Association of America, 2025), and AMC2024 (Mathematical Association of America, 2024b).

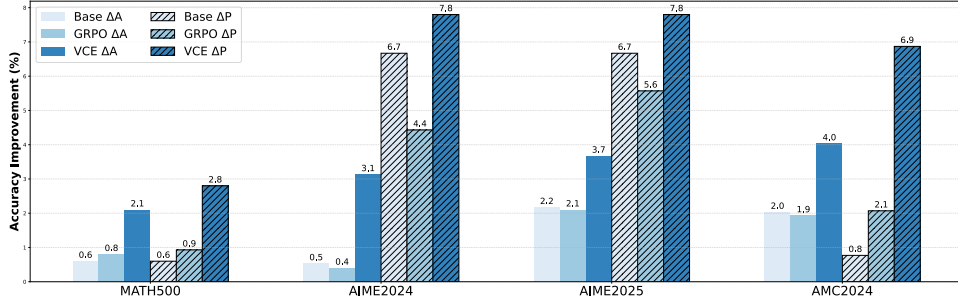


Figure 3: Performance gains over direct answering. Using direct answering without guidance prompts as the baseline, guidance prompts are generated by the base model, the GRPO-trained model, and the VCE-trained model, and then applied to the base model for final answer generation. The resulting performance improvements serve as an evaluation of prompt generation capability. Results are averaged over three model sizes (Qwen3-1.7B, 4B, and 8B) for each dataset, and reported as improvements in average accuracy (ΔA , %) and Pass@16 (ΔP , %).

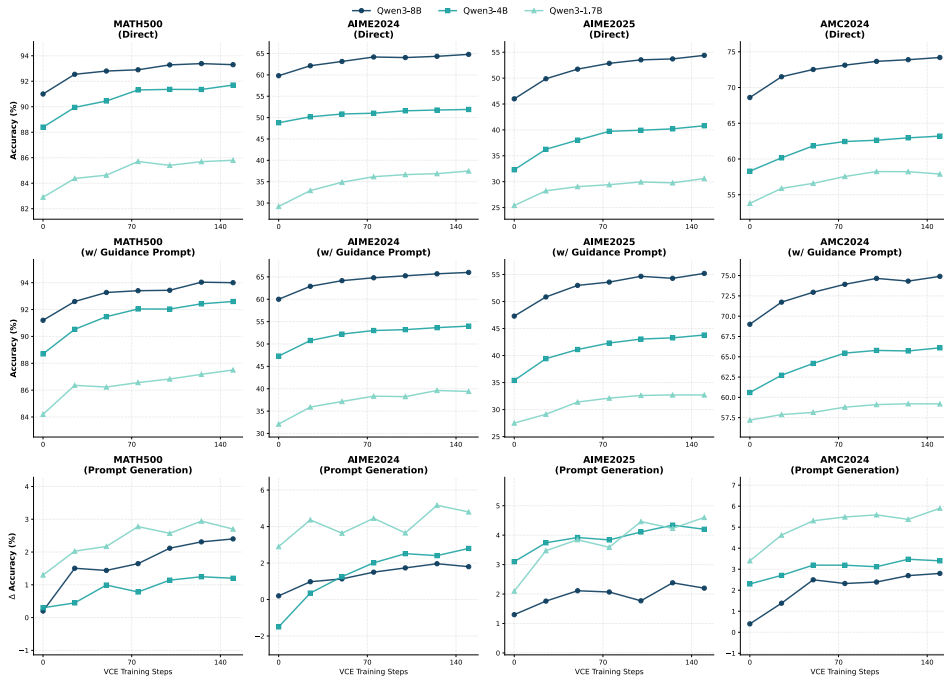


Figure 4: Learning dynamics of VCE trained on Qwen3-8B, 4B, and 1.7B across four math reasoning benchmarks. The first row shows the trajectory of average accuracy under direct answering. The second row shows the trajectory of average accuracy when answering with self-generated guidance prompts. The third row shows the trajectory of prompt generation capability, where the y-axis denotes the improvement in average accuracy obtained by applying guidance prompts generated by the trained model to its base model, relative to base model direct answering without guidance.

Models and optimization We adopt Qwen3-1.7B, 4B, and 8B (Yang et al., 2025) as our base models. During the initial prompt sampling, solution sampling under the initial prompt, and solution sampling under the refined prompt stages, we generate 8, 4, and 4 rollouts, respectively, using vLLM (Kwon et al., 2023). These generations are performed with temperature 1.0, top- p 0.95, and top- k 20. We set the learning rate to 1.5×10^{-6} and the KL coefficient to 0.01. All models are trained using 8 NVIDIA H200 GPUs.

4.2 Results

VCE markedly enhances reasoning problem-solving capability. Table 1 summarizes the solution generation performance of base models, GRPO-trained models, and VCE-trained models across four mathematical reasoning benchmarks. During training, we match the rollout token budgets for GRPO and VCE. Overall, VCE consistently achieves the strongest performance across all model sizes and datasets, yielding improvements in both average accuracy and Pass@16. The gains are particularly pronounced on harder benchmarks such as AIME2024 and AIME2025, where accurate solution generation critically depends on effective problem elicitation. For instance, on AIME2024, Qwen3-8B trained with VCE improves average accuracy from 59.8% to 64.8%, and increases Pass@16 from 80.0% to 90.0% compared to the baseline. Similar improvements are observed across all benchmarks and model scales, indicating that enhancing solution generation in isolation is often insufficient when prompt elicitation remains suboptimal. In contrast, co-evolving prompt generation with the solution policy enables the model to more reliably construct guidance that supports correct reasoning trajectories. Notably, VCE-trained models benefit more from self-generated prompts than both base models and solver-only training approaches, suggesting improved alignment between prompt generation and solution execution. These gains remain robust across model scales from 1.7B to 8B parameters, demonstrating that co-evolving prompts and solutions is an effective and scalable strategy for improving mathematical reasoning performance.

Generate more effective guidance prompts. Figure 3 evaluates prompt generation ability by measuring performance gains when guidance prompts generated by different models are applied to the base solver. Improvements are reported relative to direct answering without guidance, using average accuracy improvement (ΔA) and Pass@16 improvement (ΔP), which respectively reflect typical prompt quality and robustness under multiple generations. Across all benchmarks, VCE-generated prompts consistently yield the largest performance gains under both metrics. While prompts produced by the base model and the GRPO-trained model provide limited improvements, VCE prompts lead to notably larger ΔA and ΔP , particularly on harder datasets such as AIME2024, AIME2025, and AMC2024. For example, on AIME2024, VCE achieves a ΔP of 7.8%, compared to 6.7% for the base model and 4.4% for GRPO, suggesting that VCE improves coverage of valid solution paths across multiple generations. These results demonstrate that prompt generation quality improves substantially when it is jointly optimized with solution generation. VCE produces guidance prompts that generalize better across tasks and model sizes, validating co-evolution as an effective strategy for strengthening prompt generation ability.

Stable and coordinated improvement. Figure 4 presents the learning dynamics of VCE across three model scales and four mathematical reasoning benchmarks. As training progresses, both direct solution accuracy and accuracy with self-generated guidance prompts increase consistently, indicating simultaneous improvement of solution generation and guided reasoning performance. The performance gap between direct answering and guided answering remains stable or slightly increases, suggesting that prompt generation stays well aligned with the evolving solution policy training. In addition, prompt generation capability, measured by the gains obtained when applying generated prompts to the base model, improves steadily and converges to stable values across datasets and model sizes. Together, these trends demonstrate that prompt generation and solution generation improve in a coordinated manner under VCE, validating its design as a stable co-evolutionary framework rather than optimizing either capability in isolation.

Stronger and more robust Pass@k scaling. We further analyze model performance under Pass@k to assess both peak accuracy and the robustness of reasoning under multiple sampled solutions. As shown in Figure 5, all models benefit from larger k, reflecting increased solution diversity; however, VCE consistently exhibits steeper and more stable improvements across k than both the base model and the GRPO-trained model. Notably, the gains of VCE persist from Pass@1 to Pass@16 across all model scales and benchmarks, indicating that VCE not only improves best-case performance but also systematically increases the probability of generating correct solutions. This suggests that co-evolving prompt generation with solution policies leads to higher-quality elicitation that reliably guides reasoning trajectories, rather than relying on lucky sampling. In contrast, GRPO improves Pass@1 but

Table 2: Ablation studies on Qwen3 models with different training strategies. We compare prompt-only optimization, solution-only optimization, supervised fine-tuning (SFT), and VCE across three model scales on four mathematical reasoning benchmarks. Results report average accuracy (Avg, %) over 16 test runs and Pass@16 accuracy (P@16, %) for direct answering and for answering with self-generated guidance prompts (Avg* and P@16*). Bold indicates the best result, and underline indicates the second best.

Model	Method	MATH500				AIME2024				AIME2025				AMC2024			
		Avg	Avg*	P@16	P@16*	Avg	Avg*	P@16	P@16*	Avg	Avg*	P@16	P@16*	Avg	Avg*	P@16	P@16*
Qwen3-8B	Prompt Only	91.1	91.8	96.0	96.6	59.6	61.3	80.0	86.7	45.8	48.3	70.0	76.7	68.8	70.1	82.2	82.2
	Solution Only	91.8	92.4	<u>96.4</u>	97.2	<u>62.5</u>	63.1	83.3	86.7	51.5	<u>52.1</u>	76.7	80.0	<u>71.7</u>	72.4	84.4	88.9
	SFT	<u>92.1</u>	<u>92.6</u>	96.2	<u>97.5</u>	62.1	63.8	<u>86.7</u>	<u>90.0</u>	<u>51.9</u>	51.7	<u>80.0</u>	<u>83.3</u>	71.3	<u>72.8</u>	<u>86.7</u>	<u>91.1</u>
	VCE	93.3	94.0	98.0	98.8	64.8	66.0	90.0	96.7	54.4	55.2	83.3	86.7	74.2	74.9	91.1	93.3
Qwen3-4B	Prompt Only	88.5	89.4	94.4	95.2	49.0	48.8	66.7	80.0	32.5	36.9	53.3	63.3	58.1	61.7	73.3	77.8
	Solution Only	<u>90.4</u>	90.1	95.2	95.6	49.2	51.3	70.0	76.7	38.8	39.4	56.7	60.0	61.1	63.8	<u>77.8</u>	80.0
	SFT	90.2	<u>91.1</u>	95.8	96.5	<u>50.2</u>	51.0	73.3	<u>80.0</u>	38.5	<u>40.4</u>	60.0	63.3	61.7	<u>64.4</u>	75.6	<u>82.2</u>
	VCE	91.7	92.6	97.4	98.2	51.9	54.0	80.0	86.7	40.8	43.8	63.3	66.7	63.2	66.1	82.2	84.4
Qwen3-1.7B	Prompt Only	82.9	84.9	93.2	95.0	29.6	33.3	63.3	73.3	25.2	28.5	40.0	53.3	54.0	58.5	71.1	75.6
	Solution Only	83.8	85.0	93.4	94.6	<u>35.8</u>	35.8	66.7	70.0	27.1	<u>28.1</u>	46.7	53.3	<u>56.4</u>	57.8	75.6	77.8
	SFT	<u>84.3</u>	<u>85.6</u>	93.8	95.0	35.4	36.7	70.0	73.3	<u>27.7</u>	27.3	50.0	56.7	55.8	<u>58.3</u>	<u>77.8</u>	80.0
	VCE	85.8	87.5	95.6	97.2	37.5	39.4	73.3	80.0	30.6	32.7	53.3	63.3	57.9	59.2	80.0	82.2

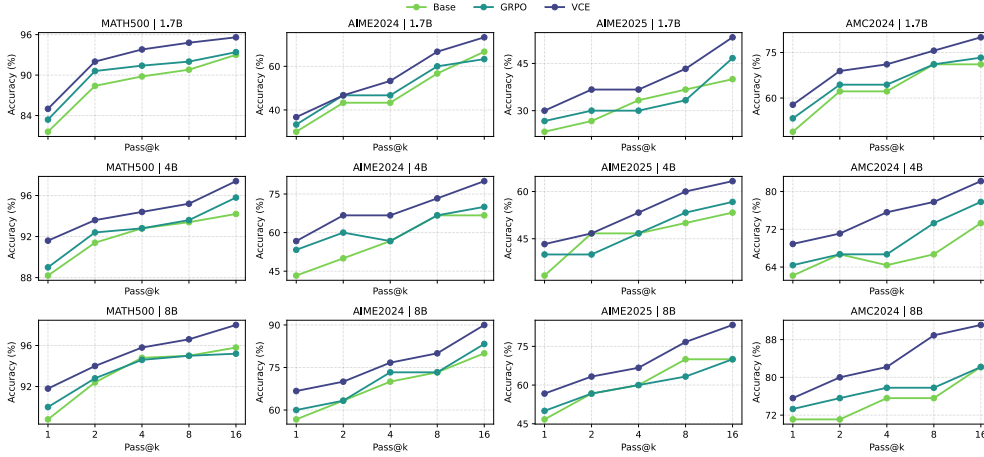


Figure 5: Pass@k comparison.

often shows weaker scaling with k , implying that solver-centric optimization alone may not sufficiently diversify or stabilize reasoning paths. Overall, the Pass@ k trends provide strong evidence that VCE enhances both the effectiveness and robustness of reasoning by aligning prompt construction and solution generation under a unified objective.

Ablation studies. Table 2 presents ablation studies on prompt-only optimization, solution-only optimization, and supervised fine-tuning (fine-tune the model on rollout samples with positive rewards). Across all benchmarks and model scales, VCE consistently achieves the best performance, highlighting the importance of jointly optimizing prompt generation and solution policies. Compared with VCE, prompt-only optimization yields weaker gains and benefits less from self-generated guidance prompts. While improved prompt elicitation can partially facilitate reasoning, the absence of concurrent updates to the solution policy limits the effectiveness of the generated guidance. As a result, prompts optimized in isolation are not fully aligned with the model’s evolving reasoning behavior. Solution-only optimization also underperforms VCE, particularly when evaluated with self-generated prompts. Although the solution policy itself is strengthened, the prompt generation process remains unchanged, leading to a mismatch between the solver’s capabilities and the guidance it later conditions on. This misalignment constrains the reliability and robustness of the resulting reasoning trajectories. Supervised fine-tuning only leverages positive examples and overlooks valuable negative samples, which limits its performance relative to VCE. These advantages hold consistently from 1.7B to 8B models, demonstrating the robustness and scalability of the proposed co-evolving approach.

5 Conclusion

In this work, we improve LLM reasoning by jointly strengthening two tightly coupled capabilities: solution generation and prompt generation for elicitation. Existing methods typically optimize them separately, limiting performance. We address this with a unified co-evolution framework that jointly improves prompting and solving through reinforcement learning, while variational inference links prompt decisions to solution quality. Across model scales and challenging math reasoning benchmarks, our approach consistently improves solution accuracy, pass@k robustness, and the quality and transferability of generated prompts. We hope this work provides a practical and theoretically grounded foundation for broader reasoning domains beyond mathematics.

References

- Paweł Batorski, Adrian Kosmala, and Paul Swoboda. Prl: Prompts from reinforcement learning. *arXiv preprint arXiv:2505.14412*, 2025.
- Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 17682–17690, 2024.
- Shicong Cen, Jincheng Mei, Katayoon Goshvadi, Hanjun Dai, Tong Yang, Sherry Yang, Dale Schuurmans, Yuejie Chi, and Bo Dai. Value-incentivized preference optimization: A unified approach to online and offline RLHF. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=SQnitDuow6>.
- Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL <https://openreview.net/forum?id=YfZ4ZPt8zd>.
- Yongchao Chen, Jacob Arkin, Yilun Hao, Yang Zhang, Nicholas Roy, and Chuchu Fan. Prompt optimization in multi-step tasks (promst): Integrating human feedback and heuristic-based sampling. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 3859–3920, 2024.
- Jiale Cheng, Xiao Liu, Kehan Zheng, Pei Ke, Hongning Wang, Yuxiao Dong, Jie Tang, and Minlie Huang. Black-box prompt optimization: Aligning large language models without model training. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3201–3219, 2024.
- Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric Xing, and Zhiting Hu. Rlprompt: Optimizing discrete text prompts with reinforcement learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 3369–3391, 2022.
- Xuan Long Do, Yiran Zhao, Hannah Brown, Yuxi Xie, James Xu Zhao, Nancy Chen, Kenji Kawaguchi, Michael Shieh, and Junxian He. Prompt optimization via adversarial in-context learning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7308–7327, 2024.
- Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: self-referential self-improvement via prompt evolution. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 13481–13544, 2024.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 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. *arXiv preprint arXiv:2103.03874*, 2021.

- Wenyang Hu, Yao Shu, Zongmin Yu, Zhaoxuan Wu, Xiaoqiang Lin, Zhongxiang Dai, See-Kiong Ng, and Bryan Kian Hsiang Low. Localized zeroth-order prompt optimization. *Advances in Neural Information Processing Systems*, 37:86309–86345, 2024.
- Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.
- Yongshuai Liu, Jiaxin Ding, and Xin Liu. Ipo: Interior-point policy optimization under constraints. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 4940–4947, 2020.
- Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36: 46534–46594, 2023.
- Mathematical Association of America. American invitational mathematics examination (aime) 2024: Aime i and aime ii. <https://maa.org/>, 2024a. Competition problems used as an evaluation dataset; original problems by MAA AMC.
- Mathematical Association of America. American mathematics competitions (amc) 2024. <https://maa.org/student-programs/amc/>, 2024b. Competition problems used as an evaluation dataset; original problems by MAA AMC.
- Mathematical Association of America. American invitational mathematics examination (aime) 2025: Aime i and aime ii. <https://maa.org/>, 2025. Competition problems used as an evaluation dataset; original problems by MAA AMC.
- Yu Meng, Mengzhou Xia, and Danqi Chen. SimPO: Simple preference optimization with a reference-free reward. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL <https://openreview.net/forum?id=3Tzcot1LKb>.
- Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with “gradient descent” and beam search. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL <https://openreview.net/forum?id=WRyhaSrThy>.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL <https://openreview.net/forum?id=HPuSIXJaa9>.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- 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.
- Dilara Soylu, Christopher Potts, and Omar Khattab. Fine-tuning and prompt optimization: Two great steps that work better together. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 10696–10710, 2024.
- Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2609–2634, 2023a.

- Xinyuan Wang, Chenxi Li, Zhen Wang, Fan Bai, Haotian Luo, Jiayou Zhang, Nebojsa Jojic, Eric Xing, and Zhiting Hu. Promptagent: Strategic planning with language models enables expert-level prompt optimization. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=22pyNMuIoa>.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023b. URL <https://openreview.net/forum?id=1PL1NIMMrw>.
- Yinjie Wang, Ling Yang, Ye Tian, Ke Shen, and Mengdi Wang. CURE: Co-evolving coders and unit testers via reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL <https://openreview.net/forum?id=wPdBe9zxNr>.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.
- Jinyu Xiang, Jiayi Zhang, Zhaoyang Yu, Fengwei Teng, Jinhao Tu, Xinbing Liang, Sirui Hong, Chenglin Wu, and Yuyu Luo. Self-supervised prompt optimization. *arXiv preprint arXiv:2502.06855*, 2025.
- 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.
- Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=Bb4VGOWELI>.
- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023a.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=WE_vluYUL-X.
- Yuxiang Zhang, Shangxi Wu, Yuqi Yang, Jiangming Shu, Jinlin Xiao, Chao Kong, and Jitao Sang. o1-coder: an o1 replication for coding. *arXiv preprint arXiv:2412.00154*, 2024.
- Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H. Chi, Quoc V Le, and Denny Zhou. Take a step back: Evoking reasoning via abstraction in large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=3bq3jsvcQ1>.
- Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scates, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*, 2023a. URL <https://openreview.net/forum?id=WZH7099tgfM>.
- Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitit, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *The Eleventh International Conference on Learning Representations*, 2023b. URL <https://openreview.net/forum?id=92gvk82DE->.

A Proof of Results from the Main Paper

A.1 Proof of equation (2)

Recall that $\pi(p \mid D, q)$ is a posterior distribution, representing the policy of the prompt generated by the LLM after self-reflection based on the observation D of the response and the question q .

Furthermore, we have the following equation:

$$\mathbb{E}_{p \sim \pi_\theta(\cdot \mid q)}[\pi_\theta(D \mid p, q)] = \pi(D \mid q). \quad (11)$$

Then, we can obtain the following variational form:

$$\log \pi(D \mid q) = \text{KL}(\omega(p \mid q) \parallel \pi(p \mid D, q)) + \mathbb{E}_{p \sim \omega(\cdot \mid q)}[\log \pi_\theta(D, p \mid q) - \log \omega(p \mid q)] \quad (12)$$

Proof. First of all, we define that the current policy of LLMs in generating prompts under a given question q is $\omega(p \mid q)$. So by the variational inference, we can obtain

$$\begin{aligned} & \mathbb{E}_{p \sim \omega(\cdot \mid q)}[\log \pi_\theta(D \mid p, q)] \\ &= \mathbb{E}_{p \sim \omega(\cdot \mid q)}\left[\log \frac{\pi_\theta(D, p \mid q)}{\omega(p \mid q)}\right] \\ &= \mathbb{E}_{p \sim \omega(\cdot \mid q)}[\log \pi_\theta(D, p \mid q) - \log \omega(p \mid q)] \\ &= \int \omega(p \mid q) \log(\pi_\theta(D, p \mid q)) dp - \int \omega(p \mid q) \log(\omega(p \mid q)) dp \\ &= \int \omega(p \mid q) \log(\pi(D \mid q) \pi(p \mid D, q)) dp - \int \omega(p \mid q) \log(\omega(p \mid q)) dp \\ &= \log \pi(D \mid q) - \int \omega(p \mid q) \log \frac{\pi(p \mid D, q)}{\omega(p \mid q)} dp \\ &= \log \pi(D \mid q) - \text{KL}(\omega(p \mid q) \parallel \pi(p \mid D, q)) \\ &\Rightarrow \log \pi(D \mid q) = \text{KL}(\omega(p \mid q) \parallel \pi(p \mid D, q)) + \mathbb{E}_{p \sim \omega(\cdot \mid q)}[\log \pi_\theta(D \mid p, q)] \\ &\Rightarrow \log \pi(D \mid q) = \text{KL}(\omega(p \mid q) \parallel \pi(p \mid D, q)) + \mathbb{E}_{p \sim \omega(\cdot \mid q)}[\log \pi_\theta(D, p \mid q) - \log \omega(p \mid q)], \end{aligned} \quad (13)$$

where the second term on the right side of the equation in (13) is denoted as $L(\theta)$. $\square$

A.2 Proof of equation (5)

Recall that we let $a_\theta(p, q) = \mathbb{E}_{D \sim \pi_\theta(\cdot \mid p, q)}[\mathbf{1}_{\text{correct}}(D)]$, which is the accuracy rate of the LLM's solution based on the prompt p and the question q , where $\mathbf{1}_{\text{correct}}(D) \in \{0, 1\}$ is the ground truth test. Through the following calculation, we can obtain

$$\begin{aligned} & \nabla_\theta a_\theta(p, q) \\ &= \mathbb{E}_{D \sim \pi_\theta(\cdot \mid p, q)}[\mathbf{1}_{\text{correct}}(D) \nabla_\theta \log \pi_\theta(D \mid p, q)]. \end{aligned} \quad (14)$$

Proof. First of all, we have the following equation

$$\nabla_\theta \log \pi_\theta(D \mid p, q) = \frac{\nabla_\theta \pi_\theta(D \mid p, q)}{\pi_\theta(D \mid p, q)} \quad (15)$$

Then, by the definition of $a_\theta(p, q)$ and substituting equation (15), we can obtain the following calculation:

$$\begin{aligned}
\nabla_{\theta} a_{\theta}(p, q) &= \nabla_{\theta} \mathbb{E}_{D \sim \pi_{\theta}(\cdot | p, q)} [\mathbf{1}_{\text{correct}}(D)] \\
&= \nabla_{\theta} \sum \pi_{\theta}(D | p, q) \mathbf{1}_{\text{correct}}(D) \\
&= \sum \nabla_{\theta} \pi_{\theta}(D | p, q) \mathbf{1}_{\text{correct}}(D) \\
&= \sum \pi_{\theta}(D | p, q) \nabla_{\theta} \log \pi_{\theta}(D | p, q) \mathbf{1}_{\text{correct}}(D) \\
&= \mathbb{E}_{D \sim \pi_{\theta}(\cdot | p, q)} [\mathbf{1}_{\text{correct}}(D) \nabla_{\theta} \log \pi_{\theta}(D | p, q)].
\end{aligned}$$

□

A.3 Proof of equation (6)

Recall that, by the calculation of equation 14, the gradient of $\mathbb{E}_{p \sim \omega^*(\cdot | q)} [\log a_{\theta}(p, q)]$ at $\theta = \theta_{\text{old}}$ is given by

$$\begin{aligned}
&\nabla_{\theta} \mathbb{E}_{p \sim \omega^*(\cdot | q)} [\log a_{\theta}(p, q)] \\
&= \mathbb{E}_{p \sim \omega^*(\cdot | q), D \sim \pi_{\theta}(\cdot | p, q)} \left[\frac{1}{a_{\theta_{\text{old}}}(p, q)} (\mathbf{1}_{\text{correct}}(D) - a_{\theta_{\text{old}}}(p, q)) \nabla_{\theta} \log \pi_{\theta}(D | p, q) \right]. \quad (16)
\end{aligned}$$

Proof. First of all, we have the following equation

$$\nabla_{\theta} \log a_{\theta}(p, q) = \frac{\nabla_{\theta} a_{\theta}(p, q)}{a_{\theta}(p, q)}. \quad (17)$$

Second, We calculate the following formula at $\theta = \theta_{\text{old}}$,

$$\begin{aligned}
&\mathbb{E}_{D \sim \pi_{\theta}(\cdot | p, q)} [\nabla_{\theta} \log \pi_{\theta}(D | p, q)]_{\theta=\theta_{\text{old}}} \\
&= \sum \pi_{\theta_{\text{old}}}(D | p, q) \frac{\nabla_{\theta} \pi_{\theta}(D | p, q)}{\pi_{\theta}(D | p, q)} \Big|_{\theta=\theta_{\text{old}}} \\
&= \sum \nabla_{\theta} \pi_{\theta}(D | p, q) \Big|_{\theta=\theta_{\text{old}}} \\
&= \nabla_{\theta} \sum \pi_{\theta}(D | p, q) \Big|_{\theta=\theta_{\text{old}}} \\
&= \nabla_{\theta} 1 \\
&= 0. \quad (18)
\end{aligned}$$

Then, by substituting equation (14), (17) and (18), we can obtain the gradient of $\mathbb{E}_{p \sim \omega^*(\cdot | q)} [\log a_{\theta}(p, q)]$ at $\theta = \theta_{\text{old}}$,

$$\begin{aligned}
&\nabla_{\theta} \mathbb{E}_{p \sim \omega^*(\cdot | q)} [\log a_{\theta}(p, q)] \\
&= \mathbb{E}_{p \sim \omega^*(\cdot | q)} \left[\frac{1}{a_{\theta_{\text{old}}}(p, q)} \nabla_{\theta} a_{\theta}(p, q) \right] \\
&= \mathbb{E}_{p \sim \omega^*(\cdot | q), D \sim \pi_{\theta}(\cdot | p, q)} \left[\frac{1}{a_{\theta_{\text{old}}}(p, q)} (\mathbf{1}_{\text{correct}}(D) - a_{\theta_{\text{old}}}(p, q)) \nabla_{\theta} \log \pi_{\theta}(D | p, q) \right].
\end{aligned}$$

□

B Details of Algorithm

We provide the detailed VCE optimization pipeline as follows. We train on the MATH training split (levels 3–5) and use Qwen3-1.7B/4B/8B. We set $n = 8$, $m = 4$, $l = 4$, $\eta = 1.5 \times 10^{-6}$ and $\beta = 0.01$.

Algorithm 1 VCE: Variational Co-Evolution of Prompt Generation and Reasoning Solution

Require: Dataset $D = \{q_1, \dots, q_N\}$; A policy π_θ parameterized by θ ; iterations M ; prompts per question n ; initial solutions per prompt m ; new solutions per reflected prompt l ; learning rate η ; KL coefficient β ; reward parameter γ .

```

1: Initialize policy parameters  $\theta$ .
2: for  $t = 1$  to  $M$  do
3:   Collect rollout samples
4:   for each question  $q \in D$  do
5:     Generate  $n$  initial prompts  $\{p_j\}_{j=1}^n$  using policy  $\pi_\phi$  with question  $q$ .
6:     for  $j = 1$  to  $n$  do
7:       for  $k = 1$  to  $m$  do
8:         Generate solution  $s_{j,k}$  using policy  $\pi_\theta$  with  $(q, p_j)$ .
9:          $b_{j,k} = \mathbf{1}_{\text{correct}(s_{j,k}) \in \{0, 1\}}$ .
10:      end for
11:    end for
12:    Self-reflection for wrong solutions
13:    for each prompt  $p_j$  such that  $\exists k, b_{j,k} = 0$  do
14:      Obtain a refined prompt  $p_j^*$  by letting the LLM self-reflect on all incorrect
      solutions.
15:      for  $t' = 1$  to  $l$  do
16:        Generate new solution  $s_{j,t'}^*$  using policy  $\pi_\theta$  with  $(q, p_j^*)$ .
17:         $b_{j,t'}^* = \mathbf{1}_{\text{correct}(s_{j,t'}^*)}$ .
18:      end for
19:    end for
20:  end for
21:  Rewards for generating prompts
22:  for  $j = 1$  to  $n$  do
23:     $R_{p_j}^{\text{prompt}} = \frac{1}{m} \sum_{k=1}^m b_{j,k}$ 
24:  end for
25:  Rewards for reasoning solutions (augmented set)
26:   $\mathcal{S}_q = \{(s_{j,k}, p_j, b_{j,k}) : \text{for } p_j, \text{ exist } b_{j,k} = 1\} \cup \{(s_{j,t'}^*, p_j^*, b_{j,t'}^*)\}$ .
27:  for each unique  $\tilde{p}$  in  $\mathcal{S}_q$  do
28:     $\hat{a}(\tilde{p}) = \text{mean}\{b : (s, \tilde{p}, b) \in \mathcal{S}_q\}$ 
29:  end for
30:  for each  $(s, \tilde{p}, b) \in \mathcal{S}_q$  do
31:     $R_s^{\text{sol}} = \frac{1}{\hat{a}(\tilde{p})} (b - \hat{a}(\tilde{p}))$ . To avoid  $a(\tilde{p}) = 0$ , we can let  $R_s^{\text{sol}} = \frac{1}{\hat{a}(\tilde{p}) + \gamma} (b - \hat{a}(\tilde{p}))$ 
32:  end for
33:  Optimize the policy  $\pi_\theta$ 
34:  Compute advantages  $A_p = \text{Normalize}_{\text{per-}q}(\{R_{p_j}^{\text{prompt}}\})$ .
35:   $\theta \leftarrow \theta - \eta \nabla_\theta J_{\text{prompt}}(\theta, \{p_j\}_{j=1}^n)$ .
36:  Compute advantages  $A_s = \text{Normalize}_{\text{per-}q}(\{R_s^{\text{sol}}\})$ .
37:   $\theta \leftarrow \theta - \eta \nabla_\theta J_{\text{solution}}(\theta, (s, \tilde{p}) \in \mathcal{S}_q)$ .
38: end for
Ensure: Trained policy  $\pi_\theta$ .

```

C Details of Experiment

C.1 Prompt Design

The prompt for generating guidance prompts:

```
prompt = """<|im_start|>system
You are a helpful assistant that improves problem-solving.
<|im_end|>
<|im_start|>user
Task: Generate a guiding prompt for the following problem.
```

Requirements:

The guiding prompt must not provide answers or explicit solution steps, and should remain cognitively activating. It may encourage the solver to clarify assumptions, constraints, and unknowns. The guidance may briefly mention common pitfalls or offer subtle directional cues, but only insofar as they preserve room for reasoning. All content should be concise and neutral in tone.

Problem:

{{problem}}

The guiding prompt MUST be enclosed within <prompt> </prompt> tags, strictly following this structure:

```
<prompt>your guiding prompt in English here</prompt>
<|im_end|>
<|im_start|>assistant
"""
```

The prompt for generating solutions under guidance prompts:

```
prompt = '''<|im_start|>system
You are a helpful assistant that help user solve problems.<|im_end|>
<|im_start|>user
Problem: {problem}
Guidance: {guidance}
Please use the guidance above as part of your reasoning when solving the problem.
Provide your final answer in a boxed format at the end: \\boxed{{}}.<|im_end|>
<|im_start|>assistant
'''
```

The prompt for generating reflected prompts:

```
prompt = """<|im_start|>system
You are a self-reflective assistant.<|im_end|>
<|im_start|>user
You previously answered a question incorrectly multiple times.
The mistakes are provided as an array of objects, each containing
an "incorrect_answer" and its corresponding "full_incorrect_response".

Your task:
1. Identify the common root cause(s) across all mistakes (e.g., recurring
calculation errors, repeated logic flaws, or misunderstandings of the problem).
2. Based on that analysis, generate a guiding prompt to help
prevent similar errors in the future.
```

Requirements:

The guiding prompt must not provide answers or explicit solution steps, and should remain cognitively activating. It may encourage the solver to clarify assumptions, constraints, and unknowns. The guidance may briefly mention common pitfalls or offer subtle directional cues, but only insofar as they preserve room for

reasoning. All content should be concise and neutral in tone.

This is the Problem:

```
{{ problem }}
```

Your mistakes (array of objects):

```
{% for m in mistakes %}
```

```
- Incorrect answer: {{ m.incorrect_answer }}
```

```
Full response: {{ m.full_incorrect_response }}
```

```
{% endfor %}
```

Correct answer:

```
{{ correct_answer }} (for analysis only, do not reference directly)
```

The new guiding prompt MUST be enclosed within `<prompt>` `</prompt>` tags, strictly following this structure:

```
<prompt> your guiding prompt in English here </prompt>
```

```
<|im_end|>
```

```
<|im_start|>assistant
```

```
"""
```

C.2 Details of Evaluation

Inference for all base models (Qwen3-1.7B, 4B, and 8B) and models trained on top of them was performed using vLLM v0.8.5 [Kwon et al. \(2023\)](#). The maximum sequence length was set to 10,000 tokens, with a temperature of 0.6. We used bfloat16 precision, and GPU memory utilization was limited to 0.85. Sampling was performed with $\text{top-}p = 0.95$ and $\text{top-}k = 20$. These settings were consistently applied across all datasets.

D Generated Guidance Prompt Examples

Examples for Qwen3-1.7B + VCE

Problem 1

Find the number of ways to place a digit in each cell of a 2×3 grid so that the sum of the two numbers formed by reading left to right is 999, and the sum of the three numbers formed by reading top to bottom is 99.

The grid below is an example of such an arrangement because $8 + 991 = 999$ and $9 + 9 + 81 = 99$.

0	0	8
9	9	1

Guidance Prompt 1

Consider a 2×3 grid in which each cell holds a single digit. The sums of the numbers formed by reading the rows and the columns impose constraints on the digits. Reflect on how to represent each digit, the relationships between rows and columns, and how these sums might interact. Clarify any assumptions about digit values and consider potential overlaps or patterns. Pay attention to the structure of the problem and the interplay of constraints, without committing to specific calculations.

Solution

To solve this problem, we start by representing the digits as variables and translating the row and column sum requirements into a system of linear equations.

Step 1: Define Variables Let the digits in the 2×3 grid be represented by variables:

a	b	c
d	e	f

where $a, b, c, d, e, f \in \{0, 1, 2, \dots, 9\}$.

Step 2: Translate Row Constraints into Equations The sum of the two numbers formed by the rows is 999:

$$(100a + 10b + c) + (100d + 10e + f) = 999$$

Grouping by powers of 10:

$$100(a + d) + 10(b + e) + (c + f) = 999$$

Since $a + d, b + e, c + f$ are sums of two digits, their maximum value is 18. For the sum to be 999 without carries propagating beyond adjacent places, it must hold that:

$$a + d = 9, \quad b + e = 9, \quad c + f = 9$$

Step 3: Translate Column Constraints and Solve The sum of the three numbers formed by reading top to bottom (the columns) is 99:

$$(10a + d) + (10b + e) + (10c + f) = 99$$

Rearranging to group a, b, c and d, e, f :

$$10(a + b + c) + (d + e + f) = 99$$

Using the relations from Step 2 ($d = 9 - a, e = 9 - b, f = 9 - c$):

$$10(a + b + c) + (9 - a + 9 - b + 9 - c) = 99$$

$$9(a + b + c) + 27 = 99 \implies 9(a + b + c) = 72 \implies a + b + c = 8$$

Step 4: Calculate the Number of Ways We need the number of non-negative integer solutions to $a + b + c = 8$. Since the sum is 8, the digit constraint ($a, b, c \leq 9$) is satisfied. Using "Stars and Bars":

$$\text{Ways} = \binom{n + k - 1}{k - 1} = \binom{8 + 3 - 1}{3 - 1} = \binom{10}{2} = 45$$

Each unique triple (a, b, c) uniquely determines (d, e, f) .

Final Answer

45

Problem 2

Let $\omega \neq 1$ be a 13th root of unity. Find the remainder when

$$\prod_{k=0}^{12} (2 - 2\omega^k + \omega^{2k})$$

is divided by 1000.

Guidance Prompt 2

Explore the structure as a quadratic in ω^k , and examine symmetries or patterns among the terms. Consider relationships with cyclotomic polynomials and polynomial evaluations at roots of unity. Focus on properties and constraints, and think about alternative perspectives, such as symmetry or algebraic simplification, to reason about the outcome.

Solution

Evaluating this product efficiently involves identifying the quadratic form of the terms and applying the algebraic properties of cyclotomic polynomials at the roots of unity.

Step 1: Define the Polynomial

Let $P(x) = x^2 - 2x + 2$. The expression to evaluate is $\Pi = \prod_{k=0}^{12} P(\omega^k)$. We find the roots of $P(x)$ using the quadratic formula:

$$x = \frac{2 \pm \sqrt{4 - 8}}{2} = 1 \pm i$$

Let $\alpha = 1 + i$ and $\beta = 1 - i$, so $P(x) = (x - \alpha)(x - \beta)$.

Step 2: Utilize Roots of Unity Identities

The values ω^k for $k = 0, 1, \dots, 12$ are the roots of the polynomial $x^{13} - 1$. Thus, for any z :

$$\prod_{k=0}^{12} (z - \omega^k) = z^{13} - 1$$

We can express the product Π as:

$$\Pi = \prod_{k=0}^{12} (\omega^k - \alpha)(\omega^k - \beta) = \left[\prod_{k=0}^{12} (\omega^k - \alpha) \right] \left[\prod_{k=0}^{12} (\omega^k - \beta) \right]$$

Since there are 13 terms (an odd number), $\prod (\omega^k - z) = -(z^{13} - 1)$:

$$\Pi = [-(\alpha^{13} - 1)] \cdot [-(\beta^{13} - 1)] = (\alpha^{13} - 1)(\beta^{13} - 1)$$

Step 3: Evaluate Powers of α and β

Converting $\alpha = 1 + i$ to polar form, $\alpha = \sqrt{2}e^{i\pi/4}$:

$$\alpha^{13} = (\sqrt{2})^{13} e^{i13\pi/4} = 64\sqrt{2} \left(\cos \frac{13\pi}{4} + i \sin \frac{13\pi}{4} \right)$$

Using the fact that $\frac{13\pi}{4} = 3\pi + \frac{\pi}{4}$:

$$\alpha^{13} = 64\sqrt{2} \left(-\frac{1}{\sqrt{2}} - \frac{i}{\sqrt{2}} \right) = -64 - 64i$$

Since β is the complex conjugate of α , $\beta^{13} = -64 + 64i$.

Step 4: Final Computation

Substitute these results back into the simplified product:

$$\Pi = (-64 - 64i - 1)(-64 + 64i - 1) = (-65 - 64i)(-65 + 64i)$$

$$\Pi = (-65)^2 - (64i)^2 = 4225 + 4096 = 8321$$

Dividing 8321 by 1000 leaves a remainder of 321.

Final Answer

321

Problem 3

Every morning, Aya does a 9 kilometer walk, and then finishes at the coffee shop. One day, she walks at s kilometers per hour, and the walk takes 4 hours, including t minutes at the coffee shop. Another morning, she walks at $s + 2$ kilometers per hour, and the walk takes 2 hours and 24 minutes, including t minutes at the coffee shop. This morning, if she walks at $s + \frac{1}{2}$ kilometers per hour, how many minutes will the walk take, including the t minutes at the coffee shop?

Guidance Prompt 3

Consider the two components that make up the total duration in each scenario: the time spent walking and the fixed time t . How can you use the relationship between speed, distance, and time to link the first two scenarios and isolate the unknown variables? As you formulate your equations, pay close attention to the consistency of units between hours and minutes.

Solution

By breaking down the total duration into walking time and stationary time, we can model Aya's schedule using a system of equations based on her varying speeds and the constant distance of her walk.

Step 1: Define Variables

Let:

- s be Aya's walking speed in km/h.
- t be the time in minutes spent at the coffee shop.
- $D = 9$ km be the walking distance.

Step 2: Translate Scenarios into Equations

For the first scenario, with total time 4 hours:

$$\frac{9}{s} + \frac{t}{60} = 4 \quad \dots (1)$$

For the second scenario, with total time 2 hours 24 minutes ($2 + 24/60 = 2.4$ hours):

$$\frac{9}{s+2} + \frac{t}{60} = 2.4 \quad \dots (2)$$

Step 3: Solve the System of Equations

Subtract equation (2) from equation (1) to eliminate the coffee shop time $\frac{t}{60}$:

$$\frac{9}{s} - \frac{9}{s+2} = 4 - 2.4 = 1.6$$

Multiply both sides by $s(s+2)$ and simplify:

$$9(s+2) - 9s = 1.6s(s+2) \implies 18 = 1.6(s^2 + 2s)$$

$$s^2 + 2s = \frac{18}{1.6} = 11.25 \implies s^2 + 2s - 11.25 = 0$$

Solving via the quadratic formula gives the positive root $s = 2.5$ km/h. Now, find t :

$$\frac{t}{60} = 4 - \frac{9}{2.5} = 4 - 3.6 = 0.4 \text{ hours} \implies t = 24 \text{ minutes.}$$

Step 4: Solve the Third Scenario

For the third scenario, the speed is $s + 0.5 = 3.0$ km/h. The walking time is:

$$\frac{9}{3} = 3 \text{ hours} = 180 \text{ minutes.}$$

Adding the $t = 24$ minutes spent at the coffee shop:

$$180 + 24 = 204 \text{ minutes.}$$

Final Answer

204

Examples for Qwen3-4B + VCE**Problem 1**

Let $\mathcal{B}$ be the set of rectangular boxes with surface area 54 and volume 23. Let r be the radius of the smallest sphere that can contain each of the rectangular boxes that are elements of $\mathcal{B}$. The value of r^2 can be written as $\frac{p}{q}$, where p and q are relatively prime positive integers. Find $p + q$.

Guidance Prompt 1

Determine the geometric property that defines the smallest sphere capable of enclosing a rectangular box. While the surface area and volume are fixed, the box's diagonal is not constant; consider how the requirement to contain every box in the set dictates the necessary radius. Investigating the conditions under which the three side lengths exist as real, positive values will help identify the limiting case that determines the value of r .

Solution

To solve the problem, we are given a set $\mathcal{B}$ of rectangular boxes with fixed surface area 54 and volume 23. We need to find the radius r of the smallest sphere that can contain **every** box in $\mathcal{B}$. This means r must be large enough to accommodate the box with the largest diagonal in the set.

Step 1: Understanding the Geometry

For a rectangular box with dimensions a, b, c , the surface area and volume are:

$$\text{Surface Area} = 2(ab + bc + ac) = 54 \Rightarrow ab + bc + ac = 27$$

$$\text{Volume} = abc = 23$$

The space diagonal of the box is $d = \sqrt{a^2 + b^2 + c^2}$. The radius r of the smallest sphere that can contain the box is $r = d/2$, so:

$$r^2 = \frac{a^2 + b^2 + c^2}{4}$$

To find the radius that contains all boxes, we must maximize $a^2 + b^2 + c^2$ subject to the constraints.

Step 2: Expressing $a^2 + b^2 + c^2$ in Terms of S

We use the algebraic identity:

$$a^2 + b^2 + c^2 = (a + b + c)^2 - 2(ab + bc + ac)$$

Substituting constraint (1):

$$a^2 + b^2 + c^2 = (a + b + c)^2 - 54$$

Maximizing the diagonal is equivalent to maximizing the sum $S = a + b + c$.

Step 3: Finding the Maximum of $S = a + b + c$

The dimensions a, b, c are the roots of the cubic polynomial:

$$f(t) = t^3 - St^2 + 27t - 23 = 0$$

For a box to exist, the dimensions must be real and positive. The maximum value of S occurs at the boundary where the cubic has a repeated root (i.e., when two dimensions of the box are equal). Let $a = b = x$. From (2), $c = 23/x^2$. Substituting into (1):

$$x^2 + 2x \left(\frac{23}{x^2} \right) = 27 \Rightarrow x^3 - 27x + 46 = 0$$

By inspection, $x = 2$ is a root ($8 - 54 + 46 = 0$). For $x = 2$, we find $c = 23/4 = 5.75$. The sum S is:

$$S = 2 + 2 + \frac{23}{4} = \frac{39}{4}$$

Thus, the maximum value of $a^2 + b^2 + c^2$ is:

$$a^2 + b^2 + c^2 = \left(\frac{39}{4} \right)^2 - 54 = \frac{1521}{16} - \frac{864}{16} = \frac{657}{16}$$

Step 4: Computing r^2

The radius squared is:

$$r^2 = \frac{a^2 + b^2 + c^2}{4} = \frac{657}{16 \cdot 4} = \frac{657}{64}$$

Since $\gcd(657, 64) = 1$, we have $p = 657$ and $q = 64$.

$$p + q = 657 + 64 = \boxed{721}$$

Problem 2

Let $ABCD$ be a tetrahedron such that $AB = CD = \sqrt{41}$, $AC = BD = \sqrt{80}$, and $BC = AD = \sqrt{89}$. There exists a point I inside the tetrahedron such that the distances from I to each of the faces of the tetrahedron are all equal. This distance can be written in the form $\frac{m\sqrt{n}}{p}$, when m, n , and p are positive integers, m and p are relatively prime, and n is not divisible by the square of any prime. Find $m + n + p$.

Guidance Prompt 2

Analyze the lengths of the edges to determine the specific type of tetrahedron formed and the relationship between its four faces. Consider how the total volume of any tetrahedron can be partitioned into smaller components defined by the distance from an internal point to its faces. For calculating the volume, it may be useful to explore how this particular symmetric shape can be embedded within a simpler orthogonal solid. Ensure that the calculated face area and volume are used to isolate the common distance while maintaining the required radical form.

Solution

To solve the problem, we identify that the tetrahedron $ABCD$ has opposite edges of equal length. This defines an **isosceles tetrahedron**, which is **equifacial** (all four faces are congruent triangles). The point I is the incenter, and the common distance to the faces is the inradius r .

Step 1: Embedding in a Rectangular Box

An isosceles tetrahedron can be embedded in a rectangular box with dimensions x, y, z such that the edges of the tetrahedron are the diagonals of the faces of the box. The edge lengths are related to x, y, z by:

$$x^2 + y^2 = AB^2 = 41$$

$$x^2 + z^2 = AC^2 = 80$$

$$y^2 + z^2 = BC^2 = 89$$

Adding these equations gives $2(x^2 + y^2 + z^2) = 41 + 80 + 89 = 210$, so $x^2 + y^2 + z^2 = 105$. Solving for each variable:

$$z^2 = 105 - 41 = 64 \Rightarrow z = 8$$

$$y^2 = 105 - 80 = 25 \Rightarrow y = 5$$

$$x^2 = 105 - 89 = 16 \Rightarrow x = 4$$

Step 2: Computing the Volume

The volume V of an isosceles tetrahedron embedded in a box of dimensions x, y, z is $1/3$ of the volume of the box (derived by subtracting four corner tetrahedra, each with volume $\frac{1}{6}xyz$):

$$V = \frac{1}{3}xyz = \frac{1}{3}(4)(5)(8) = \frac{160}{3}$$

Step 3: Computing the Area of One Face

Each face is a triangle with sides $\sqrt{41}, \sqrt{80}, \sqrt{89}$. Using the property of an isosceles tetrahedron, the area K of a face can be calculated using the box dimensions:

$$K = \frac{1}{2} \sqrt{x^2y^2 + y^2z^2 + z^2x^2}$$

$$K = \frac{1}{2} \sqrt{(4^2 \cdot 5^2) + (5^2 \cdot 8^2) + (8^2 \cdot 4^2)} = \frac{1}{2} \sqrt{400 + 1600 + 1024}$$

$$K = \frac{1}{2} \sqrt{3024} = \frac{1}{2} \cdot 12\sqrt{21} = 6\sqrt{21}$$

Step 4: Finding the Inradius r

The volume of the tetrahedron can be partitioned into four smaller tetrahedra, each with a face of the original tetrahedron as its base and point I as its vertex. Thus:

$$V = \frac{1}{3}(A_1 + A_2 + A_3 + A_4)r = \frac{1}{3}(4K)r$$

Substituting the values for V and K :

$$\frac{160}{3} = \frac{1}{3}(4 \cdot 6\sqrt{21})r \Rightarrow 160 = 24\sqrt{21}r$$

$$r = \frac{160}{24\sqrt{21}} = \frac{20}{3\sqrt{21}} = \frac{20\sqrt{21}}{63}$$

Step 5: Final Calculation

The expression is in the form $\frac{m\sqrt{n}}{p}$ with $m = 20$, $n = 21$, and $p = 63$. m and p are relatively prime ($\gcd(20, 63) = 1$). $n = 21$ is square-free. Therefore, $m + n + p = 20 + 21 + 63 = \boxed{104}$.

Problem 3

Let p be the least prime number for which there exists a positive integer n such that $n^4 + 1$ is divisible by p^2 . Find the least positive integer m such that $m^4 + 1$ is divisible by p^2 .

Guidance Prompt 3

Analyze the implications of the congruence $x^4 \equiv -1 \pmod{p}$ for the multiplicative order of x and the resulting constraints on p . When extending this to the modulus p^2 , consider the total number of distinct solutions such a polynomial equation can have. A key challenge lies in ensuring that the found value is indeed the absolute minimum; reflect on the symmetry or relationships between different roots within the full range of the modulus.

Solution

To solve the problem, we first determine the least prime p such that $n^4 + 1 \equiv 0 \pmod{p^2}$ is solvable, and then find the smallest positive integer m satisfying the congruence for that specific p .

Step 1: Constraints on the Prime p

If $n^4 + 1 \equiv 0 \pmod{p^2}$, then $n^4 \equiv -1 \pmod{p}$. Squaring both sides gives $n^8 \equiv 1 \pmod{p}$. This implies the multiplicative order of n modulo p is exactly 8. According to Fermat's Little Theorem, the order of an element must divide $p - 1$, so $8 \mid (p - 1)$, or $p \equiv 1 \pmod{8}$.

Testing the smallest primes:

- $p = 2$: $n^4 + 1$ is never divisible by $2^2 = 4$ (if n is odd, $n^4 + 1 \equiv 2 \pmod{4}$).
- $p = 3, 5, 7, 11, 13$: These do not satisfy $p \equiv 1 \pmod{8}$.
- $p = 17$: This satisfies $17 \equiv 1 \pmod{8}$.

Thus, the least prime is $p = 17$.

Step 2: Solving the Congruence Modulo 17

We need to find m such that $m^4 \equiv -1 \pmod{289}$. We first find solutions to $m^4 \equiv -1 \equiv 16 \pmod{17}$. The roots are:

$$m^2 \equiv 4 \pmod{17} \Rightarrow m \equiv 2, 15 \pmod{17}$$

$$m^2 \equiv -4 \equiv 13 \pmod{17} \Rightarrow m^2 \equiv 64 \pmod{17} \Rightarrow m \equiv 8, 9 \pmod{17}$$

The base solutions are $m_0 \in \{2, 8, 9, 15\}$.

Step 3: Lifting Solutions to Modulo 289

We use Hensel's Lemma to lift these solutions. Let $f(x) = x^4 + 1$. We look for $m = m_0 + 17k$ such that $f(m_0 + 17k) \equiv 0 \pmod{289}$. The linear lift is given by $f(m_0) + 17kf'(m_0) \equiv 0 \pmod{289}$, which simplifies to:

$$k \equiv -\frac{f(m_0)}{17} \cdot [f'(m_0)]^{-1} \pmod{17}$$

Lifting $m_0 = 8$: $f(8) = 8^4 + 1 = 4097$. Note $4097 = 17 \times 241$. $f(8)/17 = 241 \equiv 3 \pmod{17}$. $f'(8) = 4(8^3) = 4(512) \equiv 4(2) \equiv 8 \pmod{17}$. The modular inverse of 8 $\pmod{17}$ is $15 \equiv -2$. $k \equiv -3 \cdot (-2) = 6 \pmod{17}$. So, $m = 8 + 17(6) = 110$.

Step 4: Finding the Absolute Minimum

Since the polynomial $x^4 + 1$ is even, if m is a solution, $289 - m$ is also a solution. Furthermore, the four solutions modulo p^2 are symmetric. Lifting all four roots:

- $m_0 = 8 \Rightarrow m = 110$
- $m_0 = 15 \Rightarrow m = 15 + 17(7) = 134$
- $m_0 = 2 \Rightarrow m = 2 + 17(9) = 155$
- $m_0 = 9 \Rightarrow m = 9 + 17(10) = 179$

(Note that $110 + 179 = 289$ and $134 + 155 = 289$). The least positive integer is $m = 110$.

Final Answer

110

Examples for Qwen3-8B + VCE

Problem 1

A strictly increasing sequence of positive integers $a_1, a_2, a_3, \dots$ has the property that for every positive integer k , the subsequence $a_{2k-1}, a_{2k}, a_{2k+1}$ is geometric and the subsequence $a_{2k}, a_{2k+1}, a_{2k+2}$ is arithmetic. Suppose that $a_{13} = 2016$. Find a_1 .

Guidance Prompt 1

Examine how the odd- and even-indexed terms are related through the geometric and arithmetic conditions. Focus on the ratios and differences that link consecutive triples rather than trying to guess a general formula. Be careful not to treat the sequence as a standard arithmetic or geometric progression; the alternating structure is key.

Solution

We can approach this sequence problem by identifying the underlying recurrence pattern created by the alternating geometric and arithmetic constraints on consecutive terms.

Step 1: Define Initial Terms and Ratio

Let $a_1 = a$. Since a_1, a_2, a_3 is a geometric progression (GP), let the common ratio be r . Because the sequence consists of integers and is strictly increasing, we have $r > 1$. Thus:

$$a_1 = a, \quad a_2 = ar, \quad a_3 = ar^2$$

Step 2: Establish the Recurrence Pattern

Using the arithmetic progression (AP) property for a_2, a_3, a_4 :

$$a_4 = 2a_3 - a_2 = 2ar^2 - ar = ar(2r - 1)$$

Now, using the GP property for a_3, a_4, a_5 :

$$a_5 = \frac{a_4^2}{a_3} = \frac{[ar(2r - 1)]^2}{ar^2} = a(2r - 1)^2$$

Continuing to a_6 (AP property for a_4, a_5, a_6):

$$a_6 = 2a_5 - a_4 = 2a(2r - 1)^2 - ar(2r - 1) = a(2r - 1)[2(2r - 1) - r] = a(2r - 1)(3r - 2)$$

Now, for a_7 (GP property for a_5, a_6, a_7):

$$a_7 = \frac{a_6^2}{a_5} = \frac{[a(2r - 1)(3r - 2)]^2}{a(2r - 1)^2} = a(3r - 2)^2$$

Step 3: Generalize the Formula for Odd Terms

Following this logic, the odd-indexed terms follow the pattern:

$$a_{2n+1} = a(nr - (n - 1))^2$$

For a_{13} , we set $2n + 1 = 13$, which gives $n = 6$:

$$a_{13} = a(6r - 5)^2 = 2016$$

Step 4: Solve for the Initial Term

Since the sequence consists of integers, r must be a rational number, say $r = \frac{p}{q}$ in simplest form. To ensure $a_2 = ar$ and $a_3 = ar^2$ are integers, a must be a multiple of q^2 . Let $a = mq^2$ for some integer m . Substituting these into the equation:

$$mq^2 \left(6 \cdot \frac{p}{q} - 5 \right)^2 = m(6p - 5q)^2 = 2016$$

The prime factorization of 2016 is $2^5 \cdot 3^2 \cdot 7 = 14 \cdot 12^2$. We look for perfect square factors of 2016 to match $(6p - 5q)^2$:

- If $(6p - 5q)^2 = 12^2 = 144$, then $m = 14$. Thus $6p - 5q = 12$.
- Rearranging for q : $5q = 6p - 12 = 6(p - 2)$. For q to be an integer, q must be a multiple of 5. However, since $5q$ is a multiple of 6, q must be a multiple of 6. Let $q = 6$.
- Then $30 = 6p - 12 \implies 6p = 42 \implies p = 7$.

Since $\gcd(7, 6) = 1$ and $p > q$, this is a valid ratio. We calculate a_1 :

$$a_1 = a = mq^2 = 14 \cdot 6^2 = 14 \cdot 36 = 504$$

Other square factors of 2016 (like $6^2, 4^2, 3^2, 2^2, 1^2$) do not yield integer values for p and q that satisfy $p > q$ and the integer constraints of the sequence.

Final Answer

504

Problem 2

Define $f(x) = ||x| - \frac{1}{2}|$ and $g(x) = ||x| - \frac{1}{4}|$. Find the number of intersections of the graphs of

$$y = 4g(f(\sin(2\pi x))) \quad \text{and} \quad x = 4g(f(\cos(3\pi y)))$$

Guidance Prompt 2

Pay attention to the repeating "V-shaped" patterns in the inner functions and how they constrain the outputs. Think about how the sine and cosine oscillations align with these patterns along each axis. Focus on the global structure of the graphs rather than individual points, as this structure often reveals the intersection count efficiently.

Solution

We can determine the number of intersections by analyzing the periodicity and the monotonic segments of the composite functions within the unit square $[0, 1] \times [0, 1]$.

Step 1: Analyze the Core Composite Function

Let $h(u) = 4g(f(u))$. By substituting the definitions of f and g , we observe how $h(u)$ behaves for $u \in [-1, 1]$:

- $h(u) = 1$ when $u \in \{0, \pm 1/2, \pm 1\}$.
- $h(u) = 0$ when $u \in \{\pm 1/4, \pm 3/4\}$.

As u moves from 0 to 1, $h(u)$ completes two full "V-cycles" ($1 \rightarrow 0 \rightarrow 1 \rightarrow 0 \rightarrow 1$), consisting of 4 monotonic segments. Because $h(u)$ is an even function, the same occurs from 0 to -1 . Thus, for one full traverse of u from -1 to 1, $h(u)$ consists of 8 monotonic segments.

Step 2: Count Segments for the First Graph

The first graph is $y = h(\sin(2\pi x))$. Over the interval $x \in [0, 1]$, the argument $\sin(2\pi x)$ completes one full oscillation: $0 \rightarrow 1 \rightarrow 0 \rightarrow -1 \rightarrow 0$.

- $0 \rightarrow 1$: 4 segments
- $1 \rightarrow 0$: 4 segments
- $0 \rightarrow -1$: 4 segments
- $-1 \rightarrow 0$: 4 segments

This results in $N = 16$ monotonic segments for $y(x)$.

Step 3: Count Segments for the Second Graph

The second graph is $x = h(\cos(3\pi y))$. Over the interval $y \in [0, 1]$, the argument $\cos(3\pi y)$ completes 1.5 cycles: $1 \rightarrow -1 \rightarrow 1 \rightarrow -1$. Each full traverse from 1 to -1 (or vice versa) covers the 8 monotonic segments of $h(u)$. With three such traverses: $M = 3 \times 8 = 24$ monotonic segments for $x(y)$.

Step 4: Evaluate Intersections and Boundary Conditions

In a typical grid formed by N vertical segments and M horizontal segments spanning the unit square, there are $N \times M$ intersections. For these specific trigonometric functions, we must also check the endpoints:

- At $(x, y) = (1, 1)$: $y = h(\sin(2\pi)) = h(0) = 1$ and $x = h(\cos(3\pi)) = h(-1) = 1$. The point $(1, 1)$ is a shared intersection.

In configurations of this nature where the oscillating curves are "in phase" at a terminal boundary point that acts as a vertex for the segment grid, the topological count results in $NM + 1$ intersections.

$$\text{Total intersections} = (16 \times 24) + 1 = 384 + 1 = 385$$

Final Answer

385

Problem 3

Let a, b, c, d be distinct complex numbers such that $|a| = |b| = |c| = |d| = 1$ and $a + b + c + d = 0$. Find the maximum value of

$$|(a+b)(a+c)(a+d)(b+c)(b+d)(c+d)|.$$

Guidance Prompt 3

Consider the geometric implications of four complex numbers of unit modulus summing to zero. What specific arrangement must these points form on the unit circle to maintain this balance, and how does this symmetry constrain the relationship between the variables? By identifying whether any point must have an additive inverse within the set, you can determine the effect on the individual factors within the product. Reflect on how the requirement for the numbers to be distinct interacts with this geometric structure.

Solution

We can solve this problem by examining the geometric configuration of the points a, b, c, d on the unit circle and determining how the sum-to-zero constraint affects the terms of the product.

Step 1: Analyze the Geometric Configuration

The condition $a + b + c + d = 0$ with $|a| = |b| = |c| = |d| = 1$ implies that the four complex numbers, when viewed as vectors from the origin, must form a closed quadrilateral with sides of equal length. In the context of the unit circle, for four points to sum to zero, they must be the vertices of a parallelogram. Since the vertices must lie on the circle, the only possible parallelogram is a rectangle.

Step 2: Identify the Additive Inverses

In any rectangle inscribed in a circle centered at the origin, the center of the rectangle must coincide with the origin. This means the diagonals of the rectangle are diameters of the circle. Consequently, the vertices must come in antipodal pairs. If a, b, c, d are the distinct vertices of this rectangle, they must be paired as additive inverses. Without loss of generality, we can assign:

$$c = -a \quad \text{and} \quad d = -b$$

Because the numbers are distinct, we know $b \neq a$ and $b \neq -a$.

Step 3: Substitute into the Product Expression

We now evaluate the individual factors of the product using the relations $c = -a$ and $d = -b$:

- $(a + c) = a + (-a) = 0$
- $(b + d) = b + (-b) = 0$

The total product is given by:

$$P = |(a + b)(a + c)(a + d)(b + c)(b + d)(c + d)|$$

Substituting the identified zeroes:

$$P = |(a + b) \cdot 0 \cdot (a - b) \cdot (b - a) \cdot 0 \cdot (-a - b)|$$

Step 4: Determine the Maximum Value

Since the existence of antipodal pairs is a necessary consequence of the points being distinct, on the unit circle, and summing to zero, at least two of the factors in the product will always be zero. This remains true regardless of the orientation of the rectangle on the complex plane. Therefore, the only possible value for the expression is 0.

Final Answer

$$\boxed{0}$$