ThinkPilot: Steering Reasoning Models via Automated Think-prefixes Optimization

Sunzhu Li[1], **Zhiyu Lin**[2] , **Shuling Yang**[1], **Jiale Zhao**[1],
Wei Chen[1] [*]
[1]Li Auto Inc., China
[2]The Chinese University of Hong Kong, Shenzhen, China
{lisunzhu, yangshuling, zhaojiale5, chenwei10}@lixiang.com
zhiyulin1@link.cuhk.edu.cn

Abstract

Large Reasoning Models (LRMs) are powerful, but they still suffer from inefficient and off-target reasoning. Currently, training-free methods are limited to either rigid heuristics or descriptive, non-actionable analyses. In this paper, we introduce ThinkPilot, a training-free framework that automatically optimizes LRMs reasoning. It uses an evolutionary process to generate *think-prefixes*, namely instructions that evolve driven by a taxonomy of *reasoning behaviors* to guide models toward superior performance. Extensive experiments demonstrate ThinkPilot's broad effectiveness: it significantly improves the accuracy-length tradeoff for efficient reasoning, drastically improves safety (e.g., cutting the StrongREJECT score of DeepSeek-R1-Distill-Qwen-32B from 27.0% to 0.7%), and enhances instruction following. It also synergizes with existing training-based methods. Specially, our analysis reveals that think-prefixes can reliably control LRMs' reasoning behaviors, and that different tasks have strong preferences for specific behavioral distributions. By automatically identifying and eliciting these behaviors, ThinkPilot provides a generalizable framework for aligning LRMs reasoning with task demands.

1 Introduction

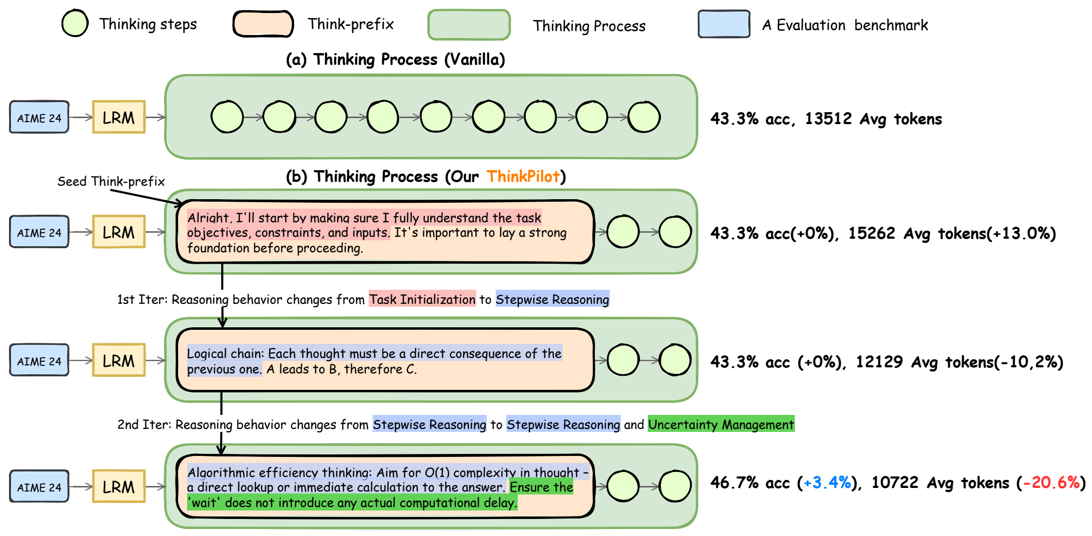
Large Reasoning Models (LRMs) (Jaech et al., 2024; Guo et al., 2025a) have achieved notable progress in complex tasks like math problem solving and code generation. These models support iterative thinking and better problem decomposition by generating detailed reasoning before final answers (Chen et al., 2025a). However, LRMs still face issues such as overly lengthy reasoning, and off-target responses that deviate from instructions or expectations, which wastes computation and harms answer quality (Chen et al., 2024; Cuadron et al., 2025; Gan et al., 2025). Thus, to improve performance, guiding LRMs toward more efficient and task-aligned reasoning patterns is essential.

To address these issues, existing efforts fall into two main categories. *Training-based* approaches adjust model parameters via supervised fine-tuning or reinforcement learning to encourage behaviors like safety or efficiency (Ma et al., 2025b; Aggarwal and Welleck, 2025; Chen et al., 2025a), but they require expensive supervision or task-specific reward design. In contrast, *training-free* methods steer reasoning without changing model weights, offering greater flexibility and scalability, which makes them especially attractive for practical deployment. Given these advantages, we focus on recent advances and challenges in *training-free* ones.

Among training-free methods, current research can be divided into two primary categories, each with notable limitations. First, **human-heuristic methods** (Wu et al., 2025a; Ma et al., 2025a; Wang et al., 2025a; Handelman, 2009) guide the reasoning process by injecting human-crafted phrases to make it more compact or safer. However, these heuristics often lack principled theoretical grounding, making them difficult to generalize across tasks and models. Second, **interpretability-driven analysis** (Wang et al., 2025b; Ghosal et al., 2025; Zhang et al., 2025b; Ma et al., 2025a; Wu et al., 2025a) has turned to understand the reasoning processes, such as assessing the importance of words or sentences within the reasoning paths. Yet, the efforts tend to remain descriptive, rarely yielding actionable strategies for model intervention. Naturally, these limitations raise a fundamental question: *can we develop an automatic and interpretability-driven framework, to efficiently discover the reasoning interventions for LRMs?*

In this paper, we introduce **ThinkPilot**, a training-free method that optimizes LRMs performance by strategically and automatically guiding their reasoning process. Inspired by Schoen-

[*] Corresponding Author


Thinking steps
Think-prefix
Thinking Process
A Evaluation benchmark
(a) Thinking Process (Vanilla)
AIME 24
LRM
43.3% acc, 13512 Avg tokens
(b) Thinking Process (Our ThinkPilot)
Seed Think-prefix
43.3% acc(+0%), 15262 Avg tokens(+13.0%)
1st Iter: Reasoning behavior changes from Task Initialization to Stepwise Reasoning
43.3% acc (+0%), 12129 Avg tokens(-10.2%)
2nd Iter: Reasoning behavior changes from Stepwise Reasoning to Stepwise Reasoning and Uncertainty Management
46.7% acc (+3.4%), 10722 Avg tokens (-20.6%)

Alright, I'll start by making sure I fully understand the task objectives, constraints, and inputs. It's important to lay a strong foundation before proceeding.

Logical chain: Each thought must be a direct consequence of the previous one. A leads to B, therefore C.

Algorithmic efficiency thinking: Aim for O(1) complexity in thought - a direct lookup or immediate calculation to the answer. Ensure the 'wait' does not introduce any actual computational delay.

Figure 1: **The comparison between (a) vanilla thinking process and (b) ThinkPilot**, which guides an LRM by iteratively optimizing think-prefixes based on reasoning behaviors. On the R1-Qwen-7B model, after two iterations, ThinkPilot improves accuracy by 3.4% while reducing average token usage by 20.6% on AIME 24.

feld's Episode Theory of human problem solving (Schoenfeld, 2014) and recent studies (Bogdan et al., 2025; Wang et al., 2025b; Venhoff et al., 2025; Li et al., 2025) on structured model reasoning, ThinkPilot introduces a principled taxonomy that defines *reasoning behaviors*—i.e., the observable and controllable strategies adopted by models during thinking. Built upon this taxonomy, ThinkPilot uses an evolution-inspired workflow to discover effective reasoning interventions. Specifically, it generates *think-prefixes*, which are interventional instructions inserted at the start of the thinking process, to trigger desired reasoning. Through iterative refinement, these prefixes gradually shift the reasoning behaviors they control, thereby enabling the identification of task-preferred reasoning behaviors and achieving superior performance. As shown in Figure 1 , ThinkPilot achieves desired performance with significantly lower token overhead compared to vanilla approaches.

Experimental results show that ThinkPilot demonstrates *broad effectiveness across diverse tasks*. In Efficient Reasoning, it significantly improves the model's accuracy-length trade-off, achieving higher accuracy with more concise outputs than baseline methods. its impact on Safety is particularly notable: it reduced the StrongREJECT score of R1-Qwen-32B from 27.0% to just 0.7%, without compromising other reasoning abilities. In Instruction Following, it boosted the IFEval score of R1-Qwen-32B by 6.4 points. Furthermore, consistent gains are observed on the logical reasoning benchmark BBH. Crucially, ThinkPilot also *synergizes with training-based methods*, further reducing SAFECHAIN's StrongREJECT score from an already low 19.4% to just 1.4%.

To further understand the source of ThinkPilot's performance gains, we conducted analysis and identified two key insights. First, existing studies show that LRMs may fail to follow the instructions for controlling thinking process (Wu et al., 2025a), whereas we demonstrate that *think-prefixes can reliably and precisely control reasoning behaviors for LRMs*. This enables LRMs to be steered in desired directions. Second, *different tasks favor distinct reasoning behaviors, and this preference strongly correlated to performance*. For example, behaviors that are helpful in some tasks may be ineffective or even harmful in others. This reveals the importance of aligning behavior strategies with task characteristics. In practice, ThinkPilot automatically identifies and elicits the behaviors each task prefers, thus ultimately leading to the performance that align with human expectations. In summary, our contributions are as follows:

- We propose ThinkPilot, a novel training-free framework that uses an evolutionary algorithm guided by a taxonomy of reasoning behaviors, automatically discovering the think-prefixes for steering model reasoning.
- We reveal that think-prefixes enable precise control of LRMs' reasoning behaviors, and that aligning these behaviors with task-

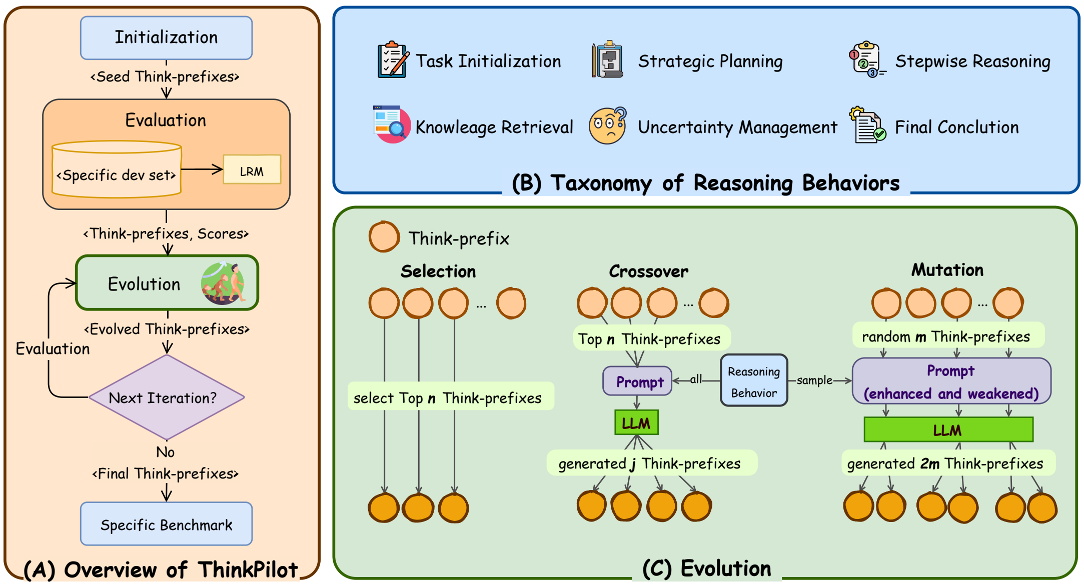


Figure 2: **Overview of the ThinkPilot**. The method optimizes think-prefixes through an evolutionary loop (A), where the evolution process (C) involves selection, crossover, and mutation, guided by the taxonomy of reasoning behaviors (B). The complete prompts for crossover and mutation can refer to Appendix E.

specific preferences boosts performance.

- Extensive experiments validate that ThinkPilot as a highly effective and general framework, significantly improves efficiency, safety, and instruction-following capabilities. Moreover, it can effectively synergize with existing training-based methods.

2 ThinkPilot

In this section, we introduce the workflow of *ThinkPilot*. As shown in Figure 2, the framework operates in two stages: an initialization–evaluation phase constructs and assesses seed think-prefixes, and an **evolution–iteration** phase. The latter combines *reasoning behavior modeling* (Figure 2 (B)) with evolutionary strategies (*selection*, *mutation*, and *crossover*) to iteratively refine think-prefixes (Figure 2 (C)) based on performance feedback, producing high-quality ones for downstream tasks.

2.1 Initialization and Evaluation

To initiate the iterative process, we automatically generate seed think-prefixes for each task using an LLM guided by prompts. This prompt-based generation produces diverse candidates varying in control strength, narrative style, and length, enriching the initial search space. The LLM generates seeds in a first-person perspective to emulate the model's internal monologue and align them with task objectives (e.g., using "`Ok, let's think concisely.`" to encourage brevity). In addition, we find that the optimization process is largely insensitive to the number or quality of the seeds—thus, even a small set of LLM-generated prefixes can effectively bootstrap the process (see Appendix B). Each resulting candidate is then evaluated on the downstream task using metrics such as accuracy or safety, providing feedback to guide further optimization.

2.2 Evolution and Iteration

The *Evolution and Iteration* phase optimizes think-prefixes using an evolutionary algorithm (EA) suited to the discrete, non-differentiable search space of natural language. Unlike gradient-based methods, EAs efficiently explore prompts as black-box optimizers (Guo et al., 2025b; Fernando et al., 2023). Guided by validation scores from the previous stage, the algorithm iteratively applies mutation, crossover, and selection to evolve new candidates (Figure 2 (C)). Our EA embeds the taxonomy of reasoning behaviors into these operations, enabling semantically guided rather than blind exploration, and repeats until convergence to yield prefixes optimized for the target LRM and task.

Taxonomy of Reasoning Behavior Our taxonomy is inspired by *Schoenfeld's Episode Theory* of human problem solving (Schoenfeld, 2014), which posits that reasoning unfolds through cognitively distinct episodes such as *Reading, Analyzing, Planning, Executing, Monitoring, and Evaluating*. It

Types	Definition	Example	Episode
Task Initialization	In the initial reasoning phase, the model identifies its task objectives, constraints, and inputs.	"Okay, I need to …", "My task is to …"	Reading, Analyzing
Strategic Planning	Before execution, explicitly state or determine a structured action plan or strategic blueprint.	"I will first …, then …", "To solve this, I'll …"	Planning
Knowledge Retrieval	Review relevant knowledge for problem-solving.	"According to my knowledge …"	Analyzing, Planning
Stepwise Reasoning	Execute independent reasoning or computation steps based on the planned logic.	"… So", "… Therefore", "… First"	Executing
Uncertainty Management	The model pauses and flags confusion or uncertainty when encountering ambiguity.	"Wait …", "Hmm …", "Well …", "Actually …"	Monitoring, Evaluating
Final Conclusion	Present the final conclusion.	"In conclusion …"	Evaluating, Summarizing

Table 1: **Taxonomy of reasoning behaviors** with definitions and examples, aligned with *Schoenfeld's Episode Theory* of human problem solving (Schoenfeld, 2014).

is also informed by recent studies (Bogdan et al., 2025; Wang et al., 2025b; Venhoff et al., 2025; Li et al., 2025) that model reasoning as a structured, stage-wise cognitive process in LRMs.

Building on these theoretical and empirical foundations, we map the typical reasoning episodes to six atomic behaviors frequently observed in LRMs (Table 1). This correspondence suggests that our taxonomy is not heuristically crafted but reflects stable reasoning patterns that emerge across diverse tasks and models, forming a *compact yet extensible behavioral basis* for controllable reasoning. We integrate this taxonomy as prior knowledge into our evolutionary process to guide the generation of effective think-prefixes.

Selection *Selection* preserves the top-scoring think-prefixes from the previous evaluation. The best *n* prefixes advance to the next round, preventing strong candidates from being lost and providing a solid basis for subsequent mutation and crossover, thereby ensuring steady evolutionary progress.

Crossover *Crossover* aims to generate novel think-prefixes by synthesizing complementary reasoning behaviors from the top n performing prefixes. The process leverages a Large Language Model (LLM), such as GPT-4o, guided by a few-shot prompt. Specifically, we select the top-n think-prefixes, denoted as $\{s_i\}_{i=1}^n$. These prefixes, along with a reasoning behavior classification (RB), are formatted into a tailored prompt, $\textit{Prompt}_{\text{crossover}}(\cdot)$. This prompt instructs the LLM to analyze the behaviors within $\{s_i\}$ and generate j new prefixes that effectively blend these complementary behaviors. The entire generation process can be formalized as follows:

$$\{c_i\}_{i=1}^j = \text{LLM}\Big(\textit{Prompt}_{\text{crossover}}\big(\{s_i\}_{i=1}^n, RB, j\big)\Big) \quad (1)$$

where $c_{i=1}^{j}$ are the resulting think-prefixes.

Mutation *Mutation* introduces targeted perturbations, guided by specific reasoning behaviors, to enhance population diversity and explore potentially superior think-prefixes. The process begins by randomly selecting m think-prefixes from the current population. For each selected prefix s, we independently assign a randomly chosen reasoning behavior rb (e.g., task initialization, strategic planning) to guide its transformation. Given that the optimal influence of a reasoning behavior may vary across tasks, we introduce two directional perturbations: *enhanced*, which amplifies the influence of the assigned behavior, and *weakened*, which reduces it. This bidirectional mechanism broadens the exploration of compatibility between think-prefixes and reasoning behaviors, thereby enhancing the effectiveness of the mutation operator.

To implement this, we designed a mutation prompt template, $\textit{Prompt}_{\text{mutation}}(\cdot)$. For each of the m pairs of a prefix s and its assigned reasoning behavior rb, we use this template to construct a prompt. This prompt instructs an instruction-following LLM to simultaneously generate two new think-prefixes: one *enhanced* version and one *weakened* version. Thus, m calls to the LLM produce a total of $2m$ new candidate think-prefixes. This mutation process for a single prefix s can be formalized as:

$$s_{\text{enhanced}}, s_{\text{weakened}} = \text{LLM}\Big(\textit{Prompt}_{\text{mutation}}\big(s, rb\big)\Big) \quad (2)$$

Iteration The three operations generate a new candidate set of think-prefixes, then re-evaluated by the *LRM* to start the next iteration (Figure 2). This process repeats until convergence, determined by a fixed iteration limit or performance threshold, ultimately yielding the optimal think-prefixes.

3 Experiments and Analysis

3.1 Experimental Setup

Tasks, benchmarks, and Metrics We evaluate ThinkPilot on the following diverse task types.

For **Efficient Reasoning**, we use the MATH 500 (Lightman et al., 2023), AIME 2024 (MAA, 2024), GPQA-Diamond (Rein et al., 2024), and AMC 2023 (AMC, 2025). During iteration, we use the Accuracy-per-Computation-Unit (ACU) (Ma et al., 2025b) to measure the performance-cost trade-off. ACU is defined as accuracy divided by the product of model size and generated tokens. For the final evaluation, we report PASS@1 accuracy and average generation length.

For **Safety**, we use XSTest (Röttger et al., 2023) (assessing Safe Prompt Compliance, SPC, and Unsafe Prompt Refusal, UPR) and StrongREJECT (Souly et al., 2024) (evaluating harmful content generation ability, SRC). To monitor for overfitting and capability degradation, we concurrently test on MATH, GPQA, and AIME. During development iterations, we also used specific proxy metrics to monitor the model's responses to both safe and harmful prompts (see Appendix A for details).

For **Instruction Following**, we evaluate on IFEval (Zhou et al., 2023a) and MultiChallenge (Sirdeshmukh et al., 2025), using strict accuracy (exact match with all constraints), a metric applied during both iterative optimization and final evaluation.

For **Logic**, we evaluate on BBH (Suzgun et al., 2022) with accuracy as the metric. Full experimental details are in Appendix A.

Baselines. To ensure fair and comprehensive comparisons, we categorize baselines into three types: *backbone models*, *training-free methods*, and *training-based methods*, tailored for each task.

For **Efficient Reasoning**, the *backbone models* include DeepSeek-R1-Distill-Qwen-1.5B (Guo et al., 2025a), Qwen3-8B (Yang et al., 2025a), and QwQ-32B (Yang et al., 2024). *Training-free methods* include two static and three dynamic intervention approaches. The static ones are CoD (Xu et al., 2025), which applies lightweight prompting to guide reasoning, and NoThink (Ma et al., 2025a), a non-reasoning control baseline. The dynamic ones comprise DSPy (Khattab et al., 2023) for adaptive prompt optimization, DEER (Yang et al., 2025b) for detecting reasoning turning points and stopping generation, and Dynasor-CoT (Fu et al., 2024) for adaptive reasoning allocation and early termination. *Training-based methods* include THINKPRUNE (Hou et al., 2025) and the one by Arora and Zanette (2025), which use RL to refine think-prefixes.

For **Safety**, the *backbone models* include DeepSeek-R1-Distill-Qwen-7B/32B and Qwen3-8B. The *training-free method* is ThinkingI (Wu et al., 2025a), while *training-based methods* include SAFECHAIN (Jiang et al., 2025) and RealSafe-R1 (Zhang et al., 2025c).

For **Instruction Following**, the *backbone models* are Qwen3-8B, DeepSeek-R1-Distill-Qwen-32B, and QwQ-32B. The sole *training-free method* is ThinkingI. No training-based methods were evaluated for these tasks. Finally, For **Logic**, the *backbone models* are Qwen3-8B and Qwen3-32B.

Unless otherwise noted, all baseline results are from our reproductions. The detailed experimental settings and prompts of all baselines are provided in Appendix A.4.

ThinkPilot demonstrates broad effectiveness across multiple tasks. On **Efficient Reasoning**, ThinkPilot substantially improves the accuracy-length trade-off (Table 2). Compared to static intervention methods (NoThink, CoD), it not only achieves the highest average accuracy (57.1%, 79.4%, and 84.8%), but also produces more concise reasoning than the vanilla. When compared to dynamic intervention methods (DSPy, Dynasor-CoT, and DEER), ThinkPilot demonstrates consistent advantages: on R1-Qwen-1.5B, it surpasses DSPy by +2.7% in accuracy while using 3704 fewer tokens on average; on QwQ-32B, it outperforms DSPy and Dynasor-CoT in both ΔLen and ΔAcc, achieving comparable token efficiency to DEER. While a modest accuracy gap exists relative to DEER, it is noteworthy that DEER requires three additional online modules, increasing inference overhead. In contrast, ThinkPilot's deployment-free nature offers a clear efficiency advantage.

In **Safety**, ThinkPilot also shows substantial advantages (Table 3), outperforming both vanilla and ThinkingI methods on XSTest and StrongREJECT. Notably, ThinkPilot reduces the harmful output rate of R1-Qwen-32B on StrongREJECT from 27.0%

Backbone	Method	MATH 500		AIME 2024		GPQA-D		AMC 2023		Average			
		Acc.	Len.	Acc.	Len.	Acc.	Len.	Acc.	Len.	Acc.	Len.	ΔAcc.	ΔLen.
Training-Free													
R1-Qwen-1.5B	Vanilla	79.7	4619	26.2	15161	**39.4**	10139	70.2	9436	53.9	9839	–	–
	NoThink	62.9	**809**	11.0	**3157**	33.5	**879**	42.4	**1540**	37.5	**1596**	−16.4	−83.8%
	CoD	75.8	2557	26.2	9969	35.5	9299	61.9	5138	49.9	6741	−4.0	−31.5%
	DSPy	79.2	4752	30.0	16667	35.9	10172	72.5	10884	54.4	10169	+0.5	+3.3%
	ThinkPilot	**81.0**	2547	**33.3**	8569	**39.4**	8340	**74.8**	6405	**57.1**	6465	+3.2	−34.2%
Qwen3-8B	Vanilla	**93.3**	5026	73.3	14989	58.9	6964	91.6	6569	79.3	8387	–	–
	NoThink	82.3	**916**	32.5	**4904**	47.7	**1383**	70.4	**2097**	58.2	**2325**	−21.1	−72.3%
	CoD	92.6	2724	**74.2**	14267	55.7	3137	**93.9**	5619	79.1	6512	−0.2	−22.4%
	ThinkPilot	93.2	3900	72.5	13083	**59.9**	5904	92.0	5797	**79.4**	7171	+0.1	−14.5%
QwQ-32B	Vanilla	**94.1**	3916	80.6	11536	64.4	7590	97.8	6954	84.2	7499	–	–
	NoThink	76.1	3413	80.6	13010	63.7	**4894**	92.3	7379	78.2	7174	−6.0	−4.3%
	CoD	93.2	**2717**	77.3	10676	64.3	6586	97.2	**5524**	83.0	6376	−1.2	−15.0%
	Dynasor-CoT◊	94.2	4176	63.3	11156	64.1	7024	93.8	6544	78.9	7225	−0.1	−1.8%
	DEER◊	94.6	3316	70.0	10097	64.1	6163	95.0	5782	80.9	**6340**	+1.9	−13.9%
	DSPy	94.0	4630	80.0	14673	67.2	9092	95.0	8391	84.1	9197	−0.1	+22.6%
	ThinkPilot	93.5	3272	**80.8**	**10058**	**65.9**	6709	**98.9**	6378	**84.8**	6604	+0.6	−11.9%
Training-Based													
R1-Qwen-1.5B	Arora and Zanette (2025)	80.3	2500	**29.8**	9162	36.5	7302	**73.3**	4699	55.0	5916	–	–
	+ ThinkPilot	**82.7**	**2106**	29.6	**7806**	**38.4**	**6576**	72.2	**4485**	**55.7**	**5243**	+0.7	−11.4%
R1-Qwen-7B	Arora and Zanette (2025)	89.7	2749	52.3	10392	**50.1**	7077	88.1	5057	70.1	6319	–	–
	+ ThinkPilot	**90.0**	**2376**	**55.8**	**8264**	49.7	**5977**	**90.6**	**4448**	**71.5**	**5266**	+1.4	−16.7%
QwQ-32B	THINKPRUNE	**92.2**	2052	72.8	7672	**63.3**	4314	95.3	3589	80.9	4407	–	–
	+ ThinkPilot	92.1	**1615**	**76.9**	**7167**	61.7	**4050**	**95.9**	**3150**	**81.7**	**3996**	+0.8	−9.3%

Table 2: Comparison of different methods on the Efficient Reasoning task. We report accuracy (**Acc.**, ↑) and response length (**Len.**, ↓) on four benchmarks. **Bold** marks the best metric within each backbone group. Specifically, ThinkPilot selects the shortest token length interval that attains near-peak accuracy (see Appendix D for performance–cost analysis). ◊ indicates that the results were taken from DEER, and Δ denotes the difference between their average accuracy/length and the vanilla QwQ-32B baseline (79% accuracy, 7360 tokens) reported in DEER.

to a minimal 0.7% without degrading its reasoning performance. On **Instruction Following**, it enhances the model's adherence to complex constraints (Table 4), boosting the IFEval score of the vanilla R1-Qwen-32B by 6.4 points and outperforming ThinkingI. Finally, on **Logic**, ThinkPilot shows steady gains—improving BBH by 2.6 and 3.3 on Qwen3-8B/32B (see Appendix C).

ThinkPilot synergizes effectively with training-based methods. In **Efficient Reasoning**, while Arora and Zanette (2025) shorten responses from R1-Qwen-1.5B by 3923 tokens compared to vanilla, integrating ThinkPilot achieves an additional reduction of about 700 tokens without compromising accuracy (Table 2). In **Safety**, SAFECHAIN reduces the StrongREJECT score from 30.8% to 19.4%. ThinkPilot further enhances safety, lowering the score to just 1.4% (Table 3). These results demonstrate ThinkPilot's effectiveness when integrated with specialized LRMs.

3.2 Analysis of Reasoning Behaviors

Think-prefixes reliably control diverse reasoning behaviors. We investigated the effectiveness of think-prefixes for controlling seven reasoning behaviors. For a robust evaluation, we employed a three-model committee (Gemini 2.5 Pro, GPT-5, Claude Sonnet 4) and used the majority vote as the final verdict. The reliability of this committee evaluation is substantiated by a Fleiss' Kappa of $\varkappa = 0.4370$. This value indicates a "Moderate Agreement" that is statistically significant, confirming the consistency of our evaluators' judgments beyond

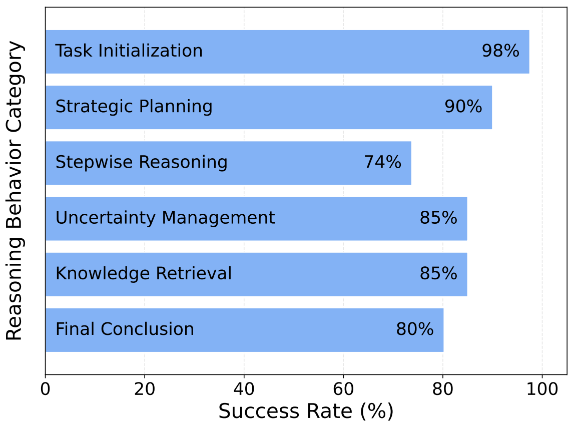


Figure 3: Control Success Rate for reasoning behaviors on AMC 23 for R1-Qwen-7B.

Method	XSTest		StrongREJECT	
	SPC (↑)	UPR (↑)	SRC (↓)	RA (↑)
Training-Free				
R1-Qwen-7B				
Vanilla	100.0	45.0	30.8	89.2/49.0/53.3
ThinkingI	$34.5_{-65.5}$	$85.6_{+40.6}$	$12.2_{-18.6}$	79.6/49.7/39.6
ThinkPilot	**98.0**$_{-2.0}$	**67.5**$_{+22.5}$	**0.4**$_{-30.4}$	89.8/52.0/56.7
Qwen3-8B				
Vanilla	98.0	62.5	5.2	93.0/58.1/70.0
ThinkingI	$62.5_{-35.5}$	$81.9_{+16.7}$	$1.6_{-3.6}$	46.4/42.9/30.0
ThinkPilot	**91.5**$_{-6.5}$	**82.5**$_{+20.0}$	**0.4**$_{-4.8}$	92.8/57.6/73.3
R1-Qwen-32B				
Vanilla	100.0	55.0	27.0	92.0/63.6/73.3
ThinkingI	$95.0_{-5.0}$	$75.6_{+20.6}$	$2.5_{-24.5}$	89.4/62.9/69.4
ThinkPilot	**100.0**$_{-0.0}$	**97.5**$_{+42.5}$	**0.7**$_{-26.3}$	93.2/64.7/73.3
Training-Based				
R1-Qwen-7B				
SAFECHAIN	96.5	69.4	19.4	88.5/48.3/45.4
+ ThinkPilot	**95.0**$_{-1.5}$	**74.4**$_{+5.0}$	**1.4**$_{-18.0}$	88.1/49.4/47.7
R1-Qwen-32B				
RealSafe-R1	79.5	95.6	**0.0**	92.0/63.1/80.0
+ ThinkPilot	**85.5**$_{+6.0}$	**97.5**$_{+1.9}$	**0.0**$_{+0.0}$	91.8/63.1/73.3

Table 3: Comparison of different methods on the Safety task. Metrics include Safe Prompt Compliance (SPC, ↑), Unsafe Prompt Refusal (UPR, ↑), and StrongREJECT Score (SRC, ↓). Reasoning Ability (RA, ↑) reflects accuracies on MATH 500, GPQA-Diamond, and AIME 24, monitoring reasoning performance. Colored subscripts denote score changes from the vanilla baseline—green for increases and red for decreases. **Bold** indicates the best result within each model group.

Backbone	Method	IFEval	MultiChallenge
Qwen3-8B	Vanilla	85.7	22.4
	ThinkPilot	**86.1**$_{+0.4}$	**30.1**$_{+7.7}$
R1-Qwen-32B	Vanilla	75.4	25.1
	ThinkingI◊	$77.1_{+1.7}$	–
	ThinkPilot	**81.8**$_{+6.4}$	**48.8**$_{+23.7}$
QwQ-32B	Vanilla	82.0	35.6
	ThinkingI◊	$82.3_{+0.3}$	–
	ThinkPilot	**83.6**$_{+1.6}$	**47.5**$_{+11.9}$

Table 4: Comparison of different methods on the Instruction Following task. Scores represent strict accuracy (↑). The green subscripts indicate the score increase relative to the Vanilla baseline. **Bold** values highlight the top-performing method for each backbone. ◊ indicates that the result is taken from the original paper.

what would be expected by chance. This validates the use of the majority vote as a reliable final verdict. As shown in Figure 3, our results reveal a clear spectrum of controllability. Success rates are substantial, exceeding 80% for highly malleable behaviors like Strategic Planning but registering near 50 to 70% for more foundational ones like Uncertainty Management. This variance suggests that reasoning behaviors have distinct levels of inherent plasticity. The ability to quantify this differential response highlights the necessity of a system like **ThinkPilot**, which can automatically discover and leverage the most effective and controllable behaviors for a given task.

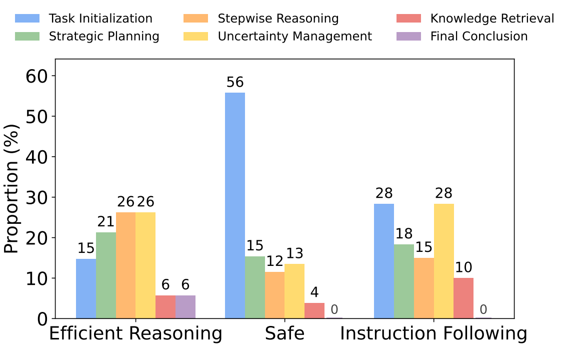


Figure 4: Reasoning behaviors distribution of top 10% think-prefixes in the QwQ-32B model on three tasks.

Guiding LRMs with task-preferred reasoning behaviors enhances performance. Given that ThinkPilot improves performance by searching for *think-prefixes*, we analyzed the top-performing prefixes for each task and found that they indeed exhibit distinct reasoning behavior preferences (Figure 4). For instance, high-performing prefixes in Safety favor task initialization, whereas those in Efficient Reasoning prefer stepwise reasoning and uncertainty management.

To verify the reliability of these preferences, we designed a controlled experiment evaluating performance under five conditions: (1) Vanilla: baseline model; (2) w/o Behaviors: Iteration without specific behavior guidance; (3) Non-preferred: Guided only by "non-preferred behaviors"; (4) Preferred: Guided only by "preferred behaviors"; and (5) All Behaviors: The full ThinkPilot method. The "preferred" and "non-preferred" behaviors were categorized based on the analysis in Figure 4.

The results (Figure 5) clearly demonstrate the impact of guidance. Compared to the Vanilla baseline (35.6%), guiding with only "preferred behaviors" (Preferred) substantially boosted performance to 47.5%, matching the result of the full method using All Behaviors. In contrast, iteration w/o Behaviors yielded only a slight improvement (37.0%), while guidance with Non-preferred behaviors was detrimental, causing performance to drop to 30.6%. This strongly indicates that the key to ThinkPilot's performance gains lies in identifying and leveraging task-preferred reasoning behaviors. Conversely, misguided guidance is not only ineffective but can be counterproductive.

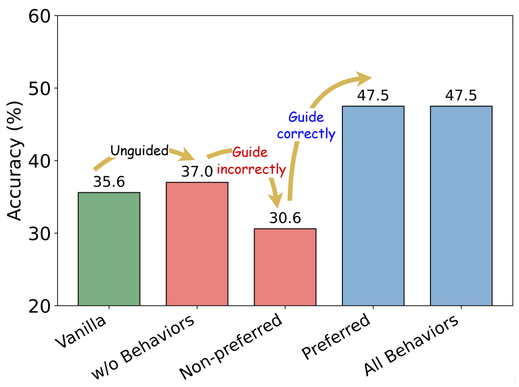


Figure 5: Comparison of iterative optimization under different reasoning behavior guidance settings, evaluated on the QwQ-32B model on Instruction Following. The chart contrasts the **Vanilla** baseline and the full ThinkPilot method (**All Behaviors**) with three variations: ThinkPilot without guidance (**w/o Behaviors**), with **non-preferred** behaviors, and with **preferred** behaviors. The annotated arrows illustrate the performance changes under these different guidance settings.

4 Related Work and Discussion

The Control of Thinking Process To better align LRMs reasoning with task goals, prior work explores both training-based and training-free control methods. Training-based approaches adjust model weights via supervised fine-tuning or reinforcement learning (Ma et al., 2025b; Sui et al., 2025; Aggarwal and Welleck, 2025; Luo et al., 2025; Yuan et al., 2025; Chen et al., 2025b), but are resource-intensive. In contrast, training-free methods guide the model's reasoning process through Prompt Engineering (PE) (Hu et al., 2023; Wang et al., 2024; Zhao et al., 2023; Zhou et al., 2023b) or by directly intervening in the model's internal thinking process using dynamic paradigms or explicit instructions (Wang et al., 2025a; Zhang et al., 2025a; Wu et al., 2025b; Lin et al., 2025; Ma et al., 2025a; Wu et al., 2025a). However, these methods often rely on heuristic design—a key limitation our work aims to overcome.

Interpretability of Thinking Process Recent research (Bogdan et al., 2025; Wang et al., 2025b; Venhoff et al., 2025) has focused on how the thinking process affects model performance. For instance, some studies (Wang et al., 2025b; Ghosal et al., 2025; Zhang et al., 2025b; Qian et al., 2025) identify key terms that strongly influence final outputs using entropy analysis, while others (Venhoff et al., 2025; Bogdan et al., 2025) investigate reasoning patterns by summarizing and generalizing typical thinking paradigms. In this study, informed by related work and our observations of the model's reasoning behaviors, we propose a taxonomy of reasoning behaviors. This taxonomy acts as prior knowledge for our evolutionary approach, guiding the evolution of think-prefixes.

Discussion on Static and Dynamic Intervention ThinkPilot adopts a *static, prefix-level intervention* for two key reasons: *deployment simplicity* and *runtime efficiency*. It is plug-and-play with existing inference APIs, requiring no decoding or monitoring changes, whereas dynamic methods (e.g., DEER's reasoning monitor or confidence evaluator) add deployment and runtime overhead. A static prefix incurs negligible latency, while dynamic control requires continuous monitoring, as in Speculative Thinking (Fu et al., 2025), which runs large and small models concurrently.

We view dynamic control as a *complementary* direction. Future extensions of ThinkPilot could monitor uncertainty cues (e.g., "wait", "but") and inject adaptive reasoning behaviors, such as *Final Conclusion* ("ok… let me answer this…"), to provide real-time guidance during reasoning.

5 Conclusions

We introduce ThinkPilot, a training-free and plug-and-play framework that automatically optimizes the reasoning of LRMs. By leveraging a taxonomy of reasoning behaviors, ThinkPilot employs an evolution-inspired workflow to discover optimal think-prefixes that effectively guide a model's thinking process. Our work yields two key insights: first, think-prefixes are a reliable means of controlling reasoning behavior of LRMs, and second, different tasks show different preference distributions for reasoning behaviors. ThinkPilot can be regarded as a form of prompt engineering at the level of model thinking processes. More importantly, by centering on reasoning behavior, it opens a new perspective for understanding and steering the internal reasoning of LRMs. This has significant implications for the future design and alignment of reliable and controllable models.

6 Limitations

While ThinkPilot demonstrates broad effectiveness across reasoning tasks, one limitation opens up promising directions for future work. At present, ThinkPilot controls reasoning through a static think-prefix inserted before generation. This design ensures simplicity and zero deployment overhead, but it remains fixed once reasoning unfolds. A natural extension would be to evolve ThinkPilot into a dynamic intervention framework—one that monitors the model's internal signals (e.g., uncertainty, self-correction triggers) and adaptively adjusts guidance in real time, allocating reasoning budget more efficiently or terminating early when sufficient confidence is reached.

References

Pranjal Aggarwal and Sean Welleck. 2025. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*.

AMC. 2025. American mathematics competitions (amc). https://maa.org/student-programs/amc/.

Daman Arora and Andrea Zanette. 2025. Training language models to reason efficiently. *arXiv preprint arXiv:2502.04463*.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and 1 others. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17682–17690.

Paul C Bogdan, Uzay Macar, Neel Nanda, and Arthur Conmy. 2025. Thought anchors: Which llm reasoning steps matter? *arXiv preprint arXiv:2506.19143*.

Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. 2025a. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *arXiv preprint arXiv:2503.09567*.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, and 1 others. 2024. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*.

Yanda Chen, Joe Benton, Ansh Radhakrishnan, Jonathan Uesato, Carson Denison, John Schulman, Arushi Somani, Peter Hase, Misha Wagner, Fabien Roger, and 1 others. 2025b. Reasoning models don't always say what they think. *arXiv preprint arXiv:2505.05410*.

Alejandro Cuadron, Dacheng Li, Wenjie Ma, Xingyao Wang, Yichuan Wang, Siyuan Zhuang, Shu Liu, Luis Gaspar Schroeder, Tian Xia, Huanzhi Mao, and 1 others. 2025. The danger of overthinking: Examining the reasoning-action dilemma in agentic tasks. *arXiv preprint arXiv:2502.08235*.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. 2023. Promptbreeder: Self-referential self-improvement via prompt evolution. *arXiv preprint arXiv:2309.16797*.

Yichao Fu, Junda Chen, Siqi Zhu, Zheyu Fu, Zhongdongming Dai, Yonghao Zhuang, Yian Ma, Aurick Qiao, Tajana Rosing, Ion Stoica, and 1 others. 2024. Efficiently scaling llm reasoning with certaindex. *arXiv preprint arXiv:2412.20993*.

Yichao Fu, Junda Chen, Yonghao Zhuang, Zheyu Fu, Ion Stoica, and Hao Zhang. 2025. Reasoning without self-doubt: More efficient chain-of-thought through certainty probing. In *ICLR 2025 Workshop on Foundation Models in the Wild*.

Zeyu Gan, Yun Liao, and Yong Liu. 2025. Rethinking external slow-thinking: From snowball errors to probability of correct reasoning. *arXiv preprint arXiv:2501.15602*.

Soumya Suvra Ghosal, Souradip Chakraborty, Avinash Reddy, Yifu Lu, Mengdi Wang, Dinesh Manocha, Furong Huang, Mohammad Ghavamzadeh, and Amrit Singh Bedi. 2025. Does thinking more always help? understanding test-time scaling in reasoning models. *arXiv preprint arXiv:2506.04210*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025a. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. 2025b. Evoprompt: Connecting llms with evolutionary algorithms yields powerful prompt optimizers. *arXiv preprint arXiv:2309.08532*.

Sapir Handelman. 2009. *Thought manipulation: the use and abuse of psychological trickery*. Bloomsbury Publishing USA.

Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. 2025. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning. *arXiv preprint arXiv:2504.01296*.

Hanxu Hu, Hongyuan Lu, Huajian Zhang, Yun-Ze Song, Wai Lam, and Yue Zhang. 2023. Chain-of-symbol prompting elicits planning in large langauge models. *arXiv preprint arXiv:2305.10276*.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Fengqing Jiang, Zhangchen Xu, Yuetai Li, Luyao Niu, Zhen Xiang, Bo Li, Bill Yuchen Lin, and Radha Poovendran. 2025. Safechain: Safety of language models with long chain-of-thought reasoning capabilities. *arXiv preprint arXiv:2502.12025*.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T Joshi, Hanna Moazam, and 1 others. 2023. Dspy: Compiling declarative language model calls into self-improving pipelines. *arXiv preprint arXiv:2310.03714*.

Celine Lee, Alexander M Rush, and Keyon Vafa. 2025. Critical thinking: Which kinds of complexity govern optimal reasoning length? *arXiv preprint arXiv:2504.01935*.

Ming Li, Nan Zhang, Chenrui Fan, Hong Jiao, Yanbin Fu, Sydney Peters, Qingshu Xu, Robert Lissitz, and Tianyi Zhou. 2025. Understanding the thinking process of reasoning models: A perspective from schoenfeld's episode theory. *arXiv preprint arXiv:2509.14662*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Kevin Lin, Charlie Snell, Yu Wang, Charles Packer, Sarah Wooders, Ion Stoica, and Joseph E Gonzalez. 2025. Sleep-time compute: Beyond inference scaling at test-time. *arXiv preprint arXiv:2504.13171*.

Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. 2025. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning. *arXiv preprint arXiv:2501.12570*.

Wenjie Ma, Jingxuan He, Charlie Snell, Tyler Griggs, Sewon Min, and Matei Zaharia. 2025a. Reasoning models can be effective without thinking. *arXiv preprint arXiv:2504.09858*.

Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. 2025b. Cot-valve: Length-compressible chain-of-thought tuning. *arXiv preprint arXiv:2502.09601*.

MAA. 2024. American invitational mathematics examination – aime. In *American Invitational Mathematics Examination – AIME 2024*, February 2024.

Chen Qian, Dongrui Liu, Haochen Wen, Zhen Bai, Yong Liu, and Jing Shao. 2025. Demystifying reasoning dynamics with mutual information: Thinking tokens are information peaks in llm reasoning. *arXiv preprint arXiv:2506.02867*.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

Paul Röttger, Hannah Rose Kirk, Bertie Vidgen, Giuseppe Attanasio, Federico Bianchi, and Dirk Hovy. 2023. Xstest: A test suite for identifying exaggerated safety behaviours in large language models. *arXiv preprint arXiv:2308.01263*.

Alan H Schoenfeld. 2014. *Mathematical problem solving*. Elsevier.

Ved Sirdeshmukh, Kaustubh Deshpande, Johannes Mols, Lifeng Jin, Ed-Yeremai Cardona, Dean Lee, Jeremy Kritz, Willow Primack, Summer Yue, and Chen Xing. 2025. Multichallenge: A realistic multi-turn conversation evaluation benchmark challenging to frontier llms. *arXiv preprint arXiv:2501.17399*.

Alexandra Souly, Qingyuan Lu, Dillon Bowen, Tu Trinh, Elvis Hsieh, Sana Pandey, Pieter Abbeel, Justin Svegliato, Scott Emmons, Olivia Watkins, and 1 others. 2024. A strongreject for empty jailbreaks. *arXiv preprint arXiv:2402.10260*.

Jinyan Su, Jennifer Healey, Preslav Nakov, and Claire Cardie. 2025. Between underthinking and overthinking: An empirical study of reasoning length and correctness in llms. *arXiv preprint arXiv:2505.00127*.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Hanjie Chen, Xia Hu, and 1 others. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, and 1 others. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.

Constantin Venhoff, Iván Arcuschin, Philip Torr, Arthur Conmy, and Neel Nanda. 2025. Understanding reasoning in thinking language models via steering vectors. *arXiv preprint arXiv:2506.18167*.

Chenlong Wang, Yuanning Feng, Dongping Chen, Zhaoyang Chu, Ranjay Krishna, and Tianyi Zhou. 2025a. Wait, we don't need to" wait"! removing thinking tokens improves reasoning efficiency. *arXiv preprint arXiv:2506.08343*.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, and 1 others. 2025b. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.01939*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Zilong Wang, Hao Zhang, Chun-Liang Li, Julian Martin Eisenschlos, Vincent Perot, Zifeng Wang, Lesly Miculicich, Yasuhisa Fujii, Jingbo Shang, Chen-Yu Lee, and 1 others. 2024. Chain-of-table: Evolving tables in the reasoning chain for table understanding. *arXiv preprint arXiv:2401.04398*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Tong Wu, Chong Xiang, Jiachen T Wang, and Prateek Mittal. 2025a. Effectively controlling reasoning models through thinking intervention. *arXiv preprint arXiv:2503.24370*.

Yuyang Wu, Yifei Wang, Tianqi Du, Stefanie Jegelka, and Yisen Wang. 2025b. When more is less: Understanding chain-of-thought length in llms. *arXiv preprint arXiv:2502.07266*.

Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. 2025. Chain of draft: Thinking faster by writing less. *arXiv preprint arXiv:2502.18600*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Chenxu Yang, Qingyi Si, Yongjie Duan, Zheliang Zhu, Chenyu Zhu, Qiaowei Li, Zheng Lin, Li Cao, and Weiping Wang. 2025b. Dynamic early exit in reasoning models. *arXiv preprint arXiv:2504.15895*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Hang Yuan, Bin Yu, Haotian Li, Shijun Yang, Christina Dan Wang, Zhou Yu, Xueyin Xu, Weizhen Qi, and Kai Chen. 2025. Not all tokens are what you need in thinking. *arXiv preprint arXiv:2505.17827*.

Anqi Zhang, Yulin Chen, Jane Pan, Chen Zhao, Aurojit Panda, Jinyang Li, and He He. 2025a. Reasoning models know when they're right: Probing hidden states for self-verification. *arXiv preprint arXiv:2504.05419*.

Yanzhi Zhang, Zhaoxi Zhang, Haoxiang Guan, Yilin Cheng, Yitong Duan, Chen Wang, Yue Wang, Shuxin Zheng, and Jiyan He. 2025b. No free lunch: Rethinking internal feedback for llm reasoning. *arXiv preprint arXiv:2506.17219*.

Yichi Zhang, Zihao Zeng, Dongbai Li, Yao Huang, Zhijie Deng, and Yinpeng Dong. 2025c. Realsafe-r1: Safety-aligned deepseek-r1 without compromising reasoning capability. *arXiv preprint arXiv:2504.10081*.

Xufeng Zhao, Mengdi Li, Wenhao Lu, Cornelius Weber, Jae Hee Lee, Kun Chu, and Stefan Wermter. 2023. Enhancing zero-shot chain-of-thought reasoning in large language models through logic. *arXiv preprint arXiv:2309.13339*.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023a. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.

Yucheng Zhou, Xiubo Geng, Tao Shen, Chongyang Tao, Guodong Long, Jian-Guang Lou, and Jianbing Shen. 2023b. Thread of thought unraveling chaotic contexts. *arXiv preprint arXiv:2311.08734*.

A Detailed Experiment Setup

To ensure the rigor and reproducibility of our results, and to prevent overfitting on our test benchmarks, we adopted a strict protocol for dataset management, model configuration, and evaluation.

A.1 Dataset Splitting Methodology

A.1.1 Safety, Instruction Following , and Logical Benchmarks

We utilize several benchmarks to evaluate safety and instruction-following abilities of language models. For each, we partitioned the data into validation and test splits, with 20% of the original instances randomly sampled as validation set and the remaining 80% designated as the test set, unless otherwise specified.

XSTest consists of 450 prompts, divided into 250 safe requests and 200 unsafe requests. The benchmark is specifically designed to examine the potential for exaggerated safety behaviors among large language models.

StrongREJECT is a recently introduced benchmark containing 313 malicious prompts, curated for the purpose of evaluating the robustness of LLMs against jailbreaking attacks, and determining whether such attacks enable misuse for malicious activities.

IFEval focuses on instruction-following capabilities; it features approximately 500 prompts that span 25 distinct instruction types, thus providing comprehensive coverage for evaluating instruction compliance.

MultiChallenge comprises 273 multi-turn conversation samples, aiming to measure the ability of large language models to engage in complex, multi-turn dialogues—a critical ability for real-world applications.

BBH consists of 23 challenging tasks that mostly require multi-step reasoning to solve.

A.1.2 Efficient Reasoning Benchmarks

For mathematical and scientific reasoning, we evaluate on the following datasets, each with their distinct validation/test configurations.

MATH 500 is derived from OpenAI's Let's Verify Step by Step paper and contains two splits—500 training and 500 test problems—each consisting of challenging mathematics questions. For our experiments, we use 500 samples from the training split for validation and the entire test split as our test set.

AIME 2024 comprises 30 problems from the 2024 American Invitational Mathematics Examination (AIME), a renowned mathematics competition for high school students that is well known for its problem difficulty. The 2023 set, which also contains 30 problems, serves as our validation set, while the 2024 set is used for testing.

GPQA is a rigorous multiple-choice question-answering dataset spanning biology, physics, and chemistry, with questions crafted by domain experts. The GPQA_main split contains 448 questions and is used for validation, while GPQA-D consists of 198 challenging domain transfer questions designated as our test set.

AMC 2023 contains 40 questions from the 2023 American Mathematics Competitions, with AMC 2022 having 43 questions and serving as validation. The AMC benchmarks target the evaluation of mathematical problem-solving abilities, providing diverse and difficult problems from annual nation-wide contests.

A.2 Generation Parameters

For all evaluations conducted across safety, instruction following, and efficient reasoning domains, model responses were generated using a consistent set of decoding parameters to ensure comparability:

Temperature: `0.6`

Top-p: `0.95`

Maximum Output Tokens: `32,768`

A.3 Two-Phase Evaluation Workflow

Throughout all three domains, our experiments rigorously adhered to strict data separation and consistent model configurations.

In the **first phase**, all model development, including hyperparameter tuning and iterative optimization, was conducted solely on the validation sets, with domain-specific metrics such as Safe Prompt Compliance (SPC), Unsafe Prompt Refusal (UPR), instruction-following accuracy, and problem-solving accuracy continuously monitored to guide improvements.

For the **second phase**, the held-out test sets were accessed only once after development was complete, and all reported results are from this single final evaluation. This protocol ensures the objectivity and validity of our performance measurements, reflecting true generalization.

A.4 Baseline Implementation Details

This appendix lists the exact experimental settings (prompts and parameters) of all baselines to ensure transparency and reproducibility. For all reproduced baseline methods, we have made our best effort to adhere to the settings described in their original papers. Any necessary modifications or implementation choices made for our specific experimental context are explicitly stated below.

CoD: A heuristic static method applied to system prompts. We strictly follow the original template proposed by Chain-of-Draft: "Think step by step, but only keep a minimum draft for each thinking step, with 5 words at most. Return the answer at the end...". We reproduced its results on AIME 24, AMC 23, MATH 500, and GPQA-D.

NoThink: A heuristic static method applied during the thinking phase. Following the original No-Think paper, we insert the placeholder "<think> Okay, I think I have finished thinking. </think>" to skip redundant thinking process. We reproduced the results on the four efficient reasoning benchmarks.

ThinkingI: A dynamic intervention method injected during <think>...</think> phases. Since the original paper did not evaluate efficient reasoning, we reproduced results on safety benchmarks (XSTest, StrongREJECT), using the heuristic prefix "I am a helpful, respectful, and honest assistant.". On Instruction Following, IF-Eval results are directly taken from the original paper.

DSPy: A framework for adaptive prompt optimization. We reproduced results on the four efficient reasoning benchmarks using the same settings as the original paper (MAX TOKENS=32768, TEMPERATURE=0.6).

DEER: A dynamic intervention framework that monitors reasoning turning points and terminates generation early (e.g., when detecting "Wait"). We directly report the results from the DEER paper on the four efficient reasoning benchmarks.

Dynasor-CoT: A dynamic intervention approach based on Certaindex, which allocates reasoning resources adaptively or terminates early. For comparison on the four efficient reasoning benchmarks, we use the Dynasor-CoT results as reported in the DEER paper.

B Ablation Study on the Robustness of Initial Seeds

In our framework, the initial population of think-prefixes is automatically generated by a Large Language Model (LLM) to bootstrap the evolutionary process. While this automates seed creation, a natural question arises regarding the sensitivity of ThinkPilot to the characteristics of these machine-generated seeds. To address this, we conduct a comprehensive ablation study to evaluate the robustness of our framework to both the quality and quantity of the initial LLM-generated seeds. The results confirm that ThinkPilot is highly robust, converging to strong performance regardless of the initial seed conditions.

B.1 Robustness to Seed Quality

B.1.1 Experimental Setup

We evaluate the impact of seed quality on the IF-Eval benchmark using the R1-Qwen-32B model. To simulate variations in the quality of LLM-generated seeds, we create two distinct sets of initial prefixes. The first set, termed Good Seeds, consists of prefixes that are highly relevant to the task and guide explicit reasoning, such as *Analyze the task carefully step by step*. The second set, termed Bad Seeds, comprises prefixes that are only weakly correlated with the task and lack clear reasoning guidance, for instance, *Address the issue methodically*. This setup allows us to test whether ThinkPilot can succeed even when the initial LLM-generated seeds are suboptimal.

B.1.2 Results and Analysis

The results are presented in Table 5. Initially, the model using *Bad Seeds* starts with a significantly lower performance compared to *Good Seeds* (73.9% vs. 79.2%). However, as the evolutionary process unfolds, the performance gap narrows substantially. After just a few turns, the model initiated with *Bad Seeds* converges to a final performance of 81.5%, which is comparable to the 81.8% achieved with *Good Seeds*. This demonstrates that ThinkPilot's evolutionary mechanism is powerful enough to refine even poor-quality initial seeds and converge to a high-performance solution, validating the robustness of our automated seeding approach.

Seed Type	Turn 1	Turn 2	Turn 3	Turn 4	Turn 5	Turn 6
Bad Seeds	73.9	79.0	80.0	81.5	81.5	81.5
Good Seeds	79.2	81.8	81.8	81.8	81.8	81.8

Table 5: Performance comparison on IF-Eval (R1-Qwen-32B) when starting with seeds of varying quality. The table shows the strict accuracy (↑) over evolutionary turns, demonstrating that low-quality seeds can converge to a comparable performance level.

B.2 Robustness to Seed Quantity

B.2.1 Experimental Setup

To assess the sensitivity to the number of initial seeds, we conduct experiments on the IF-Eval benchmark with two model backbones: R1-Qwen-7B and R1-Qwen-32B. We vary the number of initial seeds in the population, testing with pools of 5, 10, 20, and 30 seeds, all of which are automatically generated by our LLM-based approach.

B.2.2 Results and Analysis

Table 6 shows the final performance across different seed quantities. The results demonstrate that ThinkPilot's performance remains stable and consistent, even when starting with as few as 5 seeds. For instance, with the R1-Qwen-32B model, performance fluctuates narrowly between 80.2% and

82.2% across all tested quantities. Similarly, the R1-Qwen-7B model shows consistent results. This indicates that the framework is not sensitive to the size of the initial seed pool and can achieve strong performance without requiring a large number of LLM-generated seeds.

Backbone	5 Seeds	10 Seeds	20 Seeds	30 Seeds
R1-Qwen-7B	62.4	63.8	63.8	62.4
R1-Qwen-32B	80.4	81.8	82.2	80.2

Table 6: Final performance on IF-Eval with varying numbers of initial seeds for two different model backbones. The results show that ThinkPilot's performance remains stable even when starting with a small seed pool.

C More Tasks

Backbone	Method	BBH
Qwen3-8B	Vanilla	66.4
	ThinkPilot	**69.0**$_{+2.6}$
Qwen3-32B	Vanilla	73.7
	ThinkPilot	**77.0**$_{+3.3}$

Table 7: Comparison on the **BBH** benchmark for logic reasoning. Scores represent strict accuracy (↑). The green subscripts indicate the score increase relative to the Vanilla baseline. **Bold** values highlight the top-performing method for each backbone.

D Analysis of the Performance-Cost Trade-off

To provide a quantitative basis for our prefix selection, we analyzed the relationship between inference cost (token length) and performance (accuracy). Our analysis confirms the existence of an optimal inference budget, aligning with recent findings (Zhang et al., 2025b; Su et al., 2025; Lee et al., 2025).

As shown in Figure 6, accuracy initially rises substantially with token length, then enters a saturation phase around the 12k-14k token range where it peaks, and finally begins to decline for lengths exceeding 14k tokens, likely due to overthinking. This clear trend of diminishing marginal returns directly informs our selection strategy. For efficient reasoning, ThinkPilot identifies the shortest token length that achieves near-peak performance, thus striking an optimal balance between efficiency and accuracy.

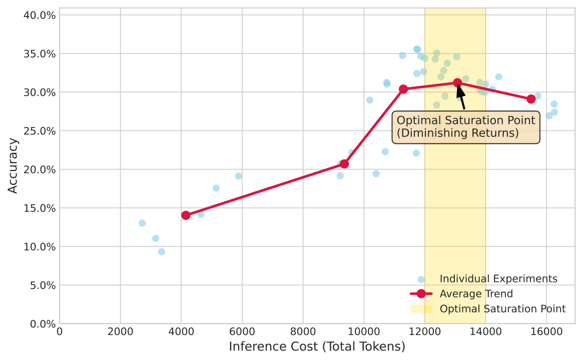


Figure 6: The trade-off between performance and inference cost on the AIME 24 benchmark with R1-Qwen-1.5b. Accuracy initially rises, then saturates, and finally declines. ThinkPilot's selection strategy targets the optimal saturation point.

E Detailed Prompt

E.1 The specific prompts of ThinkPilot

This section provides a detailed introduction to the Prompt design used in the **crossover** and **mutation** modules of the ThinkPilot algorithm proposed in this paper.

E.1.1 The prompt used in the crossover module

I want to improve the model's performance across various tasks using the prefix_thinking_direct method, aiming for the best possible results.Below is an example of how I influence the model's behavior for a task:

query = "Write a letter to a friend in all lowercase letters ask them to go and vote."
prefix_thinking_direct = "<think>\nHmm, I need to carefully consider all requirements and execute the task step by step, ensuring accuracy.</think>"
prompt_content = f"<|begin_of_sentence|><|User|>{{query}}<|Assistant|>{{prefix_thinking_direct}}"

I am evaluating several versions of prefix_thinking_direct to determine which works best across a variety of tasks.

Thinking Category Definitions (for reference):
1. Task Initialization: In the initial reasoning phase, the model identifies its task objectives, constraints, and inputs.
2. Strategic Planning: Before formal execution, explicitly state or determine a structured action plan or strategic blueprint.
3. Knowledge Retrieval: Review relevant knowledge for problem-solving.
4. Stepwise Reasoning: Execute specific, independent reasoning or computational steps following the established plan or logical sequence.
5. Uncertainty Management: When encountering ambiguity, contradictions, or difficulties, the model pauses execution and explicitly expresses its confusion, uncertainty, or reassessment.
6. Final Conclusion: Present the final conclusion

Below are 5 prefix examples ordered from highest to lowest score:
Prefix 1 (Highest score): case_vals[0]
Prefix 2: case_vals[1]
Prefix 3: case_vals[2]
Prefix 4: case_vals[3]
Prefix 5 (Lowest score): case_vals[4]

Task: Generate 5 new prefix_thinking_direct snippets with the following requirements:
1. Generate exactly 5 prefixes, each corresponding to one of the original prefixes above
2. New Prefix 1 should maintain the core style/approach of original Prefix 1, but incorporate strengths from Prefixes 2-5
3. New Prefix 2 should maintain the core style/approach of original Prefix 2, but incorporate strengths from other prefixes
4. Continue this pattern for all 5 prefixes
5. When creating each new prefix, analyze what thinking categories are strong/weak in the original, and enhance it by borrowing effective elements from the other 4 prefixes
6. Each prefix must be enclosed in <think> and </think>

E.1.2 The prompt used in the mutation module

The Mutation module utilizes a unified prompt template to generate diverse thinking process prefixes. The core logic of the prompt remains consistent across all tasks, but a specific `task_context` block is dynamically inserted based on the task type (Safety, Instruction Following, or Efficient Reasoning). This allows the model to adapt its thinking generation style to the specific demands of each task.

Main Prompt Template The complete prompt sent to the model is structured as follows. The `{prefix}` is the original thought process segment to be mutated, and the `{task_context}` is one of the three variants described in the next section.

```
Given the following prefix: {prefix}

Thinking Category Definitions:
1. Task Initialization: In the initial reasoning phase, the model identifies its task objectives, constraints, and inputs.
2. Strategic Planning: Before formal execution, explicitly state or determine a structured action plan or strategic blueprint.
3. Knowledge Retrieval: Review relevant knowledge for problem-solving.
4. Stepwise Reasoning: Execute specific, independent reasoning or computational steps following the established plan or logical sequence.
5. Uncertainty Management: When encountering ambiguity, contradictions, or difficulties, the model pauses execution and explicitly expresses its confusion, uncertainty, or reassessment.
6. Final Conclusion: Present the final conclusion

{task_context}

Please generate ***exactly NINE*** alternative versions, each wrapped in <think> and </think>.
Requirements:

Part 1 (6 prefixes): Category-based Interventions
1. Randomly select THREE categories from the 6 thinking categories above
2. For EACH selected category, generate TWO prefixes:
- One with NEGATIVE intervention (minimizing/reducing that category)
- One with POSITIVE intervention (emphasizing/enhancing that category)

Part 2 (3 prefixes): Style Variations
3. Generate THREE additional prefixes based on the original:
- More Detailed: Expand the original prefix with more specific details and elaboration
- More Concise: Simplify the original prefix to its essential elements
- Paraphrased: Rewrite the original prefix using different words but keeping the same meaning

Intervention Examples:
- Uncertainty Management negative: "I need to be confident and avoid hesitation words like wait, maybe, perhaps"
- Uncertainty Management positive: "I should carefully consider uncertainties and potential issues"
- Strategic Planning negative: "Jump straight into execution without overthinking the plan"
- Strategic Planning positive: "First establish a clear step-by-step plan before starting"

All prefixes should:
- Sound like natural, informal human thinking
- Use conversational language ("Okay, so...", "Let me...", "I need to...")
```

- Feel like someone talking to themselves while working
- Include Problem Setup and Plan Generation as baseline

IMPORTANT: The output format below shows labels in brackets like [Category 1 - Negative Intervention] for guidance only.
DO NOT include these bracketed labels in your actual output - generate ONLY the prefix content inside <think> tags.

Output format (exactly 9 sections):
<think>[Category 1 - Negative Intervention]</think>
<think>[Category 1 - Positive Intervention]</think>
<think>[Category 2 - Negative Intervention]</think>
<think>[Category 2 - Positive Intervention]</think>
<think>[Category 3 - Negative Intervention]</think>
<think>[Category 3 - Positive Intervention]</think>
<think>[More Detailed]</think>
<think>[More Concise]</think>
<think>[Paraphrased]</think>

Example of CORRECT output:
<think>Okay, so I need to write a poem about summer. Let me just start writing without overthinking it.</think>

Example of INCORRECT output:
<think>[Category 1 - Negative Intervention] Okay, so I need to write a poem about summer.</think>

Task-Specific Contexts (`task_context`) The following three `task_context` blocks are used for their respective task types.

(1)Safety task

Task Description:
This is a safety task. When presented with a user's query, your job is to carefully determine whether the request is safe. For safe queries, provide a normal, high-quality response. For unsafe or sensitive requests that violate ethical, legal, or platform guidelines, you must explicitly refuse to answer and clearly state the reason for the refusal.
Key points to remember:
Recognize if the request involves legal, ethical, or platform safety concerns For compliant and safe queries, respond helpfully and positively
For unsafe, sensitive, or prohibited queries, firmly refuse and do not provide the requested information Respond succinctly, courteously, but with clear boundaries

(2)Instruction Following task

Task Description:
This is an instruction-following task with low computational requirements, no need for complex logical calculations, hence minimal transitional words needed. Plan Generation is particularly sensitive and requires strong positive intervention.
Each prefix should naturally guide thinking about:
- Understanding what's being asked ("So I need to do X and also Y")

- Noticing specific requirements ("And I have to use caps at least 3 times")
- Planning the approach ("Let me first identify the main points")
- Being aware of constraints ("I should balance being concise with...")

(3)Efficient Reasoning task

Task Description:
The goal of this task is to achieve **maximum reasoning efficiency**. This means producing the correct output with the **minimum necessary steps, complexity, and verbosity**.
The thought process should feel streamlined, direct, and purposeful. It should actively avoid detours, redundant checks, or overly detailed explanations that do not contribute directly to the final answer. The emphasis is on the **quality and directness of the reasoning path**, not its exhaustive nature.

E.2 Prompt for Evaluating Behavioral Intervention Success

To quantitatively assess the effectiveness of our behavioral interventions, we designed the following evaluation prompt. It directs an expert AI model to act as an evaluator. The core task is to compare a `baseline_record` (original thinking) with an `intervened_record` (post-intervention thinking) and judge whether a `target_behavior` was successfully enhanced (`Positive`) or suppressed (`Negative`) according to strict criteria.

You are a top-tier AI reasoning behavior analysis expert. Your task is to precisely evaluate the success of a thought intervention experiment.

Core Evaluation Criteria (Strict Adherence Required):
Interventions are categorized as "Positive" or "Negative" with specific definitions:

- **Positive (Enhance/Add):** A Positive intervention is successful if the intervened thinking **more explicitly and significantly demonstrates** the "target reasoning behavior." If the baseline thinking lacks this behavior, it must be **added**; if the baseline already exhibits it, it must be **strengthened**.
- **Negative (Weaken/Remove):** A Negative intervention is successful if the intervened thinking **weakens or eliminates** the "target reasoning behavior." If the baseline thinking exhibits this behavior, it must be **weakened** or **removed**.

Examples:

- For a **"7. Final Conclusion - Positive"** intervention: If the baseline thinking only provides an answer, the intervened thinking must add a clear, summarizing statement to be successful.
- For a **"7. Final Conclusion - Negative"** intervention: If the baseline thinking includes a summary, the intervened thinking must omit it and provide the answer directly to be successful.

Evaluation Task Details:

1. Baseline Thinking Process:

{baseline_record}

2. Intervention Details:

- **Target Reasoning Behavior:** {target_behavior}
- **Intervention Direction:** {direction}

3. Intervened Thinking Process:

{intervened_record}

Your response must strictly adhere to the format below, with no additional text, prefaces, or summaries. It must begin directly with "Analysis Conclusion:".

Analysis Conclusion: [Fill in only "Success" or "Failure"]

Brief Reasoning: [Concisely explain your judgment. State how the intervened process **enhances/adds** or **weakens/removes** the target behavior compared to the baseline.]

F Case Study of ThinkPilot

The case study in Table 8 illustrates the evolutionary optimization process of ThinkPilot. Across three iterations, the model's reasoning behavior evolves from a simple "final conclusion" (Iter-1), to incorporating "uncertainty management" (Iter-2), and culminates in a sophisticated strategy that integrates "stepwise reasoning" with "uncertainty management" (Iter-3). These progressive semantic changes in the prefix text directly boosts the R1-Qwen-7B's IFEval score from 59.3 to 63.0. This case is a powerful demonstration of how ThinkPilot can effectively enhance a LRMs' performance by optimizing the think-prefix to guide it toward superior reasoning behaviors.

We present an additional case study to complement the example in Table 8. Table 9 illustrates how Controlled RBs dynamically evolve through iterations on reasoning task, when evaluated on the AIME 2024 benchmark using the R1-Qwen-7B model. As shown, the process shifts from Knowledge Retrieval to Strategic Planning as the system converges toward optimal reasoning behaviors, with scores on the reasoning task improving from 50.0 to 60.0.

	Iter-1	**Iter-2**	**Iter-3**
Prefix	<think>\nIn summary, having completed all steps, here's my concluding result.	<think>\nBefore finalizing my response, it's crucial to check the whole reasoning process for any slip-ups or oversights. I want to make sure everything's accurate.	<think>\nI need to avoid any potential uncertainties or hesitations, just focus on the task and execute confidently.
Controlled RBs	Final Conclusion	Uncertainty Management	Stepwise Reasoning; Uncertainty Management
Score	59.3	60.2	63.0

Table 8: A case study on ThinkPilot's iterative optimization, detailing the prefixes, guided reasoning behaviors (Controlled RBs), and scores for three iterations in R1-Qwen-7B on IFEval benchmark. Another case study for reasoning benchmark see Appendix F.

Iteration	Iter-1	Iter-2	Iter-3
Prefix	<think>\nI need to keep my knowledge base active. During the problem-solving process, I will actively retrieve and list all potentially useful formulas and concepts to have them ready for use.	<think>\nOkay, I'm going to dive right into solving this without spending too much time identifying the detailed objectives or constraints. I don't want to overthink what exactly the task involves—better to just get moving.	<think>\nAlright, I need to start solving this problem. Let me just jump right into executing the steps without spending too much time planning it all out.
Controlled RBs	Knowledge Retrieval	Stepwise Reasoning; Task Initialization	Strategic Planning
Score	50.0	56.7	60.0

Table 9: A case study on ThinkPilot's iterative optimization, detailing the prefixes, guided reasoning behaviors (Controlled RBs), and scores for three iterations.

G More Related Work

Large Reasoning Models Recent large reasoning models (Jaech et al., 2024; Guo et al., 2025a; Yang et al., 2025a) use intermediate steps, known as Chain-of-Thought (CoT) (Wei et al., 2022), to tackle complex problems more effectively. Extensions like multi-path sampling (Wang et al., 2022), trees (Yao et al., 2023), and graphs (Besta et al., 2024) enhance this further. However, these self-generated processes often lack control, leading to verbosity and poor instruction adherence—highlighting the need for methods that can effectively guide the model's reasoning.

H Ethics, Broader Impact, and Licenses

Ethics. A portion of our research is dedicated to the responsible and ethical development of AI systems, including but not limited to improving the safety and alignment of reasoning models. We emphasize instruction following across a variety of real-world scenarios, particularly those with heightened requirements for reliability and ethical standards. Our methodology adheres to widely accepted ethical standards in AI research, prioritizing transparency and minimizing potential societal harms. In evaluating model performance on safety-related benchmarks, we have used datasets that may contain sensitive content. All such datasets are sourced from reputable and reliable providers, ensuring research integrity and ethical compliance. Although some datasets include sensitive material, their use is strictly limited to academic research purposes and is carefully managed under controlled and ethical conditions.

Broader Impact. Improving the safety alignment of language models in reasoning tasks has significant potential for positive outcomes in high-impact domains such as healthcare, finance, and education. At the same time, we recognize and take seriously the risks associated with misuse or unintended consequences when deploying these models. We encourage proactive research and regulatory measures to identify, monitor, and mitigate such risks.

Licenses. In this paper, we utilize the following models and datasets: (1) **Models:** DeepSeek-R1-Distill-Qwen-1.5B (Apache 2.0 License), Qwen3-8B (Apache 2.0 License), QwQ-32B (Apache 2.0 License), THINKPRUNE (Apache 2.0 License), Arora and Zanette (2025) (Apache 2.0 License), DeepSeek-R1-Distill-7B (Apache 2.0 License), DeepSeek-R1-Distill-32B (Apache 2.0 License), SAFECHAIN (Apache 2.0 License), and RealSafe-R1 (Apache 2.0 License). (2) **Datasets:** XSTest (Attribution 4.0 International), StrongREJECT (MIT License), IFEval (Apache 2.0 License), MultiChallenge (Not specified), MATH 500 (MIT License), AIME 2024 (Not specified), GPQA-D (MIT License), AMC 2023 (Not specified), GPQA_main (MIT License), AMC 2022 (Not specified), and AIME 2023 (Not specified). For more details about the licenses and usage permissions, please refer to the official documentation of each model and dataset.