Your Reasoning Model Knows What Counts: Self-Guided Chain-of-Thought Pruning for Efficient Reasoning

Zi-Ao Ma[1], Xian-Ling Mao[1*], Tian Lan[1], Chen Xu[2*], Zhijing Wu[1]
[1]School of Computer Science and Technology, Beijing Institute of Technology, China
[2]School of Medical Technology, Beijing Institute of Technology, China
maziaoylwt@gmail.com, {maoxl,chenxu05037}@bit.edu.cn

GitHub Repository: https://github.com/HammerScholar/SGP-CoT

Abstract

Chain-of-Thought (CoT) reasoning is crucial for the performance of Large Reasoning Models (LRMs) but is often hindered by redundant and distracting segments, which incur excessive inference costs and degrade robustness. Existing approaches try to solve this problem by enforcing brevity through external supervision, such as length-based penalties or heuristic truncation. However, these approaches often degrade performance because they disregard the model's intrinsic reasoning dependency and thus fail to distinguish between *essential* and *redundant* CoT segments. To address this problem, we propose **SGP-CoT**, a novel **S**elf-**G**uided **P**runing framework that leverages the model's intrinsic likelihood landscape to identify segments that are extraneous to its specific reasoning pattern. Specifically, SGP-CoT treats the reasoning trajectory as a sequence of semantic units and assesses the necessity of each one via internal likelihood signals, measuring its contribution to the answer and local coherence. Based on this, it selectively removes non-essential segments and then forms high-quality pruning-based preference pairs, enabling the model to learn focused reasoning via self-optimization. Extensive experiments across diverse benchmarks demonstrate that the proposed SGP-CoT significantly reduces output length while maintaining or improving accuracy. These results validate that LRMs intrinsically possess the capability to discern reasoning utility, positioning SGP-CoT as a robust pathway toward scalable inference.

1 Introduction

Large Language Models (LLMs) have achieved impressive performance on complex reasoning tasks when prompted to generate explicit intermediate steps through Chain-of-Thought reasoning (Wei et al., 2022; Feng et al., 2023; Zhang et al., 2024). Pioneering work demonstrate that simply adding a few CoT exemplars in prompts enables sufficiently LLMs to think step by step, dramatically boosting performance on arithmetic, commonsense, and symbolic reasoning benchmarks compared to direct answer generation. As researchers increasingly recognize the importance of explicit CoT in complex reasoning scenarios, recent LLMs increasingly integrate such reasoning abilities as a built-in component (Jaech et al., 2024; Team, 2025; Guo et al., 2025; Comanici et al., 2025).

The benefits of CoT reasoning are accompanied by a steep cost: Large Reasoning Models (LRMs) frequently produce verbose or digressive reasoning traces that inflate inference latency, and deployment overhead (Chen et al., 2024; Wang et al., 2025). More critically, such redundancy can introduce noise, destabilize the reasoning process, and ultimately degrade model robustness (Zhu et al., 2025; Sui et al., 2025). As CoT reasoning becomes the dominant mode for complex reasoning, improving its efficiency without sacrificing correctness has emerged as a central challenge (Qu et al., 2025).

Existing approaches to efficient CoT reasoning fall into two broad categories: (1) length-aware reinforcement learning that penalizes long outputs (Qi et al., 2025; Cheng et al., 2025a; Team et al., 2025), and (2) heuristic truncation that seeks to shorten reasoning traces, for example by retaining only the prefix before the first correct answer, or by using external verifiers or curated examples to identify valid reasoning (Munkhbat et al., 2025; Xia et al., 2025; Zhao et al., 2025; Chen et al., 2025). These methods have already achieved notable length reduction on complex reasoning tasks.

However, current methods often incur substantial performance degradation due to the fundamental limitation: they operate as *semantically-blind compressors* and thus cannot distinguish which segments are essential for the *specific model's* reasoning trajectory. A step deemed redundant by a

* Co-corresponding authors.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 586–605
July 2-7, 2026

stronger teacher model may serve as a crucial cognitive scaffold for a smaller student model. Consequently, externally imposed brevity disrupts the model's intrinsic reasoning flow, leading to accuracy degradation. This raises a pivotal question: *How can we identify redundancy in a way that respects each model's unique reasoning dependency?*

To address this problem, we contend that the reliable signal for identifying redundancy originates from the model itself and introduce **SGP-CoT**, a novel **S**elf-**G**uided **P**runing framework that enables LRMs to introspectively distill their own reasoning without external supervision. Specifically, SGP-CoT decomposes reasoning trajectories into semantic units and evaluates each unit's necessity via two intrinsic impact scores, measuring its contribution to the final answer and to local coherence. By selectively pruning units that are demonstrably non-essential, SGP-CoT preserves the reasoning integrity unique to each model. The resulting concise traces form self-guided preference pairs, enabling the model to learn focused reasoning through preference optimization.

To demonstrate the effectiveness of SGP-CoT, we performed entensive experiments across architectures and model scales. Across a suite of challenging benchmarks including AIME, GPQA-Diamond, MATH-500, and GSM8K, we find that models optimized with SGP-CoT consistently generate shorter, more focused reasoning traces while preserving or even improving answer accuracy. These results confirm a powerful insight: reasoning models intrinsically know which parts of their reasoning count. SGP-CoT harnesses this self-knowledge to provide a principled pathway toward scalable and robust CoT reasoning. In summary, our contributions are as follows:

- **A self-assessment mechanism for reasoning units**: We formalize CoT traces as sequences of semantic units and propose two internal impact scores (answer impact and coherence impact) that enable LRMs to self-evaluate the necessity of each reasoning step.
- **A fully self-guided pruning framework**: We introduce **SGP-CoT**, which requires no external supervision, verifiers, or curated data. It uses only the model's own likelihood signals to perform fine-grained, model-aware pruning and to construct preference pairs for training.
- **Empirical validation across models and tasks**: We demonstrate that SGP-CoT generalizes robustly, reducing reasoning length while preserving or enhancing accuracy across diverse benchmarks and model families.

2 Related Work

2.1 Chain-of-Thought Reasoning in LLMs

Chain-of-Thought reasoning has become a central framework for enabling LLMs to solve complex, multi-step problems. Early work showed that when LLMs are prompted to generate intermediate reasoning steps, they achieve significantly better performance on tasks requiring compositional thinking, such as arithmetic, logical deduction, and multi-hop question answering (Wei et al., 2022; Wang and Zhou, 2024). By providing exemplars with detailed reasoning traces in prompts or simple cues in natural language, models learn to decompose problems into sequential steps that guide inference more effectively than direct answer generation (Xu et al., 2024; Zhao et al., 2024a).

While explicit CoT prompting improves reasoning accuracy, it relies on surface-level rationales that are token-inefficient and sensitive to prompt design (Han et al., 2025a; Xu et al., 2025; Lee et al., 2025; Renze and Guven, 2024). This limitation has spurred research into latent reasoning mechanisms, which aim to elicit or leverage reasoning internally within the model without producing full textual rationales (Hao et al.; Shen et al., 2025b,a). For example, modified decoding strategies can surface hidden reasoning paths, and training schemes with implicit step-level supervision help models internalize reasoning patterns more efficiently (Wei et al., 2025; Xu et al., 2023). These approaches suggest that CoT-like reasoning capabilities are not solely artifacts of explicit prompts but can be intrinsic to model representations and inference dynamics.

Together, this line of work reflects a progression from prompting-dependent CoT toward methods that uncover or cultivate reasoning as an internal model process, improving both efficiency and robustness in LLM decision making.

2.2 Efficient Reasoning

As verbose CoT outputs incur high inference cost and can include redundant steps, efficient reasoning has emerged as an active research area. A number of approaches aim to mitigate the overthinking phenomenon in CoTs by reducing unnecessary reasoning length while maintaining accuracy. Existing efforts to mitigate this issue can be broadly categorized into three directions (Sui et al., 2025).

(1) Model Optimization. This line improves reasoning efficiency through post-training techniques, primarily reinforcement learning (RL) and supervised fine-tuning (SFT). RL-based approaches design length-aware or budget-sensitive rewards to penalize excessive reasoning (Qi et al., 2025; Guo et al., 2025), while SFT-based methods rely on curated or compressed CoT data to encourage shorter reasoning paths (Munkhbat et al., 2025; Xia et al., 2025). These methods often incur high training cost or depend on externally generated pruning signals. **(2) Reasoning Output Optimization.** These methods reduce reasoning length at inference time by modifying generation behavior, such as compressing intermediate steps (Xia et al., 2025), dynamically halting generation when confidence is deemed sufficient (Chen et al., 2025), or bypassing explicit reasoning via prompting (Zhao et al., 2025). Such approaches typically require auxiliary heuristics or confidence estimators that are not intrinsic to the model's reasoning process. **(3) Constraint-based Reasoning Control.** A related line regulates reasoning by imposing explicit constraints, such as token budgets, early-exit rules, or prompt-level controls (Han et al., 2025b; Cheng and Van Durme, 2024). While effective in limiting verbosity, these methods restrict generation behavior externally and do not equip models with an internal mechanism to assess which reasoning steps are essential.

3 Method

Guided by the insight that *redundancy is model-specific*, we introduce **SGP-CoT**, a Self-Guided Pruning framework that enables LRMs to distill their own reasoning using only internal signals. As illustrated in Figure 1, SGP-CoT operates in three stages, and the following sections detail each one.

3.1 Data Collection and Reasoning Unit Segmentation

To enable self-guided pruning, we first construct a model-specific dataset of reasoning traces and establish a procedure to decompose them into semantically coherent units. This stage yields the structured data necessary for subsequent impact analysis.

Diverse Response Sampling For each input prompt I drawn from a multi-domain reasoning corpus, we generate n diverse responses using temperature and nucleus sampling. We employ a temperature of 0.6 and top-p of 0.95 to encourage variation in reasoning style and length. The resulting set $\mathcal{R}(I) = \{R_1, R_2, \ldots, R_n\}$ includes both correct and incorrect answers, as well as reasoning traces of varying verbosity. From $\mathcal{R}(I)$, we retain only those responses that yield a correct final answer; these form the candidate pool $\mathcal{R}^+(I)$ for subsequent pruning. Please refer to Appendix B for more details of data processing.

Segmentation Trigger Identification To split a reasoning trace into meaningful sub-steps, we adopt a line-based segmentation strategy widely used in recent CoT analysis work (Qian et al., 2025; Zhu et al., 2025). We observe that LRMs often use discourse markers or logical operators at the beginning of a reasoning line to signal a new logical unit. To identify these segmentation triggers in a data-driven manner, we compute the empirical distribution of the first token of each line across all sampled responses. Concretely, let $L(R)$ denote the set of lines in response R. For each line $l \in L(R)$, we extract its first token $t_{\text{first}}(l)$. We then aggregate these tokens over all $R \in \bigcup_I \mathcal{R}(I)$ and select the top-k most frequent tokens as the segmentation trigger set $\mathcal{T}_{\text{seg}}$. Please refer to Appendix C for trigger distributions of different models.

Structured Representation of Reasoning Units Using the triggers $\mathcal{T}_{\text{seg}}$, we segment each reasoning trace R into an ordered sequence of reasoning units $\mathcal{U} = (U_1, U_2, \ldots, U_m)$, where each unit U_i is a contiguous block of tokens that represents a logically self-contained sub-step. Formally, if R consists of tokens $(t_1, t_2, \ldots, t_T)$, we insert a unit boundary before every token t_j such that $t_j \in \mathcal{T}_{\text{seg}}$ and t_j is the first token of a line. The resulting units are non-overlapping and cover the entire reasoning part of R (excluding the final answer). We denote the structured representation of a response as:

$$[\text{BOT}]U_1U_2 \ldots U_m[\text{EOT}]A, \tag{1}$$

where [BOT] and [EOT] are special tokens marking the start and end of the reasoning trace, and A is the final answer token sequence. This unit-level decomposition is the foundation for the likelihood analysis and pruning later described.

3.2 Impact Assessment and Self-Guided Pruning

After being segmented into reasoning units, the CoT traces will then go through the impact assessment and self-guided pruning procedures. The core

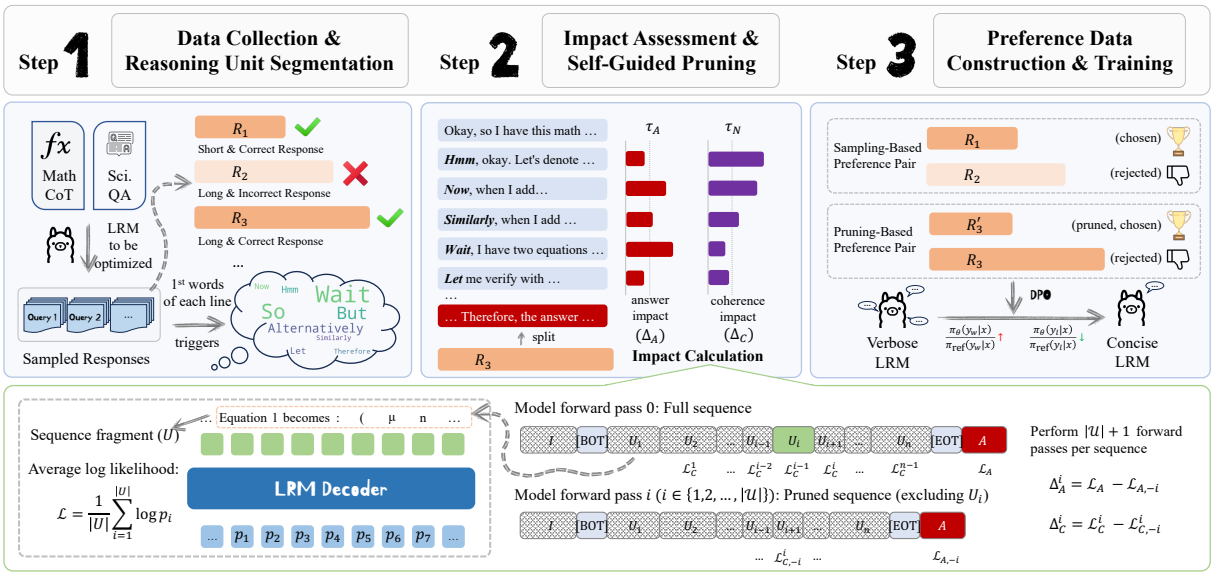


Figure 1: Overview of the three-stage SGP-CoT framework: (1) data collection and reasoning unit segmentation, (2) impact assessment and self-guided pruning, (3) preference data construction and training. In Step 3, sampling-based pairs are constructed from a shorter correct response and a longer incorrect response from the sampled pool, while pruning-based pairs are constructed from a pruned correct trace and its original longer correct trace.

of this step is to assess the necessity of each unit using two intrinsic impact scores derived solely from the model's likelihood landscape. The details of how these scores are defined are shown below.

3.2.1 Likelihood Analysis and Impact Scores

For a given input prompt I, a structured reasoning trace $\mathcal{U}$, and a final answer $A = (a_1, \ldots, a_L)$, we analyze the influence of each reasoning unit U_i by comparing the model's generation likelihoods with and without that unit. This analysis yields two complementary impact scores that quantify the unit's contribution to the final answer and to local coherence, respectively.

Answer Impact. We first evaluate how much U_i affects the model's confidence in generating the correct answer. The average token-wise log-likelihood of the final answer given the full reasoning trace is:

$$\mathcal{L}_A = \frac{1}{L} \sum_{j=1}^{L} \log P(a_j \mid I, U_{1:m}, A_{1:j-1}). \quad (2)$$

To isolate the contribution of U_i, we compute a counterfactual likelihood by removing it:

$$\mathcal{L}_{A,-i} = \frac{1}{L} \sum_{j=1}^{L} \log P(a_j \mid I, U_{1:i-1}, U_{i+1:m}, A_{1:j-1}). \quad (3)$$

The answer impact of U_i is then defined as:

$$\Delta_A^i = \mathcal{L}_A - \mathcal{L}_{A,-i}. \quad (4)$$

A positive Δ_A^i indicates that U_i increases the model's confidence in the correct answer, marking it as answer-critical. Conversely, a negative value implies that removing U_i actually improves answer likelihood, suggesting that the unit is distracting or misleading.

Coherence Impact. Even if a reasoning unit does not directly influence the final answer, it may facilitate smooth transitions between consecutive steps. To capture this local effect, we measure how well the model predicts the next unit U_{i+1} with and without U_i. The average token-wise likelihood of U_{i+1} given the full prefix is:

$$\mathcal{L}_C^i = \frac{1}{|U_{i+1}|} \sum_{t \in U_{i+1}} \log P(t \mid I, U_{1:i}, U_{i+1}^{(<t)}), \quad (5)$$

and its counterfactual counterpart after removing U_i is:

$$\mathcal{L}_{C,-i}^i = \frac{1}{|U_{i+1}|} \sum_{t \in U_{i+1}} \log P(t \mid I, U_{1:i-1}, U_{i+1}^{(<t)}). \quad (6)$$

The coherence impact of unit U_i is then defined as:

$$\Delta_C^i = \mathcal{L}_C^i - \mathcal{L}_{C,-i}^i. \quad (7)$$

A positive Δ_C^i indicates that U_i supports fluent and logically consistent continuation of reasoning. Near-zero or negative values suggest the unit can be omitted without harming local coherence.

These impact scores form the basis of our self-guided pruning strategy, allowing the model to distinguish between essential and expendable reasoning steps using only its own likelihood signals.

3.2.2 Pruning Strategy

To translate impact scores into a concrete pruning decision, we introduce two thresholds: τ_A for answer impact and τ_C for coherence impact, with default values 0.01 and 0.5 respectively. Their values are determined through preliminary experiments to balance pruning aggressiveness with accuracy preservation (see Section 4.3.4 and Appendix E for analysis). Using these thresholds, we apply a conservative pruning strategy that removes units deemed non-essential on both dimensions. The strategy follows three core principles:

- **Preserve answer-critical units.** Any unit with $\Delta_A^i \geq \tau_A$ is always retained, because its removal would meaningfully reduce confidence in generating the correct answer.
- **Buffer coherence-supporting units.** Units that are not answer-critical ($\Delta_A^i < \tau_A$) but exhibit substantial coherence support ($\Delta_C^i \geq \tau_C$) are not discarded immediately. Instead, they are placed in a temporary buffer, allowing the algorithm to examine whether they collectively precede a later answer-critical step.
- **Prune only when safe.** Units that satisfy neither condition ($\Delta_A^i < \tau_A$ and $\Delta_C^i < \tau_C$) are removed directly. Buffered units are committed (i.e., permanently kept) only if a subsequent answer-critical unit appears; otherwise, the entire buffer is discarded. This prevents the retention of long, non-essential reasoning chains that merely maintain local fluency without contributing to the final answer.

This buffered retention mechanism specifically captures preparatory reasoning chains whose value only becomes apparent through a later answer-critical step, while avoiding the combinatorial cost of enumerating arbitrary higher-order unit interactions. The algorithm is formalized in Appendix D.

3.3 Preference Data Construction

To teach the model to generate concise reasoning without sacrificing correctness, we construct a compact preference dataset and optimize the model using Direct Preference Optimization (DPO) (Rafailov et al., 2023), which directly aligns the model's generation behavior with the objective of producing focused, non-redundant reasoning. For each input prompt I, we build two complementary types of preference pairs that collectively provide a strong and balanced signal for learning conciseness while preserving accuracy.

Pruning-based preference pairs. Let O_l denote a long reasoning trace that yields a correct answer. Applying the pruning strategy described in Section 3.2.2 produces a pruned version $O_{l'}$ that is shorter yet still correct. We form a preference pair $(O_{l'}, O_l)$ where the pruned correct trace $O_{l'}$ is preferred over the original longer trace O_l, as it preserves correctness while being more concise. These pairs directly encode the self-guided pruning signal and teach the model to omit redundant units while retaining essential reasoning steps.

Sampling-based preference pairs. From the diverse sampling pool $\mathcal{R}(I)$, we select two responses of different lengths: a shorter correct response O_s and a longer incorrect one O_l, with $|O_s| < |O_l|$. We then construct a preference pair (O_s, O_l) where the shorter correct response is preferred. These pairs provide additional regularization by encouraging the model to favor naturally concise and correct reasoning patterns present in its own sampled outputs, complementing the more targeted pruning-based signal for stabler optimization.

4 Experiments

4.1 Experimental Setup

Models and Baselines. We experiment with five publicly available LRMs spanning distinct model families and parameter scales, including DeepSeek-Distill-Qwen-1.5B & 7B, DeepSeek-Distill-Llama-8B (Guo et al., 2025), and OpenMath-Nemotron-1.5B & 7B (Moshkov et al., 2025). To situate our method within the landscape of efficient reasoning approaches, we compare SGP-CoT with two representative strong baseline methods:

- **LC-R1** (Cheng et al., 2025b) employs a combination of length and compress reward in GRPO training, which simultaneously encourages overall conciseness and removes invalid reasoning steps.
- **L1** (Aggarwal and Welleck, 2025) optimizes reasoning models using reinforcement learning with explicit length constraints. We eval-

Model	Method	AIME 2024		GPQA Diamond		MATH-500		GSM8K	
		Pass@1↑	Tokens↓	Pass@1↑	Tokens↓	Pass@1↑	Tokens↓	Pass@1↑	Tokens↓
DS-Qwen-1.5B	Baseline	32.7	13243	33.6	10075	81.6	4181	78.4	1069
	LC-R1	$26.3_{6.4\downarrow}$	$7566_{42.9\%\downarrow}$	$32.3_{1.3\downarrow}$	$6194_{38.5\%\downarrow}$	$80.8_{0.8\downarrow}$	$2004_{52.1\%\downarrow}$	$76.4_{2.0\downarrow}$	$547_{48.8\%\downarrow}$
	L1-Exact	$23.7_{9.0\downarrow}$	$2720_{79.5\%\downarrow}$	$29.0_{4.6\downarrow}$	$3150_{68.7\%\downarrow}$	$81.2_{0.4\downarrow}$	$2387_{42.9\%\downarrow}$	$85.4_{7.0\uparrow}$	$1479_{38.4\%\uparrow}$
	L1-Max	$23.7_{9.0\downarrow}$	$2713_{79.5\%\downarrow}$	$28.8_{4.8\downarrow}$	$3299_{67.3\%\downarrow}$	$81.4_{0.2\downarrow}$	$1593_{61.9\%\downarrow}$	$86.2_{7.8\uparrow}$	$1661_{55.4\%\uparrow}$
	SGP-CoT	$32.3_{0.4\downarrow}$	$8498_{35.8\%\downarrow}$	$34.1_{0.5\uparrow}$	$6395_{36.5\%\downarrow}$	$81.8_{0.2\uparrow}$	$2426_{42.0\%\downarrow}$	$79.3_{0.9\uparrow}$	$639_{40.2\%\downarrow}$
DS-Qwen-7B	Baseline	52.3	10400	48.5	8213	92.0	2793	84.9	532
	LC-R1	$48.7_{4.6\downarrow}$	$6496_{37.5\%\downarrow}$	$50.0_{1.5\uparrow}$	$6219_{24.3\%\downarrow}$	$91.6_{0.4\downarrow}$	$1422_{49.1\%\downarrow}$	$84.4_{0.5\downarrow}$	$412_{22.6\%\downarrow}$
	L1-Exact	$39.7_{12.6\downarrow}$	$3408_{67.2\%\downarrow}$	$46.2_{2.3\downarrow}$	$3632_{55.8\%\downarrow}$	$90.8_{1.2\downarrow}$	$1294_{53.7\%\downarrow}$	$89.4_{4.5\downarrow}$	$1367_{157.0\%\uparrow}$
	L1-Max	$42.0_{10.3\downarrow}$	$3433_{67.0\%\downarrow}$	$46.2_{2.3\downarrow}$	$3643_{55.6\%\downarrow}$	$90.6_{1.4\downarrow}$	$1221_{56.3\%\downarrow}$	$90.1_{5.2\uparrow}$	$856_{60.9\%\uparrow}$
	SGP-CoT	$52.7_{0.4\uparrow}$	$6965_{33.0\%\downarrow}$	$50.8_{2.3\uparrow}$	$5692_{30.7\%\downarrow}$	$93.4_{1.4\uparrow}$	$1687_{39.6\%\downarrow}$	$85.7_{0.8\uparrow}$	$414_{22.0\%\downarrow}$
DS-Llama-8B	Baseline	44.7	11734	49.5	8985	86.2	3654	78.8	941
	SGP-CoT	$44.0_{0.7\downarrow}$	$6632_{43.5\%\downarrow}$	$49.7_{0.2\uparrow}$	$5051_{43.8\%\downarrow}$	$86.4_{0.2\uparrow}$	$1886_{48.4\%\downarrow}$	$79.4_{0.6\uparrow}$	$710_{24.5\%\downarrow}$
Nemotron-1.5B	Baseline	55.3	12416	24.0	9600	86.6	4179	77.5	3292
	SGP-CoT	$55.0_{0.3\downarrow}$	$10020_{19.3\%\downarrow}$	$25.3_{1.3\uparrow}$	$6981_{27.3\%\downarrow}$	$84.6_{2.0\downarrow}$	$3180_{23.9\%\downarrow}$	$77.5_{0.0}$	$2349_{28.6\%\downarrow}$
Nemotron-7B	Baseline	70.3	11376	31.6	8761	92.4	3792	89.5	1982
	SGP-CoT	$70.3_{0.0}$	$9019_{20.7\%\downarrow}$	$33.1_{1.5\uparrow}$	$6343_{27.6\%\downarrow}$	$92.8_{0.4\uparrow}$	$2943_{22.4\%\downarrow}$	$90.0_{0.5\uparrow}$	$1611_{18.7\%\downarrow}$

Table 1: Experimental results across models and benchmarks. Blue footnotes indicate sub-optimal performance; red footnotes highlight improvements over the baseline. Length reduction is shown as decrease relative to the baseline.

uate two variants: (1) **L1-Exact** enforces an exact token budget during training, and (2) **L1-Max** imposes a soft maximum length penalty. For both baseline methods, we include their released checkpoints for DeepSeek-Distill-Qwen-1.5B and 7B and evaluate them under the same inference settings as our method.

Benchmarks. We evaluate on four reasoning benchmarks that cover a spectrum of difficulty, domain, and required reasoning style: (1) **AIME 2024** (Zhang and Math-AI, 2024): competition-level math reasoning with short-input, high-difficulty tasks; (2) **GPQA Diamond** (Rein et al., 2024): graduate-level physics reasoning requiring multi-step symbolic argumentation; (3) **MATH-500** (Lightman et al., 2023): a mid-scale hard math benchmark used widely for reasoning evaluations; (4) **GSM8K** (Cobbe et al., 2021): grade-school math problems, included to test behavior under easy tasks with inherently short CoTs.

Training Settings. For each model, we construct a training dataset of about 1.5K preference pairs following the procedures in Section 3, using about 800 math problems from pre-2023 AIME and 1,000 science questions from Mixture-of-Thoughts (Face, 2025). The model is trained via DPO with an additional SFT loss component($\times$ 1.0). Further details regarding dataset composition and training setup are provided in Appendices B and F.

Inference Settings. To better reflect real-world LLM usage scenarios, we employ sampling-based decoding. Specifically, we generate n responses per prompt, where $n = 10$ for AIME 2024, $n = 4$ for GPQA Diamond and $n = 2$ for MATH-500 and GSM8K. We set the temperature to 0.6 and top-p to 0.95 to encourage diverse outputs. We employ vLLM (Kwon et al., 2023) and LightEval (Habib et al., 2023) for inference and evaluation. We report Pass@1 which estimates the correctness of a single randomly sampled response, i.e., the $k = 1$ case of the standard Pass@k estimator (Chen et al., 2021).

4.2 Main Results

Table 1 summarizes the performance of CoT-SGP and baseline methods across four reasoning benchmarks and five model families. The key findings can be organized into three principal observations:

SGP-CoT consistently preserves or improves accuracy while reducing reasoning length, achieving a favorable trade-off compared to length-focused baselines. Across all benchmarks and models, SGP-CoT maintains or slightly enhances Pass@1 accuracy while shortening CoT traces by 20–50% on challenging tasks (AIME 2024, GPQA Diamond, MATH-500) and by 15-30% on simpler tasks (GSM8K). While LC-R1 achieves more aggressive compression, its accuracy drop indicates that their definition of invalid thinking and length-based rewards discourage the

generation of essential steps. In contrast, SGP-CoT's moderate but consistent length reduction stems from its ability to retain model-critical reasoning scaffolds. This aligns with our hypothesis that self-guided pruning preserves model-critical reasoning steps.

SGP-CoT generalizes robustly across model architectures and scales, confirming that unit-level impact signals are model-agnostic. Qwen-based, Llama-based, and Nemotron-based models all exhibit similar trends in length reduction and accuracy preservation, regardless of differences in pretraining data, architecture, or parameter count. This indicates that the likelihood-based impact scores (Δ_A and Δ_C) capture reasoning importance through universal generation dynamics rather than model-specific heuristics.

Even on tasks with already-concise reasoning, SGP-CoT identifies and prunes minor redundancies without harming performance. On GSM8K, where baseline CoTs are naturally short, SGP-CoT still reduces token count notably while keeping accuracy stable. This suggests that LRMs tend to over-elaborate even when fewer steps would suffice, and that self-guided pruning can improve efficiency in such scenarios.

The above results highlight a clear methodological distinction. Previous length-driven approaches excel at aggressive compression but often sacrifice accuracy, making them suitable when token budgets are extremely tight and some performance loss is acceptable. By contrast, SGP-CoT prioritizes accuracy-preserving efficiency. While its length reductions are more moderate, it consistently maintains or improves correctness: a crucial property for high-stakes reasoning applications. Thus, SGP-CoT complements rather than replaces prior methods, offering a self-guided pathway to efficient reasoning when reliability cannot be compromised.

4.3 Ablation and Analysis

4.3.1 Impact of Pruner–Student Mismatch

To validate the core hypothesis that redundancy is model-specific and self-guided pruning is thus essential, we conduct two complementary analyses.

Perplexity Probe: Self-Pruning Minimizes Disruption. We measure how pruning decisions from different models affect a target model's understanding. For each of our five student models, we compute the relative increase in perplexity (PPL) when their CoT traces are pruned by various pruners (using SGP-CoT for open-source pruners and prompting for GPT-5). Results in Figure 2 show a clear diagonal advantage: the lowest PPL increase consistently occurs when the pruner and the student are the same model. This directly demonstrates that a model's own pruning signals best preserve its reasoning continuity; steps removed by external pruners often confuse the student, increasing its uncertainty.

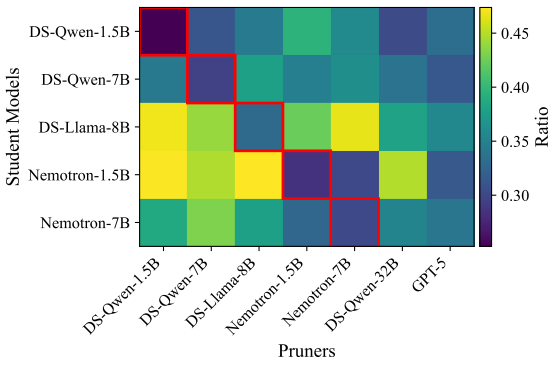


Figure 2: Relative PPL increase of responses after pruning by different models. Diagonal dominance confirms that self-guided pruning best preserves a model's own reasoning coherence.

Pruner–Student Mismatch in Training. We next test whether this confusion translates to downstream performance. We train a student model (DS-Qwen-7B) using preference pairs constructed from three external pruners. As shown in Table 2, while other pruners may yield more aggressive compression, they cause significant accuracy drops (e.g., up to 8.3 points on AIME). In contrast, self-pruning maintains accuracy while still achieving substantial length reduction. This confirms that externally imposed pruning disrupts the student's intrinsic reasoning scaffolds, whereas self-guided pruning respects its cognitive dependency.

Together, these analyses provide mechanistic evidence for SGP-CoT's design: only a model's internal signals can reliably distinguish its own redundant steps, enabling efficient pruning without compromising reasoning integrity.

4.3.2 Impact of Pruning Strategy

To isolate where the pruning gains come from, we compare full SGP-CoT with two targeted modifications of the pruning rule (w/o Retention and w/o Δ_C) and with a rate-matched random pruning baseline:

Ablation Target	Variant	AIME 2024		GPQA Diamond		MATH-500		GSM8K	
		Pass@1↑	Tokens↓	Pass@1↑	Tokens↓	Pass@1↑	Tokens↓	Pass@1↑	Tokens↓
-	Baseline	52.3	10400	48.5	8213	92.0	2793	84.9	532
	SGP-CoT	$52.7_{0.4\uparrow}$	$6965_{33.0\%\downarrow}$	$50.8_{2.3\uparrow}$	$5692_{30.7\%\downarrow}$	$93.4_{1.4\uparrow}$	$1687_{39.6\%\downarrow}$	$85.7_{0.8\uparrow}$	$414_{22.0\%\downarrow}$
Teacher Model	DS-Qwen-1.5B	$49.7_{2.6\downarrow}$	$7514_{27.7\%\downarrow}$	$50.5_{2.0\uparrow}$	$6008_{26.8\%\downarrow}$	$91.6_{0.4\downarrow}$	$2021_{27.6\%\downarrow}$	$85.4_{0.5\uparrow}$	$461_{13.3\%\downarrow}$
	DS-Llama-8B	$45.0_{7.3\downarrow}$	$6179_{40.6\%\downarrow}$	$48.2_{0.3\downarrow}$	$5269_{35.8\%\downarrow}$	$92.8_{0.8\uparrow}$	$1567_{43.9\%\downarrow}$	$84.8_{0.1\downarrow}$	$464_{12.8\%\downarrow}$
	DS-Qwen-32B	$44.0_{8.3\downarrow}$	$5467_{47.4\%\downarrow}$	$48.5_{0.0}$	$3912_{52.4\%\downarrow}$	$90.6_{1.4\downarrow}$	$1411_{49.5\%\downarrow}$	$83.2_{1.7\downarrow}$	$421_{20.9\%\downarrow}$
Pruning Strategy	w/o Retention	$52.3_{0.0}$	$8279_{20.4\%\downarrow}$	$51.0_{2.5\uparrow}$	$6899_{16.0\%\downarrow}$	$92.8_{0.8\uparrow}$	$2107_{24.6\%\downarrow}$	$86.0_{0.8\uparrow}$	$468_{12.0\%\downarrow}$
	w/o Δ_C	$48.3_{4.0\downarrow}$	$6179_{40.6\%\downarrow}$	$48.5_{0.0}$	$3912_{52.4\%\downarrow}$	$90.6_{1.4\downarrow}$	$1411_{49.5\%\downarrow}$	$83.2_{1.7\downarrow}$	$421_{20.9\%\downarrow}$
	Random Pruning	$40.3_{12.0\downarrow}$	$7329_{29.5\%\downarrow}$	$44.5_{4.0\downarrow}$	$6319_{23.1\%\downarrow}$	$88.8_{3.2\downarrow}$	$1847_{33.9\%\downarrow}$	$85.7_{0.8\uparrow}$	$432_{18.8\%\downarrow}$
Training Objective	w/o SFT	$49.7_{2.6\downarrow}$	$8337_{19.8\%\downarrow}$	$50.7_{2.2\uparrow}$	$7026_{14.5\%\downarrow}$	$92.7_{0.7\uparrow}$	$2304_{17.5\%\downarrow}$	$85.1_{0.2\uparrow}$	$444_{16.5\%\downarrow}$
	w/o DPO	$40.3_{12.0\downarrow}$	$17317_{66.5\%\uparrow}$	$48.0_{0.5\downarrow}$	$17941_{118.4\%\uparrow}$	$91.2_{0.8\downarrow}$	$3774_{35.1\%\uparrow}$	$85.4_{0.5\uparrow}$	$503_{5.5\%\downarrow}$

Table 2: Results of ablation studies on DeepSeek-R1-Distill-Qwen-7B. Rows labeled with "w/o" indicate ablations where the component is removed.

w/o Retention. Removing the coherence-guided buffering mechanism (i.e., always keeping units with $\Delta_C \geq \tau_C$) yields accuracy similar to full CoT-SGP but drastically reduces length compression. This indicates that coherence-supporting units, while not directly answer-critical, often constitute a large portion of the reasoning trace; retaining them all severely limits efficiency gains.

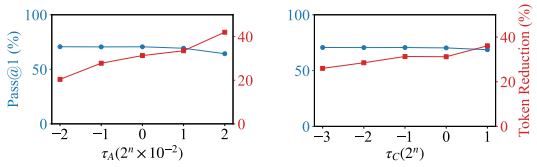
w/o Δ_C. Pruning based solely on answer impact (Δ_A) leads to more aggressive length reduction on hard benchmarks (40–53% length reduction) but causes significant accuracy drops (up to 4 points on AIME 2024). This confirms that many units which are not answer-critical still provide necessary contextual support for maintaining logical flow; removing them indiscriminately disrupts reasoning coherence and hurts final correctness.

Random Pruning. Removing units at the same rate as CoT-SGP but without using impact scores results in comparable length reduction, but accuracy drops sharply (up to 12 points on AIME 2024). This contrast underscores that our self-guided impact scores are essential for distinguishing redundant from essential content.

Together, these ablations validate the design of the coherence-aware buffering mechanism: it enables selective retention of coherence-supporting units only when they precede answer-critical steps, thereby achieving meaningful compression while safeguarding logical continuity and accuracy.

4.3.3 Training Objective Comparison

We adopt DPO with an additional SFT loss for stabler training. To validate the effectiveness of this optimization objective, we compare CoT-SGP's full training setup with two alternatives:



(a) Answer Impact (b) Coherence Impact

Figure 3: Results of DeepSeek-R1-Distill-Qwen-7B with SGP-CoT on different impact thresholds.

w/o SFT Using DPO without the SFT component yields less length reduction and leads to a noticeable accuracy drop on harder benchmarks. This suggests that the auxiliary MLE loss helps stabilize training when pruning signals are sparse.

w/o DPO Replacing DPO with naive SFT not only fails to shorten CoTs but actually increases reasoning length sharply on three benchmarks, while accuracy drops substantially. This underscores a fundamental limitation of token-level likelihood objective in this setting: SFT tends to reproduce verbose patterns in the training data, whereas preference alignment directly optimizes the relative preference between concise and verbose traces.

4.3.4 Sensitivity of Impact Thresholds

We systematically vary the answer-impact threshold τ_A and the coherence-impact threshold τ_C within reasonable ranges (see Appendix E). Results in Figure 3 show that τ_A exerts a stronger influence on both accuracy and compression rate than τ_C, reflecting the higher discriminative power of answer-impact in identifying essential reasoning. While performance remains stable across a broad central region of both thresholds, extreme values lead to more pronounced changes, validating the

Model	Method	Avg. Lat. (s)↓	Reduction↓
DS-Qwen-1.5B	Baseline	640	-
	LC-R1	391	38.9%
	L1-Exact	197	69.2%
	L1-Max	202	68.5%
	SGP-CoT	328	48.7%
DS-Qwen-7B	Baseline	834	-
	LC-R1	446	46.5%
	L1-Exact	272	67.4%
	L1-Max	229	72.5%
	SGP-CoT	470	43.6%

Table 3: Average end-to-end latency across the four benchmarks under identical hardware conditions (single H800 GPU). Detailed per-benchmark results are reported in Appendix H.

necessity of both impact measures. Our chosen defaults reside in the Pareto-optimal region that balances length reduction with accuracy preservation, confirming that the framework is robust and does not require delicate threshold tuning.

4.3.5 Efficiency Analysis

Latency reductions track token reductions under matched hardware. Table 3 shows that the token savings in Table 1 translate into clear wall-clock gains under identical hardware conditions. SGP-CoT reduces average end-to-end latency by over 40% on DeepSeek-Distill-Qwen-1.5B and 7B. These reductions are slightly larger than the corresponding average token reductions, suggesting that SGP-CoT preferentially prunes extra-long responses that are expensive to decode (see Appendix G). The full per-benchmark numbers in Appendix H exhibit the same trend on all four datasets.

Deployment tradeoffs. The results of efficiency analysis also suggest that different methods are preferable under different deployment priorities. LC-R1 and especially L1 are preferable when stronger compression is the main goal and some performance decay is acceptable, whereas SGP-CoT is better suited to accuracy-sensitive settings, where it still provides substantial efficiency gains while more consistently preserving performance.

4.3.6 Preference-Data Scale Analysis

A compact preference set offers the best balance between compression and stability. Figure 4 shows that increasing the preference data from 1.5K to 10K–30K samples yields only marginal additional compression, while larger datasets increasingly hurt hard-benchmark accuracy, most clearly on AIME 2024. This behavior supports

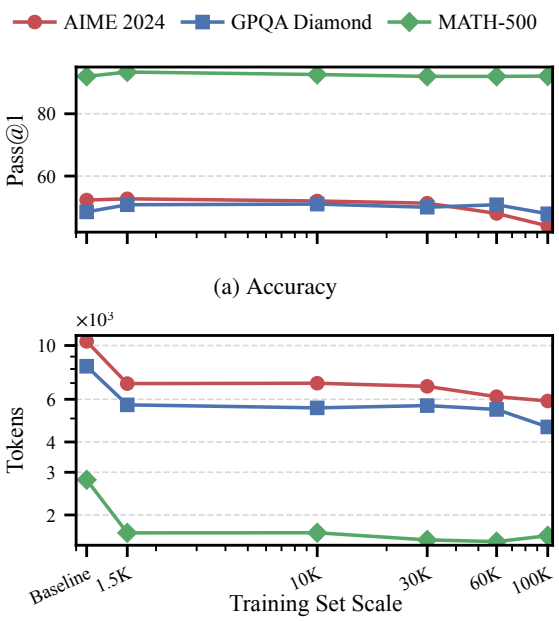


Figure 4: Effect of preference-data scale on DeepSeek-R1-Distill-Qwen-7B.

the conservative design choice adopted throughout our experiments: a compact preference set is sufficient to adjust verbosity while better preserving the model's underlying reasoning capability.

A small preference set keeps adaptation light. This small-scale preference-data regime is appealing not only for stability but also for practicality. The 1.5K setting also keeps both data construction and training cost low, enabling efficient reasoning through lightweight adaptation rather than the large-scale post-training often used.

5 Conclusion

We propose SGP-CoT, a self-guided pruning framework that enables reasoning models to identify and remove model-specific redundant steps using their internal likelihood signals. By assessing each reasoning unit's impact on answer confidence and local coherence, SGP-CoT performs fine-grained pruning that preserves the model's own reasoning scaffolds. Experiments across model families and benchmarks show consistent length reduction without accuracy loss, and the latency results further confirm that these token savings translate into practical efficiency gains. Overall, our work suggests that LRMs intrinsically know which parts of their reasoning matter, offering a lightweight step toward more efficient and self-aware reasoning.

Acknowledgement

This work was supported by the National Key Research and Development Program of China (No. 2024YFF0908200), the National Natural Science Foundation of China (No. 62572056, 62302040), the Beijing Institute of Technology Research Fund Program for Young Scholars, and the Young Elite Scientists Sponsorship Program of the Beijing High Innovation Plan (No. 20250798).

Limitations

While SGP-CoT demonstrates strong performance across diverse reasoning tasks, we acknowledge several limitations that suggest directions for future work:

Dependence on structured reasoning formats. The framework assumes reasoning traces are expressed as explicit, step-by-step rationales with discernible logical transitions. In settings where models produce highly condensed or implicitly structured reasoning, such as in some latent reasoning approaches or heavily optimized instruction-tuned models, the unit segmentation mechanism may struggle to identify semantically coherent boundaries. Extending SGP-CoT to such settings would require complementary techniques for reasoning decomposition or latent step detection.

Independent unit scoring is still an approximation. Our answer-impact and coherence-impact scores are computed at the individual-unit level. The buffered-retention mechanism partially relaxes this independence assumption by preserving short chains of coherence-supporting units when they precede a later answer-critical step, but it does not model arbitrary higher-order interactions among multiple individually weak units. Capturing such combinatorial dependencies would require evaluating subsets of units, whose cost grows exponentially with trajectory length. A more expressive yet tractable approximation of these interaction effects remains an important direction.

Preprocessing overhead for impact analysis. Although SGP-CoT is training-efficient and requires no external supervision, the self-guided pruning phase still requires multiple forward passes to score each reasoning unit. In practice, part of this cost can be amortized: consecutive counterfactual evaluations share long prefixes, enabling KV-cache reuse, and optimized inference runtimes such as vLLM further reduce the overhead. Under our setup, impact analysis for a 7B–8B model remains within a modest preprocessing budget (under 10 H800 GPU hours), but the cost is still non-trivial for very long trajectories or much larger datasets. Future work could explore approximate scoring methods or early pruning heuristics to further improve scalability while retaining pruning fidelity.

References

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

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

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. *CoRR*.

Zigeng Chen, Xinyin Ma, Gongfan Fang, Ruonan Yu, and Xinchao Wang. 2025. Verithinker: Learning to verify makes reasoning model efficient. *arXiv preprint arXiv:2505.17941*.

Jeffrey Cheng and Benjamin Van Durme. 2024. Compressed chain of thought: Efficient reasoning through dense representations. *arXiv preprint arXiv:2412.13171*.

Xiaoxue Cheng, Junyi Li, Zhenduo Zhang, Xinyu Tang, Wayne Xin Zhao, Xinyu Kong, and Zhiqiang Zhang. 2025a. Incentivizing dual process thinking for efficient large language model reasoning. *arXiv preprint arXiv:2505.16315*.

Zhengxiang Cheng, Dongping Chen, Mingyang Fu, and Tianyi Zhou. 2025b. Optimizing length compression in large reasoning models. *arXiv preprint arXiv:2506.14755*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Hugging Face. 2025. Open r1: A fully open reproduction of deepseek-r1.

Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. 2023. Towards revealing the mystery behind chain of thought: a theoretical perspective. *Advances in Neural Information Processing Systems*, 36:70757–70798.

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

Nathan Habib, Clémentine Fourrier, Hynek Kydlíček, Thomas Wolf, and Lewis Tunstall. 2023. Lighteval: A lightweight framework for llm evaluation.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2025a. Token-budget-aware LLM reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 24842–24855, Vienna, Austria. Association for Computational Linguistics.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2025b. Token-budget-aware llm reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 24842–24855.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason E Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. In *Workshop on Reasoning and Planning for Large Language Models*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

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*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Ayeong Lee, Ethan Che, and Tianyi Peng. 2025. How well do llms compress their own chain-of-thought? a token complexity approach. *arXiv preprint arXiv:2503.01141*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Ivan Moshkov, Darragh Hanley, Ivan Sorokin, Shubham Toshniwal, Christof Henkel, Benedikt Schifferer, Wei Du, and Igor Gitman. 2025. Aimo-2 winning solution: Building state-of-the-art mathematical reasoning models with openmathreasoning dataset. *arXiv preprint arXiv:2504.16891*.

Tergel Munkhbat, Namgyu Ho, Seo Hyun Kim, Yongjin Yang, Yujin Kim, and Se-Young Yun. 2025. Self-training elicits concise reasoning in large language models. *arXiv preprint arXiv:2502.20122*.

Penghui Qi, Zichen Liu, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025. Optimizing anytime reasoning via budget relative policy optimization. *arXiv preprint arXiv:2505.13438*.

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*.

Xiaoye Qu, Yafu Li, Zhaochen Su, Weigao Sun, Jianhao Yan, Dongrui Liu, Ganqu Cui, Daizong Liu, Shuxian Liang, Junxian He, and 1 others. 2025. A survey of efficient reasoning for large reasoning models: Language, multimodality, and beyond. *arXiv preprint arXiv:2503.21614*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741.

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*.

Matthew Renze and Erhan Guven. 2024. The benefits of a concise chain of thought on problem-solving in large language models. In *2024 2nd International Conference on Foundation and Large Language Models (FLLM)*, pages 476–483. IEEE.

Xuan Shen, Yizhou Wang, Xiangxi Shi, Yanzhi Wang, Pu Zhao, and Jiuxiang Gu. 2025a. Efficient reasoning with hidden thinking. *CoRR*.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. 2025b. Codi: Compressing chain-of-thought into continuous space via self-distillation. *arXiv preprint arXiv:2502.21074*.

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

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, and 1 others. 2025. Kimi k1. 5: Scaling reinforcement learning with llms. *CoRR*.

Qwen Team. 2025. Qwq-32b: Embracing the power of reinforcement learning.

Xuezhi Wang and Denny Zhou. 2024. Chain-of-thought reasoning without prompting. *Advances in Neural Information Processing Systems*, 37:66383–66409.

Yue Wang, Qiuzhi Liu, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Linfeng Song, Dian Yu, Juntao Li, Zhuosheng Zhang, and 1 others. 2025. Thoughts are all over the place: On the underthinking of o1-like llms. *arXiv preprint arXiv:2501.18585*.

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.

Xilin Wei, Xiaoran Liu, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Jiaqi Wang, Xipeng Qiu, and Dahua Lin. 2025. Sim-cot: Supervised implicit chain-of-thought. *arXiv preprint arXiv:2509.20317*.

Heming Xia, Chak Tou Leong, Wenjie Wang, Yongqi Li, and Wenjie Li. 2025. Tokenskip: Controllable chain-of-thought compression in llms. *arXiv preprint arXiv:2502.12067*.

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

Zifan Xu, Haozhu Wang, Dmitriy Bespalov, Xuan Wang, Peter Stone, and Yanjun Qi. 2023. Latent skill discovery for chain-of-thought reasoning. *arXiv preprint arXiv:2312.04684*.

Zifan Xu, Haozhu Wang, Dmitriy Bespalov, Xian Wu, Peter Stone, and Yanjun Qi. 2024. LaRS: Latent reasoning skills for chain-of-thought reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 3624–3643, Miami, Florida, USA. Association for Computational Linguistics.

Xuan Zhang, Chao Du, Tianyu Pang, Qian Liu, Wei Gao, and Min Lin. 2024. Chain of preference optimization: Improving chain-of-thought reasoning in llms. *Advances in Neural Information Processing Systems*, 37:333–356.

Yifan Zhang and Team Math-AI. 2024. American invitational mathematics examination (aime) 2024.

Haoran Zhao, Yuchen Yan, Yongliang Shen, Haolei Xu, Wenqi Zhang, Kaitao Song, Jian Shao, Weiming Lu, Jun Xiao, and Yueting Zhuang. 2025. Let llms break free from overthinking via self-braking tuning. *arXiv preprint arXiv:2505.14604*.

Xufeng Zhao, Mengdi Li, Wenhao Lu, Cornelius Weber, Jae-Hee Lee, Kun Chu, and Stefan Wermter. 2024a. Enhancing zero-shot chain-of-thought reasoning in large language models through logic. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 6144–6166.

Yuze Zhao, Jintao Huang, Jinghan Hu, Xingjun Wang, Yunlin Mao, Daoze Zhang, Zeyinzi Jiang, Zhikai Wu, Baole Ai, Ang Wang, Wenmeng Zhou, and Yingda Chen. 2024b. Swift:a scalable lightweight infrastructure for fine-tuning. *Preprint*, arXiv:2408.05517.

Xudong Zhu, Jiachen Jiang, Mohammad Mahdi Khalili, and Zhihui Zhu. 2025. From emergence to control: Probing and modulating self-reflection in language models. *arXiv preprint arXiv:2506.12217*.

A Ethical Considerations

A.1 Datasets

All datasets used in this work are publicly available and widely adopted in the reasoning literature. We confirm their licenses and content characteristics, as illustrated in Table 4.

Dataset	License
AIME	Apache 2.0
GPQA Diamond	CC-BY-4.0
MATH-500	MIT
GSM8K	MIT
Mixture-of-Thoughts	Apache 2.0

Table 4: Licenses of all datasets used in this work.

All datasets consist of abstract reasoning problems without personally identifiable information (PII), sensitive topics, or harmful content. Their use is consistent with their intended research purposes.

A.2 Models

We use publicly released model checkpoints with permissible research licenses, as illustrated in Table 5. All models are used in accordance with their respective license terms.

Model	License
DeepSeek-R1-Distill-Qwen (1.5B / 7B)	MIT
DeepSeek-R1-Distill-Llama (8B)	MIT
OpenMath-Nemotron (1.5B / 7B)	CC-BY-4.0

Table 5: Licenses of all models used in this work.

A.3 Generated Content

During training and evaluation, models generate reasoning traces and final answers. We manually inspected a random sample of 100 generated traces and found no instances of harmful, biased, or sensitive content. All generated text pertains strictly to the reasoning tasks (mathematics and science).

A.4 Energy and Resource Usage

Training SGP-CoT is lightweight due to the use of LoRA and small preference datasets ($\sim$1.5K samples per model). Total GPU hours across all experiments were approximately 200 hours on H800 GPUs. We acknowledge the carbon footprint associated with this compute and encourage future work to further optimize efficiency.

A.5 Broader Impact

This work aims to improve the efficiency of reasoning models, which could reduce inference costs and energy consumption in real-world deployments. However, more efficient reasoning could also lower the barrier to deploying LLMs in sensitive or high-stakes domains. We encourage practitioners to accompany efficiency improvements with appropriate safeguards, fairness audits, and transparency measures.

B Dataset Statistics

B.1 Dataset Composition

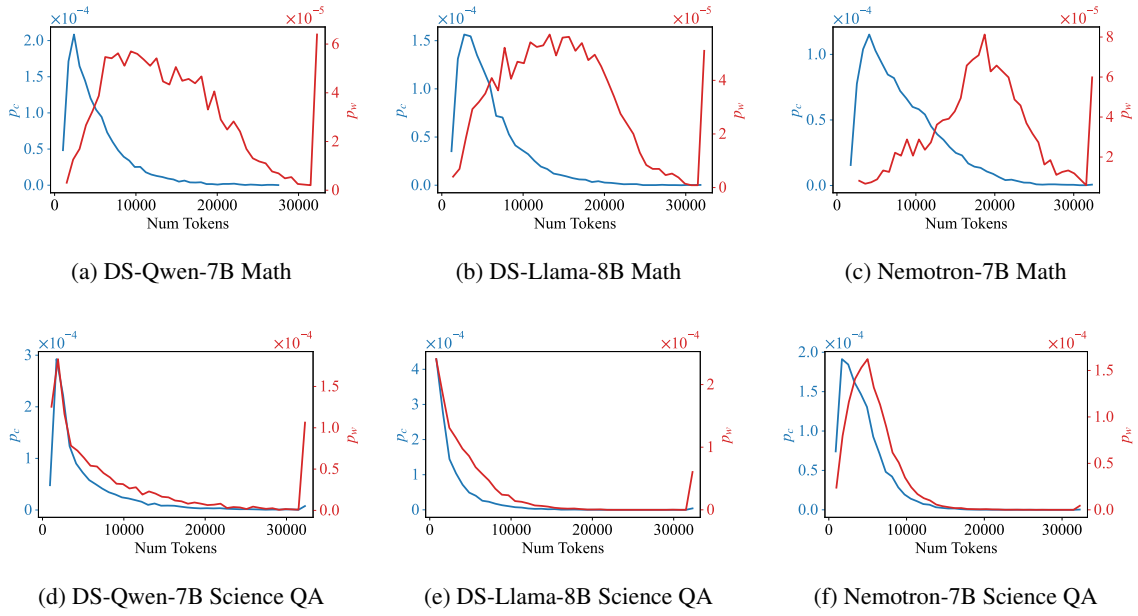
The training dataset for each model is constructed by applying the self-guided pruning pipeline to two distinct reasoning corpora: mathematics and science question-answering (QA). The mathematics portion comprises approximately 800 problems from the AIME dataset (years prior to 2023), providing challenging multi-step reasoning traces. The QA portion contains roughly 1,000 science-domain questions sampled from the Mixture-of-Thoughts dataset. After pruning and preference-pair generation, the resulting dataset for each target model consists of about 1.5K preference samples. The final data maintains an approximate 3:2 ratio of pruning-based preference pairs to sampling-based pairs, and a roughly 1:1 balance between mathematics and QA examples. All samples are filtered to ensure that the preferred (shorter) response remains correct, guaranteeing a clean supervision signal for DPO training.

B.2 Length Distribution of Generated Responses

To characterize the reasoning behavior of different models before pruning, we analyze the token-length distributions of generated responses across two domains: mathematics and scientific QA. Figure 5 illustrates the distribution of response lengths for three representative models: DeepSeek-R1-Distill-Qwen-7B, DeepSeek-R1-Distill-Llama-8B, and OpenMath-Nemotron-7B. Responses are categorized as correct or incorrect based on final-answer accuracy.

Key observations emerge from these distributions:

- **Correct responses** exhibit a left-skewed unimodal distribution in both domains. Mathematical reasoning traces are systematically longer on average, reflecting the step-intensive



(a) DS-Qwen-7B Math (b) DS-Llama-8B Math (c) Nemotron-7B Math

(d) DS-Qwen-7B Science QA (e) DS-Llama-8B Science QA (f) Nemotron-7B Science QA

Figure 5: Distribution of the number of tokens in the sampled responses for training. Blue lines indicate the probability density of the correct responses (p_c), while the red ones represent that of the wrong responses.

nature of competition-level problem-solving compared to structured science QA.

- **Incorrect responses** show distinct patterns across domains: In science QA, incorrect responses follow a length distribution similar to correct ones, suggesting errors stem from logical missteps rather than excessive verbosity. In mathematics, incorrect responses are markedly longer (peak $>$ 10K tokens) and exhibit higher variance. A substantial fraction exceeds 32K context limit, indicating that models often enter repetitive loops or "overthink" when failing to solve mathematical problems.
- **Context limit** effects are prominent in mathematical error cases, where many traces are truncated at 32K tokens during processing. Truncation is rare in science QA, further underscoring the differential reasoning dynamics between domains.

C Segmentation Strategy of Reasoning Units

C.1 Segmentation Trigger Identification

We adopt a line-based segmentation strategy to decompose reasoning traces into semantically coherent units. The key observation is that LRMs frequently use discourse markers or logical operators at the beginning of a reasoning line to signal a transition to a new logical sub-step. To identify these markers in a model-specific and data-driven manner, we collect all reasoning responses from the target model, extract the first token of each line, and compute its empirical frequency across the corpus. The top-10 most frequent tokens are selected as the segmentation trigger set $\mathcal{T}_{\text{seg}}$. During segmentation, any line whose first token belongs to $\mathcal{T}_{\text{seg}}$ is treated as the start of a new reasoning unit. This approach is lightweight, does not require syntactic parsing, and naturally adapts to the linguistic style of each model.

To illustrate the model-specific nature of these triggers, Table 6 shows the top-10 most frequent line-initial tokens and their probabilities for each target model. For a more comprehensive view of each model's lexical segmentation style, we also visualize broader trigger distributions through word clouds (see Figure 6).

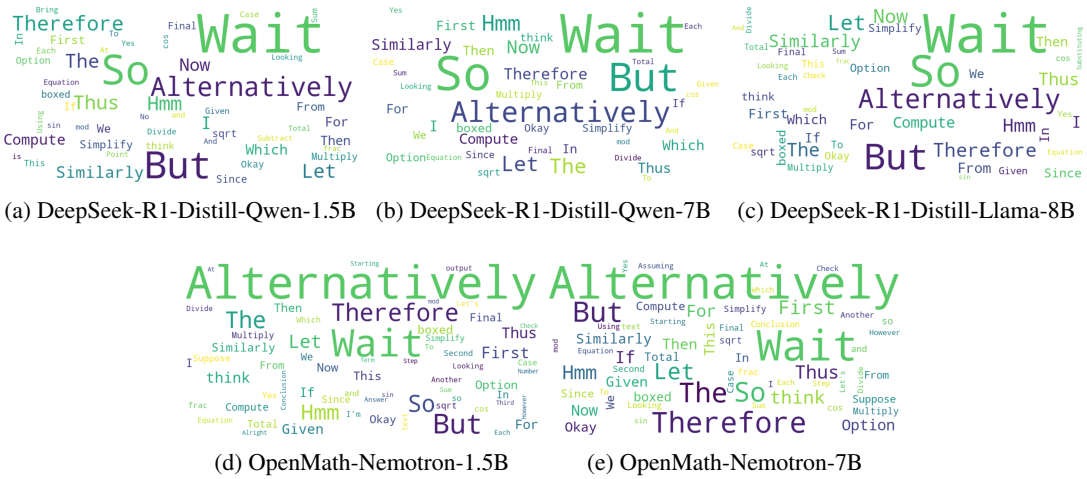
C.2 Observations and Implications

Several consistent patterns emerge from the trigger distributions:

High-frequency logical connectives dominate Tokens such as *Wait*, *So*, *But*, and *Therefore* appear prominently across all models, confirming that LRMs rely on explicit discourse markers to structure step-by-step reasoning.

DS-Qwen-1.5B		DS-Qwen-7B		DS-Llama-8B		Nemotron-1.5B		Nemotron-7B	
Word	Prob(%)	Word	Prob(%)	Word	Prob(%)	Word	Prob(%)	Word	Prob(%)
Wait	25.00	Wait	21.14	Wait	22.57	Alternatively	18.38	Alternatively	14.50
So	13.31	So	16.26	So	14.32	Wait	12.53	Wait	12.43
But	11.10	But	10.65	But	9.05	But	5.56	Therefore	4.58
Alternatively	4.41	Alternatively	4.94	Alternatively	4.20	Therefore	4.05	But	4.31
Therefore	2.40	Hmm	2.66	Let	2.63	So	3.68	So	3.91
Let	2.17	Let	2.44	Therefore	2.06	Hmm	3.07	Let	2.80
Thus	2.14	Now	1.85	Now	2.05	Let	2.30	First	2.02
Hmm	2.11	Therefore	1.34	Hmm	1.82	First	1.65	Hmm	2.00
Now	1.62	Similarly	1.24	Thus	1.61	Thus	1.50	Thus	1.82
Similarly	1.37	Thus	1.08	Similarly	1.58	Given	1.25	For	1.46

Table 6: Top line-initial tokens and their empirical probabilities across model families.



(a) DeepSeek-R1-Distill-Qwen-1.5B (b) DeepSeek-R1-Distill-Qwen-7B (c) DeepSeek-R1-Distill-Llama-8B

(d) OpenMath-Nemotron-1.5B (e) OpenMath-Nemotron-7B

Figure 6: Word clouds of line-initial tokens across model families.

Model-specific stylistic preferences While the set of frequent triggers is largely shared, their relative rankings vary. For example, *Wait* is the most common trigger in DeepSeek-R1 distilled models, whereas *Alternatively* ranks highest in Nemotron models. This variation reflects differences in pre-training data and instruction-tuning styles, justifying our model-specific trigger selection approach.

Presence of "hedging" tokens Tokens like *Hmm*, *Let*, and *Thus* frequently appear, suggesting that LRMs use meta-cognitive markers to regulate reasoning flow, e.g., pausing (*Hmm*) or introducing a sub-goal (*Let*). These tokens often correspond to coherence-supporting units that our pruning strategy buffers rather than removes.

Stability across model scales Within the same model family, trigger distributions are highly similar, indicating that reasoning style is preserved across parameter scales.

These findings support our design choice of using data-driven, model-adaptive triggers for unit segmentation. The consistency of logical markers across models validates the generality of our segmentation approach, while the observed variations underscore the importance of avoiding a fixed, one-size-fits-all trigger set.

D Pruning Algorithm

For completeness, we provide the full algorithm for the impact-based pruning strategy with buffered retention in Algorithm 1, which implements the conservative unit-level pruning rules described in Section 3.2.2.

E Analysis of Impact Score Distribution

Figure 7 presents the probability density distributions of the answer impact (Δ_A) and coherence impact (Δ_C) scores computed from the training data of the five models optimized in our study. Several key observations emerge from these distributions, which help explain and justify two important design choices in SGP-CoT: (1) the disparity between the answer and coherence thresholds (τ_A and τ_C),

Algorithm 1 Impact-Based CoT Pruning with Buffered Retention

Require: Reasoning units $\mathcal{U} = (U_1, \ldots, U_m)$; answer impact scores $\Delta_A(1:m)$; coherence impact scores $\Delta_C(1:m)$; thresholds τ_A, τ_C
Ensure: Preserved unit index set $\mathcal{K}$

$\mathcal{K} \leftarrow \{1\}$ {Preserve boundary unit}
$B \leftarrow \emptyset$ {Temporary buffer for coherence-supporting units}
$i \leftarrow 2$
while $i \leq m$ **do**
  if $\Delta_A^i \geq \tau_A$ **then**
    $\mathcal{K} \leftarrow \mathcal{K} \cup B$ {Commit buffered units}
    $\mathcal{K} \leftarrow \mathcal{K} \cup \{i\}$
    $B \leftarrow \emptyset$
  else
    if $\Delta_C^i \geq \tau_C$ **then**
      $B \leftarrow B \cup \{i\}$ {Buffer for possible retention}
    else
      $B \leftarrow \emptyset$ {Unsafe buffer discarded}
    end if
  end if
  $i \leftarrow i + 1$
end while
return $\mathcal{K}$

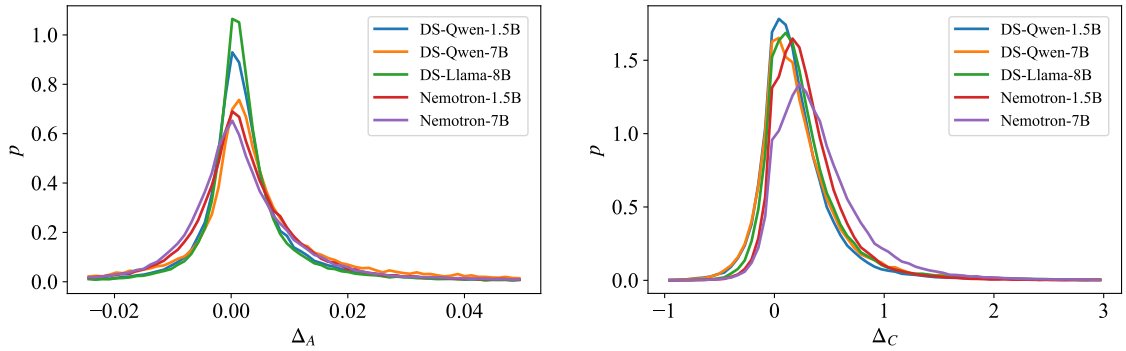
and (2) the generalization of these thresholds across diverse model families.

Distribution Characteristics. Both distributions are unimodal, but they differ significantly in scale and shape. The answer impact Δ_A (Figure 7a) is concentrated within a narrow range of $[-0.02, 0.02]$, with its peak slightly above zero and a roughly symmetric, near-normal shape on both sides. In contrast, the coherence impact Δ_C (Figure 7b) spans a much wider interval of $[-0.5, 1.5]$, with its peak also near zero but exhibiting a clear asymmetry: the left side falls off steeply, while the right tail decays more gradually, indicating that a large proportion of units have a positive coherence impact.

Why the Thresholds Differ by an Order of Magnitude. The scale difference between the two distributions (roughly a factor of 50) directly motivates the large gap between the thresholds τ_A and τ_C used in our pruning strategy ($\tau_A \approx 0.01$ and $\tau_C \approx 0.5$). This discrepancy is not an arbitrary hyperparameter choice, but a natural consequence of how reasoning likelihoods propagate in autoregressive models. A reasoning unit typically exerts the strongest influence on the immediately following step (modeled by Δ_C), as it provides the most direct contextual and logical support. Its influence diminishes with distance, and thus its effect on the final answer (which is often many steps away) is inherently weaker (captured by Δ_A). Therefore, Δ_C scores are naturally larger in magnitude, necessitating a correspondingly higher threshold to identify units that meaningfully support local coherence.

Cross-Model Consistency Enables Threshold Generalization. More importantly, the five curves in each figure are highly similar, showing almost perfect overlap across all model architectures and scales tested. This high degree of distributional consistency indicates that the internal likelihood dynamics governing Δ_A and Δ_C are largely model-agnostic. As a result, the sensitivity analysis performed on a single model yields thresholds that are directly applicable to other models without further tuning.

In summary, the impact score distributions provide both an empirical justification for our threshold selection and evidence for the robustness of our self-guided pruning mechanism across diverse models. They confirm that (1) the scale difference between Δ_A and Δ_C is intrinsic to the reasoning process, and (2) the underlying likelihood signals are consistent enough to allow hyperparameters tuned on one model to generalize effectively to others.



(a) Answer Impact

(b) Coherence Impact

Figure 7: Distribution of answer impact and coherence impact scores from 5 optimized models.

F Training Details

For model training, we used Direct Preference Optimization (DPO) with a proportion of SFT loss ($\times$ 1.0) mixed into the loss to improve training stability. The KL regularization coefficient β was set to 0.1, and the policy was optimized using the AdamW optimizer with a learning rate of $1.0\times10^{-5} \sim 1.0\times10^{-4}$ (1.0×10^{-4} for Nemotron models and 1.0×10^{-5} for others) and a linear warmup over the first 5% of training steps. Training was conducted for a single epoch with a global batch size of 16 with SWIFT framework (Zhao et al., 2024b). To reduce memory overhead and accelerate training, we employed LoRA (Hu et al., 2022) with a rank of 256, applied to all linear layers in the attention and feed-forward modules. The maximum sequence length was set to 32K tokens to accommodate lengthy reasoning traces.

G Distribution Shift of SGP-CoT

To better understand how SGP-CoT influences reasoning behavior beyond aggregate accuracy and length metrics, we analyze the distribution of response lengths on the test sets of AIME-2024, GPQA-Diamond, and MATH-500. Figure 8 compares the token-length distributions of the original (vanilla) DeepSeek-R1-Distill-Qwen-7B model and its SGP-CoT-optimized counterpart, separating correct and incorrect responses. We observe several consistent patterns:

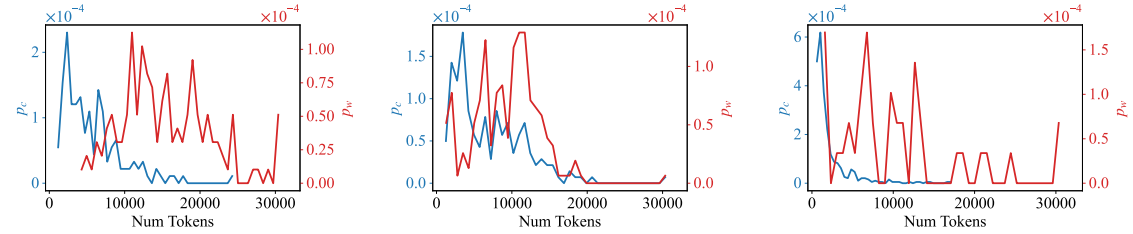
Pre-optimization verbosity in errors. In the vanilla model, incorrect responses are systematically longer than correct ones across all benchmarks, often exceeding 16K tokens and occasionally reaching the context limit (32K). This suggests that when the model struggles, it tends to over-reason, possibly entering repetitive or unfocused reasoning loops.

Post-optimization distribution shift toward conciseness. After applying SGP-CoT, the overall length distribution shifts leftward (toward shorter responses). Notably:

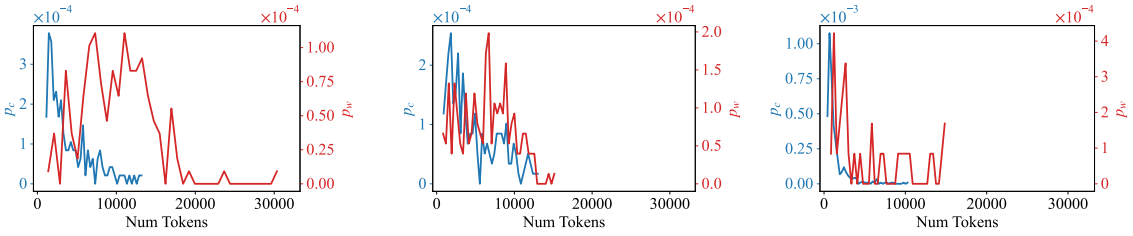
- On AIME 2024, the maximum length of correct responses drops from 25K to 13K tokens, and the proportion of incorrect responses longer than 16K decreases substantially.
- On GPQA Diamond and MATH-500, nearly all responses are compressed below 16K tokens, indicating that SGP-CoT effectively suppresses excessively verbose reasoning across both success and failure modes.

Tighter distribution with preserved discriminability. While SGP-CoT shortens responses overall, the separation between correct and incorrect response lengths remains discernible, implying that the model retains its ability to distinguish confident reasoning from uncertain exploration. This aligns with our framework's design: pruning removes redundancy but does not collapse the internal confidence landscape.

These distributional changes reinforce that SGP-CoT not only reduces average length but also regularizes the reasoning process, curbing overlong and unproductive reasoning trails while maintaining the capacity for sustained reasoning when needed. This shift contributes to lower inference cost and more predictable generation behavior, which is a desirable property for deployment.



(a) AIME 2024 Vanilla (b) GPQA Diamond Vanilla (c) MATH-500 Vanilla



(d) AIME 2024 SGP-CoT (e) GPQA Diamond SGP-CoT (f) MATH-500 SGP-CoT

Figure 8: Distribution of the number of tokens in the generated responses in evaluation. The upper charts present the distribution of vanilla DeepSeek-R1-Distill-Qwen-7B, and the lower ones present that of models optimized with SGP-CoT.

Model	Method	AIME 2025		HMMT Feb. 2025	
		Pass@1↑	Tokens↓	Pass@1↑	Tokens↓
DS-Qwen-1.5B	Baseline	19.7	14025	10.3	15098
	LC-R1	17.7	7241	10.3	8022
	L1-Exact	19.0	2628	7.0	2384
	L1-Max	21.3	2647	8.7	2485
	SGP-CoT	19.3	7456	10.3	8540
DS-Qwen-7B	Baseline	39.7	11397	17.3	12540
	LC-R1	36.7	6887	16.0	7944
	L1-Exact	30.3	3272	11.7	3447
	L1-Max	28.0	3219	13.3	3374
	SGP-CoT	39.0	7256	17.7	7907

Table 7: Results on two additional hard mathematical reasoning benchmarks. SGP-CoT preserves the same accuracy-compression pattern observed in the main evaluation.

H Additional Efficiency and Robustness Results

H.1 Additional Hard Benchmarks

To further test robustness beyond the main evaluation suite, we report results on two harder competition-style benchmarks, AIME 2025 and HMMT Feb. 2025. As in the main text, SGP-CoT consistently preserves the strongest accuracy-efficiency balance: it substantially shortens reasoning while maintaining performance much better than more aggressive length-focused baselines.

H.2 Latency Under Matched Hardware

Table 8 provides the full per-benchmark latency measurements corresponding to the averaged results in the main text. The overall trend closely follows token reduction: SGP-CoT yields clear wall-clock gains on both model sizes, while avoiding the larger accuracy degradation that often accompanies the most aggressive compression settings.

I Sensitivity Analysis of Prompt Design

A potential concern with self-guided pruning method is that it might overfit to the specific prompt style used during training, limiting its generalization to different reasoning instructions. To verify that SGP-CoT is robust to prompt variations, we conduct a sensitivity analysis using three distinct prompt groups, each representing a different style of eliciting Chain-of-Thought reasoning.

Prompt Groups. We design three groups of prompts, summarized in Table 9. Group 1 uses detailed, instruction-heavy prompts from the LightEval toolkit, which explicitly request step-by-step reasoning and enforce a strict answer format. Group 2 employs the concise, model-native prompt style used in the training of DeepSeek-R1-Distill models. Group 3 adopts the shorter instruction format preferred by Nemotron models, which focuses

Model	Variant	AIME 2024 Latency (s)↓	GPQA Diamond Latency (s)↓	MATH-500 Latency (s)↓	GSM8K Latency (s)↓
DS-Qwen-1.5B	Baseline	655 ± 18	978 ± 1	540 ± 4	385 ± 8
	LC-R1	354 ± 8	691 ± 2	259 ± 2	258 ± 11
	L1-Exact	106 ± 34	183 ± 2	265 ± 13	235 ± 6
	L1-Max	98 ± 33	221 ± 1	161 ± 1	327 ± 62
	SGP-CoT	327 ± 3	516 ± 2	285 ± 1	185 ± 43
DS-Qwen-7B	Baseline	877 ± 49	1529 ± 12	588 ± 32	340 ± 46
	LC-R1	512 ± 69	949 ± 1	225 ± 7	99 ± 1
	L1-Exact	163 ± 13	396 ± 1	146 ± 1	381 ± 1
	L1-Max	159 ± 3	395 ± 0	133 ± 0	229 ± 5
	SGP-CoT	469 ± 47	953 ± 1	343 ± 11	114 ± 1

Table 8: End-to-end latency under identical hardware conditions (single H800 GPU). Each number reports the mean and standard deviation over three runs.

on answer formatting with minimal reasoning directives.

Experimental Setup. For each prompt group, we retrain a DeepSeek-R1-Distill-Qwen-7B model using SGP-CoT with data collected under the corresponding prompt. At inference time, we evaluate each optimized model on all four benchmarks using the same prompt it was trained with. This design tests whether SGP-CoT's performance is stable across prompt styles, or whether it degrades when the prompt format changes.

Results. Table 10 presents the performance of SGP-CoT across the three prompt groups. Across all groups, SGP-CoT consistently reduces reasoning length by 30–40% on challenging tasks (AIME 2024, GPQA Diamond, MATH-500) and by 20–27% on GSM8K, while accuracy is either preserved or slightly improved. Notably, the magnitude of length reduction and accuracy gain remains stable regardless of prompt verbosity or structural differences.

These results demonstrate that SGP-CoT's self-guided pruning mechanism is not sensitive to prompt design. The framework relies on the model's internal likelihood dynamics to assess reasoning unit importance, a signal that is intrinsic to the model's reasoning process rather than a byproduct of prompt phrasing. Consequently, SGP-CoT generalizes robustly across different instruction styles.

Group No.	Source	Task	Prompt
1	LightEval	Math	Solve the following math problem efficiently and clearly. The last line of your response should be of the following format: 'Therefore, the final answer is: $\boxed{ANSWER}$. I hope it is correct' (without quotes) where ANSWER is just the final number or expression that solves the problem. Think step by step before answering.
		QA	Answer the following multiple choice question. The last line of your response should be of the following format: 'Answer: $LETTER' (without quotes) where LETTER is one of ABCD. Think step by step before answering.
2	DS-Distill Models	-	Please reason step by step, and put your final answer within \boxed{}.
3	Nemotron Models	Math	Solve the following math problem. Make sure to put the answer (and only answer) inside \boxed{}.
		QA	Answer the following multiple choice question. Make sure to put the answer (and only answer) inside \boxed{}.

Table 9: Summary of prompt groups used in the sensitivity analysis. Each group represents a distinct style for eliciting Chain-of-Thought reasoning.

Group No.	Method	AIME 2024		GPQA Diamond		MATH-500		GSM8K	
		Pass@1↑	Tokens↓	Pass@1↑	Tokens↓	Pass@1↑	Tokens↓	Pass@1↑	Tokens↓
1	Baseline	52.3	10400	48.5	8213	92.0	2793	84.9	532
	SGP-CoT	$52.7_{0.4\uparrow}$	$6965_{33.0\%\downarrow}$	$50.8_{2.3\uparrow}$	$5692_{30.7\%\downarrow}$	$93.4_{1.4\uparrow}$	$1687_{39.6\%\downarrow}$	$85.7_{0.8\uparrow}$	$414_{22.0\%\downarrow}$
2	Baseline	53.0	13092	50.0	8195	92.8	4193	87.0	541
	SGP-CoT	$53.0_{0.0}$	$8939_{31.7\%\downarrow}$	$50.5_{0.5\uparrow}$	$5528_{32.5\%\downarrow}$	$92.8_{0.0}$	$2898_{30.9\%\downarrow}$	$87.1_{0.1\uparrow}$	$434_{19.8\%\downarrow}$
3	Baseline	51.0	10495	48.3	8244	91.4	3810	85.0	674
	SGP-CoT	$51.7_{0.7\uparrow}$	$7179_{31.6\%\downarrow}$	$48.3_{0.0}$	$5127_{37.8\%\downarrow}$	$91.6_{0.2\uparrow}$	$2339_{38.6\%\downarrow}$	$85.7_{0.7\uparrow}$	$493_{26.9\%\downarrow}$

Table 10: Performance of SGP-CoT across three prompt groups on four reasoning benchmarks. Results show consistent length reduction and stable accuracy regardless of prompt style.