ChainPrune: Evaluating and Reducing Redundancy in Long Chain-of-Thought Reasoning

Weihang Pan[1,†], Zhengxu Yu[2], Yuxiang Zhang[1], Wenzhi Li[1], Zhongming Jin[2], Binbin Lin[1,*], Xiaofei He[1], Jieping Ye[2]

[1]Zhejiang University
[2]Alibaba Group
panweihang@zju.edu.cn, binbinlin@zju.edu.cn

Abstract

Chain-of-Thought (CoT) reasoning has significantly enhanced the multi-step problem-solving capabilities of large language models (LLMs) by introducing explicit intermediate reasoning. However, advanced Large Reasoning Models (LRMs) often exhibit overthinking behaviors, including excessively long reasoning steps, redundant steps, and high computational overhead. Existing token-length reward strategies aim to promote concise outputs, but often result in pseudo-conciseness, where token count is reduced, yet redundant reasoning persists, leading to longer and less structurally efficient chains. To address these limitations, we propose **ChainPrune**, a novel reasoning path semantic structural optimization method to efficiently and controllably synthesize self-generated high-quality training data. We initially consolidate self-generated reasoning paths into a tree-based structure, followed by a multi-criteria dominant path selection process for preference data construction that formulates shallow reasoning trajectories while preserving essential reasoning steps. To further enhance the quality of reasoning, we incorporate a DPO-based preference learning method combined with supervised loss, effectively mitigating false reward suppression. This innovative integration significantly enhances both the efficiency and effectiveness of our reasoning framework. Comprehensive experimental results demonstrate significant reductions in step length and computational overhead, while maintaining or even enhancing accuracy.

Introduction

The emergence of Chain-of-Thought (CoT) (Wei et al. 2022) reasoning has signaled a paradigm shift in large language models (LLMs) by facilitating multi-step problem-solving through the articulation of explicit intermediate reasoning steps. Initial CoT methods revealed that breaking down complex tasks into structured steps—mirroring human deliberation—can significantly improve performance across various domains, including mathematics and symbolic reasoning.

Recent advances in Large Reasoning Models (LRMs) (Xu et al. 2025a), such as OpenAI's O1 (Jaech et al. 2024) and DeepSeek-R1 (Guo et al. 2025), have systematized the Chain-of-Thought (CoT) process through differentiated reinforcement learning paradigms, incorporating innovative architectures like Monte Carlo Tree Search (MCTS) (Browne et al. 2012; Coulom 2006) and pure reinforcement learning strategies. These o1-like models emulate human-like "slow thinking" (Li et al. 2025) by exploring multiple solution strategies, engaging in self-reflection, and iteratively correcting errors (Liang et al. 2024). However, this deliberative approach gives rise to the **overthinking phenomenon** (Chen et al. 2024b; Team et al. 2025), characterized by exponential proliferation of reasoning steps, excessive verbosity, and redundancy. Such inefficiencies result in significant computational overhead and error accumulation, ultimately impeding the practical deployment of these models. Addressing these challenges is crucial for enhancing the utility and applicability of LRMs in real-world scenarios.

To mitigate this, researchers have pursued efficient reasoning techniques to shorten reasoning sequences while preserving accuracy (Han et al. 2024; Hao et al. 2024; Luo et al. 2025; Ma et al. 2025; Yeo et al. 2025). Recent guide-based

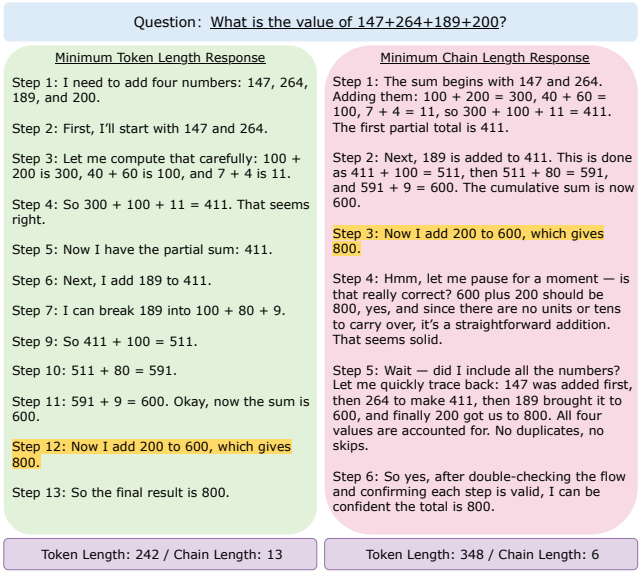

Figure 1: Comparison between a Minimum Token Length Response and a Minimum Chain Length Response for the same addition problem.

* Corresponding author.
† This work was completed during an internship at Alibaba Group in May 2025.

methods (Liao et al. 2025) attempt to address this by framing the reasoning process as a tree search problem, then steering the search toward concise outputs through manual rules or step-by-step supervision. However, these approaches introduce reward bias that propagates reasoning errors. Furthermore, several methods (Luo et al. 2025; Team et al. 2025; Shen et al. 2025a; Arora and Zanette 2025) introduce token-length reward mechanisms during RL training to guide concise outputs. Yet, such token–length–only optimization often leads to what we call pseudo-conciseness: while the token count is reduced, the reasoning path is fragmented into many micro-steps, inflating chain length and reducing structural efficiency. Figure 1 illustrates this phenomenon with a simple arithmetic example: the Minimum Token Length Response minimizes tokens by using terse phrases per step, but its chain length becomes unnecessarily long. In contrast, the Minimum Chain Length Response consolidates the reasoning into fewer, semantically richer steps, achieving shorter chain length but not necessarily minimizing tokens. The ideal scenario—and the target of our work—is to reduce both token usage and chain length simultaneously.

To this end, we propose ChainPrune, a novel low-cost reasoning path semantic structural optimization method that synthesizes high-quality training data by merging semantically equivalent reasoning steps across multiple sampled paths. By consolidating redundant reasoning into a compact, logically complete chain, ChainPrune produces reasoning paths that are shorter in both tokens and steps, without sacrificing correctness or completeness. This joint optimization ensures that our models avoid pseudo-conciseness and instead generate genuinely efficient reasoning sequences. We initially consolidate self-generated reasoning paths into a tree-based structure, with dynamic semantic node merging to prune redundancy while preserving critical reasoning points. Subsequently, we introduce a multi-criteria dominant path selection mechanism to construct the preference dataset, which jointly optimizes chain and token efficiency. For chosen response selection, we prioritize Pareto-dominant reasoning paths when available; otherwise, we favor shorter responses among non-dominated candidates. Rejected responses are required to exhibit substantially longer reasoning steps or incorrect logic to ensure meaningful comparison. We then train a variant of DPO that includes a negative log-likelihood (NLL) loss term for the winning pairs, which proves crucial for enforcing both correctness and efficiency in reasoning quality.

Our main contributions are as follows:

- We reveal that current token-length optimization methods suffer from pseudo-conciseness, where token count is reduced but reasoning paths remain redundant. We propose ChainPrune, a low-cost compression method based on semantic structural similarity, which merges semantically equivalent nodes in sampled reasoning paths to reduce redundant steps while preserving core logic.
- We design a novel preference data construction pipeline that dynamically searches for Pareto-optimal paths from merged reasoning trees, automatically generating high-quality training data without manual annotation. Additionally, our experiments compare three preference learning strategies (DPO (Rafailov et al. 2023), SimPO (Meng, Xia, and Chen 2024), and DPO + NLL Loss (Pang et al. 2024)), showing that the DPO + NLL Loss objective significantly shortens reasoning steps while maintaining reasoning quality.
- We conduct extensive experiments across 5 reasoning benchmarks, showing **ChainPrune** improves reasoning efficiency by 26.8% chain lengths and 28.1% token lengths reduction without accuracy drop. To holistically evaluate reasoning quality, we design an **LLM-as-a-Judge** framework with human verification, analyzing faulty reasoning, reflection mechanisms, and step efficiency. The results demonstrate significant reductions of 57.8% in faulty steps, 62.1% in invalid reflections, and 65.9% in redundant steps compared to the base model. This reveals our method's superiority in maintaining coherent logic while minimizing redundant deliberation.

Formal Analysis of Reasoning

Formal Modeling of Autoregressive Inference

The inference process of large language models (LLMs) can be formalized as an *autoregressive generation paradigm*: Given an initial context $\boldsymbol{x}_0 = (x_{01}, \ldots, x_{0k}) \in \mathcal{A}^k$, the model generates tokens sequentially as:

$$x_{i+1} \sim \mathcal{C}_{i+1} = f(\boldsymbol{x}_0 \oplus x_1 \oplus \cdots \oplus x_i), \tag{1}$$

where $\oplus$ denotes sequence concatenation, and $\mathcal{C}_{i+1}$ is a probability distribution over vocabulary $\mathcal{A}$. This process can be naturally viewed as a **tree search problem**: The root node is the initial input $\boldsymbol{x}_0$, each branch represents a candidate token expansion $a \in \mathcal{A}$, and the resulting inference tree $\mathcal{T}$ grows exponentially with depth ($\mathcal{O}(b^d)$, $b = |\mathcal{A}|$).

Recent reasoning-focused LLMs, such as OpenAI-O1 (Jaech et al. 2024) and DeepSeek-R1 (Guo et al. 2025), extend this paradigm from single-step prediction to structured chain-of-thought generation. Implicit CoT methods construct reasoning chains without explicit intermediate supervision, whereas explicit-format approaches separate reasoning from final answers via semantic tags (e.g., <think>, <answer>) and apply format-constrained rewards. State-of-the-art methods further integrate self-verification (e.g., code execution, symbolic checks) to detect and correct reasoning errors within differentiable feedback loops. While these advances improve reasoning quality, they also reveal inefficiencies in how reasoning paths are generated and selected—particularly in reinforcement learning–optimized models—leading to the challenges described next.

Challenges in RL-Optimized Reasoning Paths

Although reinforcement learning (RL) fine-tuning enhances reasoning ability, our analysis shows that reward designs often prioritize *token-level conciseness* while neglecting the *structural length* of reasoning chains. This bias produces what we call **pseudo-conciseness**: responses that minimize tokens by using short phrases per step, yet fragment the logic into many steps, resulting in long and inefficient chains.

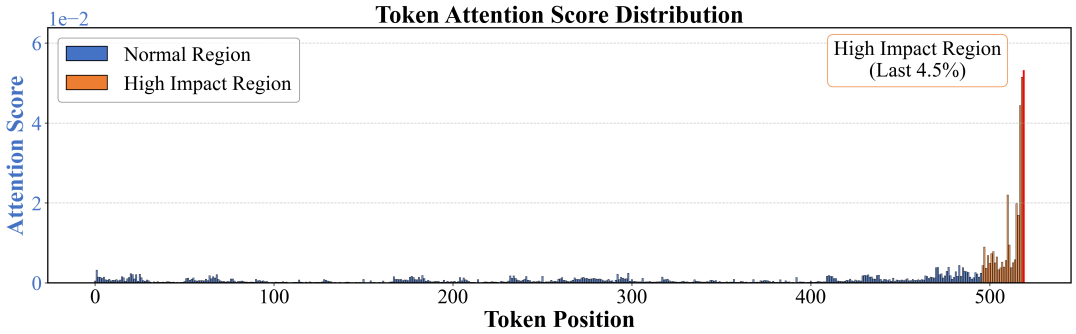

Figure 2: Attention weight distribution following semantically equivalent steps, showing localized focus on the most recent tokens (x-axis: Token Position, y-axis: Attention Score). This supports our tree-merging approach: merging equivalent nodes preserves the conditional distribution of subsequent tokens, enabling both token and step reduction without loss of correctness.

This issue directly connects to the phenomenon illustrated in Figure 1: for the same arithmetic question, a *minimum-token-length* path can have far more reasoning steps than a more compact chain. However, most existing preference-learning pipelines (e.g., DPO) still select "chosen" responses solely by minimal token length among correct outputs. Our statistical study confirms the misalignment—only 33.95% of the shortest-token correct paths are also the shortest in steps (see Appendix for detailed analysis).

To address this gap, we define a **path domination** criterion: Given two correct paths τ_1 and τ_2, τ_1 *dominates* τ_2 ($\tau_1 \prec \tau_2$) if τ_1 has *both* fewer reasoning steps and shorter token length. A **Pareto-dominant** path dominates all others; when no such path exists, the set of *non-dominated* candidates forms the Pareto frontier. This dual-objective selection principle directly aligns with our goal: identify reasoning chains that are concise in both *expression* and *structure*.

Cross-Path Semantic Reusability in Tree Search

Beyond selection criteria, our investigation of sampled reasoning paths reveals another key observation: **semantically equivalent intermediate steps occur frequently across different paths**. These equivalent steps appear in both "chosen" and "rejected" candidates and often emerge at similar structural positions.

This recurrence implies an opportunity for *semantic composability* in the tree search space. If two reasoning paths share a semantically equivalent node (node_i), their subsequent continuations (node_{i+1}, node_{i+2}, ...) are largely determined by the *local* context of that node. Figure 2 shows that, following semantically equivalent nodes, the model's attention focuses on the most recent tokens, indicating that merging at this point preserves the conditional distribution.

We exploit this property via a **tree-merging** strategy: Merge semantically equivalent nodes across sampled paths and reuse the best-performing continuations from these nodes. This process eliminates duplicated segments, producing composite paths that are *shorter in steps* and *shorter in tokens*, without compromising correctness.

This cross-path semantic reusability is a central mechanism in our proposed **ChainPrune** framework. By integrating it with the Pareto-dominance criterion, ChainPrune transforms diverse, verbose sampled outputs into compact, high-quality reasoning chains—achieving simultaneous token and step efficiency while preserving logical integrity.

Methodology

In this section, we present **ChainPrune**, a framework for generating concise and reliable reasoning chains by pruning redundancy in sampled paths and learning from structured preference signals. The core novelty lies in integrating *semantic path merging* with *dual-objective path selection* (optimizing both token length and reasoning step length) and *enhanced preference optimization*. As illustrated in Figure 3, ChainPrune operates in three synergistic stages: (1) **Reasoning Paths Merging**, which consolidates multiple sampled reasoning paths into a unified tree structure via semantic node merging, while pruning redundant steps and preserving key reasoning points; (2) **Multi-Criteria Dominant Path Selection and Preference Dataset Construction**, which applies Pareto-dominance criteria to jointly optimize chain and token efficiency; (3) **Preference Optimization**, which fine-tunes the model with an enhanced DPO objective incorporating negative log-likelihood (NLL) regularization.

Reasoning Paths Merging

Given K candidate reasoning paths $\{\tau_1, \ldots, \tau_K\}$ generated by the base LLM, this stage constructs a compact reasoning tree that preserves all essential trajectories while removing redundant steps. Each reasoning step is represented as an embedding vector, and merging proceeds as follows.

(1) Semantic Node Matching. We match a new step u to an existing node v in the tree based on *cosine similarity*:

$$\text{Sim}(u, v) = \frac{\langle \mathbf{e}_u, \mathbf{e}_v \rangle}{\|\mathbf{e}_u\| \, \|\mathbf{e}_v\|} \tag{2}$$

where $\mathbf{e}_u$ and $\mathbf{e}_v$ are the embeddings of steps u and v. If $\text{Sim}(u, v) \geq \theta_{\text{sim}}$, the step is considered semantically equivalent to v and is merged; otherwise, a new branch is created. Here θ_{sim} is the similarity threshold controlling merge strictness, distinct from any path-related thresholds used elsewhere in our framework.

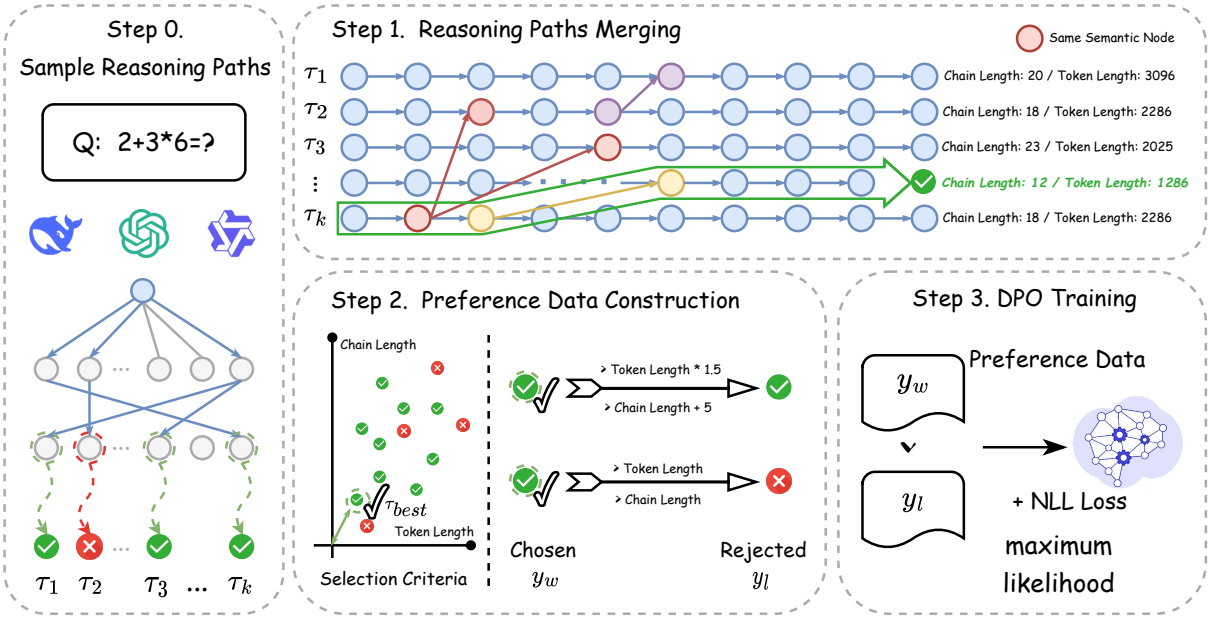

Figure 3: The pipeline of ChainPrune, featuring three synergistic stages: (1) Reasoning Paths Merging with dynamic semantic node pruning, (2) Multi-criteria dominant path selection and Preference Data Construction, (3) Direct preference learning with NLL Loss.

(2) Avoiding Incorrect Merges. Semantic similarity alone may mistakenly identify nodes that are lexically similar but logically different as equivalent. We therefore add a *token-level entropy criterion* to verify merge candidates. Given two candidate nodes u and v, we simulate replacing u with v in its original context and compute the Shannon entropy of the token probability distribution over the entire reasoning chain before and after the replacement:

$$H = -\sum_{t \in \mathcal{A}} P(t) \log P(t), \tag{3}$$

where $P(t)$ is the model's predicted token probability and $\mathcal{A}$ is the vocabulary. Let H_{before} and H_{after} be the entropy before and after replacement. The merge is accepted only if $\Delta H = H_{\text{after}} - H_{\text{before}} \leq \epsilon$, meaning predictive uncertainty does not increase. This two-stage filtering—combining cosine similarity with entropy change—ensures merges remain semantically and logically consistent.

Multi-criteria Dominant Path Selection and Preference Data Construction

From the merged reasoning tree, we construct preference datasets via a two-phase selection process.

(1) Pareto-Dominant Selection phase. For each candidate responses $\{\tau_1, \ldots, \tau_K\}$, we define the efficiency score:

$$\mathcal{E}(\tau_j) = \sqrt{\ell^2_{\text{token}}(\tau_j) + \ell^2_{\text{step}}(\tau_j)} \tag{4}$$

where $\ell_{\text{token}}(\tau_j)$ and $\ell_{\text{step}}(\tau_j)$ denote token length and chain length, respectively. The preferred answer τ^+ is:

$$\tau^+ = \arg\min_{\tau_j \in \tau_{\text{correct}}} \mathcal{E}(\tau_j) \tag{5}$$

with τ_{correct} being the set of responses matching the ground truth. If a unique Pareto-dominant path exists (shortest in both token and step length), it is chosen; otherwise, we select the shortest-token path among non-dominated Pareto-front candidates.

(2) Threshold-based Rejection phase. We filter out low-quality responses τ^- using two rules:

- **Overlong but Correct**: Correct responses that are significantly longer than the chosen τ^+ in both dimensions: $\ell_{\text{step}}(\tau_j) \geq \ell_{\text{step}}(\tau^+) + 5$ *and* $\ell_{\text{token}}(\tau_j) \geq 1.5\,\ell_{\text{token}}(\tau^+)$.
- **Longer and Incorrect**: Incorrect responses that exceed τ^+ in both step and token length: $\ell_{\text{step}}(\tau_j) > \ell_{\text{step}}(\tau^+)$ *and* $\ell_{\text{token}}(\tau_j) > \ell_{\text{token}}(\tau^+)$.

This ensures that rejected samples are unambiguously inferior in both efficiency and correctness.

Direct Preference Optimization

We adopt Direct Preference Optimization (DPO) combined with a supervised fine-tuning (SFT) loss to optimize the model's reasoning path towards task-specific optimality criteria (conciseness and correctness). The joint objective function is defined as:

$$\begin{aligned}\mathcal{L}_{\text{DPO+SFT}} = \mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}[-\log\sigma(\beta\log\frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} \\ - \beta\log\frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)})] \\ + \lambda\mathbb{E}_{(x,y_w)\sim\mathcal{D}}\left[-\log\pi\theta(y_w|x)\right]\end{aligned} \tag{6}$$

where π_θ is the current policy model, π_{ref} is the reference model, y_w and y_l are the preferred and less-preferred responses, β scales the preference term, and λ controls the

Methods	AIME24			AIME25			AMC23			LiveCodeBench		
	ACC	Tokens	Chains	ACC	Tokens	Chains	ACC	Tokens	Chains	ACC	Tokens	Chains
DeepSeek-R1-Distilled-Qwen-7B												
Base Model	0.5479	7346	253	0.4229	6892	240	0.9047	5407	189	0.3127	3961	166
SFT	0.5437	6296	**199**	0.3979	5570	192	**0.9219**	3824	**120**	0.3344	3494	143
Kimi-1.5	0.5125	6144	306	0.3833	**5292**	254	0.8688	4265	237	0.1703	**1953**	**57**
DAST	0.5330	6337	-	-	-	-	-	-	-	-	-	-
DAST (reproduce)	0.5375	6909	244	0.4062	6595	242	0.8891	4774	160	0.3220	3370	136
ChainPrune	**0.5833**	**5688**	225	**0.4250**	**5351**	**189**	0.9125	**3771**	122	**0.3498**	3048	116
DeepSeek-R1-Distilled-Qwen-1.5B												
Base Model	0.3104	7015	225	0.2167	5248	158	0.7016	5316	167	0.1455	2661	90
SFT	0.2750	6549	219	0.2313	5471	171	0.7094	4859	152	0.1331	2370	83
Kimi-1.5	0.3021	**5497**	230	0.2354	4445	175	0.7203	3913	161	0.1455	2302	97
DAST (reproduce)	0.2667	5531	218	0.2250	**4010**	183	0.7125	3539	153	0.1455	**2043**	90
ChainPrune	**0.3292**	5542	**178**	**0.2458**	4079	**134**	**0.7250**	**2982**	**87**	**0.1734**	2246	**82**

Table 1: Performance comparison of ChainPrune against baseline methods across multiple reasoning tasks. Bold indicates the best result, underline indicates the second-best result. Token Length and Chain Length are averaged *only over correct answers*. In some cases, ChainPrune appears slightly worse than the best baseline on these metrics; this is because its higher accuracy means that more difficult problems are included in the averaging, which naturally increases the average token and chain length.

SFT regularization strength. The SFT term explicitly maximizes the likelihood of preferred outputs, preventing degeneration into overly terse or incomplete reasoning, while the DPO term implicitly aligns with preference signals. In our later experiments, we provide extensive empirical validation of the necessity of including the NLL term, and in the Appendix, we offer a theoretical analysis based on the *gradient entanglement* phenomenon (Yuan et al. 2024), which explains why margin-based methods like DPO and SimPO may inadvertently synchronize the log-probabilities of chosen and rejected responses.

Experiments

Experiment Setup

Long-COT Models: We evaluate two long-chain-of-thought models: **DeepSeek-R1-Distilled-Qwen-7B** and **DeepSeek-R1-Distilled-Qwen-1.5B**. Both are fully fine-tuned with a learning rate of 5.0×10^{-6} using cosine scheduling, 10% warmup, and trained on 8 $\times$ NVIDIA Tesla A100 GPUs.

Datasets: Training data is based on the cleaned MATH benchmark, containing 9,967 high-quality problem–answer pairs. For each problem, we sample 16 reasoning paths, each capped at 8,192 tokens. Evaluation covers **MATH500** (Hendrycks et al. 2021) (500 competition problems), **AMC23** (MAA 2024b) (30 high-school problems), **AIME24/25** (MAA 2024a) (60 olympiad-level problems), and **LiveCodeBench** (Jain et al. 2024) (programming problems from LeetCode, AtCoder, CodeForces, Aug 2024–Jan 2025; released_v5). All datasets use a maximum sequence length of 32,768 tokens.

Baselines: To systematically evaluate the performance of the methods, we compared the following baselines: (1) **Kimi-1.5** (Team et al. 2025): DPO dataset selects correct+shortest samples, rejects correct responses $\geq$ 1.5 longer or incorrect+longer ones; reproduced faithfully since the model is closed-source and differs in scale. (2) **DAST** (Shen et al. 2025a): dynamic token budget scoring with ranked contrastive pairs; reproduced from paper as **DAST (reproduce)** for reliability. (3) **Supervised Fine-Tuning (SFT)**: trained on Kimi-1.5 chosen samples.

Evaluation Metrics: We report **Accuracy**, **Token Length**, and **Chain Length** (number of reasoning steps to final answer). The latter two are computed only over correctly answered questions. Steps are segmented by `\n\n`, following the standard convention in Qwen models. Each experiment is run 16 times and averaged.

Main Results

Overall Performance We evaluate ChainPrune on two long-COT large reasoning models across four math benchmarks and one code-generation dataset. As shown in Table 1 and 2, ChainPrune consistently outperforms all baselines in accuracy while also reducing both token and chain length. These results demonstrate that our framework can improve reasoning efficiency without sacrificing correctness, and in many cases, even enhance it.

Accuracy: Across all datasets, ChainPrune achieves state-of-the-art accuracy, consistently matching or surpassing base model performance. This indicates that our pruning and preference optimization steps retain essential reasoning content and can guide the model toward more reliable reasoning patterns. By contrast, SFT shows slight degradation due to its lack of explicit efficiency constraints, while DAST (reproduce) and Kimi-1.5 exhibit noticeable accuracy drops.

Token Efficiency: ChainPrune achieves substantial token savings, averaging a 28.1% reduction over base models while maintaining or improving accuracy. For example, on AIME24 it reduces token usage by 22.6% and on LiveCodeBench by 23.0%. Token averages are computed only over correct answers; since ChainPrune solves more difficult problems, these are included in the average, which can slightly increase its reported token length compared to the

Methods	ACC	Tokens	All Tokens	Chains	All Chains	Faulty Reasoning	Invalid Reflection	Redundant Steps
Base Model	0.9240	3709	4621	111	146	102	124	132
SFT	0.9080	**1891**	2717	**52**	77	68	86	98
Kimi-1.5	0.9040	2777	4074	124	138	84	92	102
DAST	0.9260	2802	-	-	-	-	-	-
DAST (reproduce)	0.9180	3264	3839	99	123	92	104	107
ChainPrune	**0.9300**	2170	**2643**	62	**72**	**43**	**47**	**45**

Table 2: Reasoning path evaluation using LLM-as-a-Judge on MATH500. All methods are initialized from DeepSeek-R1-Distilled-Qwen-7B. Tokens and Chains report the average token count and reasoning step count only over correct responses. All Tokens and All Chains report the averages computed over all responses, including incorrect ones. Bold values indicate the best result, and underlined values indicate the second-best result.

Methods	AIME24		MATH500	
	ACC	Tokens	ACC	Tokens
Base Model	0.5479	7346	0.9240	3709
Kimi-1.5 w/ DPO	0.5125	6144	0.9040	2777
Kimi-1.5 w/ DPO + NLL Loss	0.5583	6077	0.9280	2373
DAST w/ SimPO	0.5375	6909	0.9180	3264
DAST w/ DPO + NLL Loss	0.5833	6188	0.9280	2572
ChainPrune	**0.5833**	**5688**	**0.9300**	**2170**

Table 3: Comparative analysis of different optimization methods under identical training datasets.

best baseline on that metric. Nevertheless, ChainPrune remains the best-performing method in most datasets, even under this stricter evaluation.

Chain Length: While Kimi-1.5 explicitly optimizes for the shortest token usage, ChainPrune not only achieves shorter or comparable tokens but also produces substantially shorter reasoning chains. This shows that our method effectively reduces redundancy in both dimensions, rather than sacrificing one for the other. Across all benchmarks, ChainPrune maintains concise chains without harming reasoning quality, further validating the effectiveness of our joint token–step optimization strategy.

Evaluate with LLM-as-a-Judge To assess reasoning quality beyond accuracy, we note that raw step counts measured via "`\n\n`" delimiters cannot fully capture whether shorter chains preserve correctness and coherence. Therefore, we adopt a confidence-calibrated **LLM-as-a-Judge** framework, further verified by human checks for reliability. Specifically, we use three diverse judge models: **GPT-o1**, **DeepSeek-R1**, and **Qwen-QwQ**, to evaluate outputs along three fine-grained dimensions: faulty reasoning, invalid reflection, and Redundant steps. Each judgment is aggregated via majority voting to mitigate individual model biases, and cases with high disagreement are manually reviewed.

As shown in Table 2, **ChainPrune** achieves the lowest rates across all dimensions—43 faulty reasoning steps, 47 invalid reflections, and 45 redundant steps—corresponding to reductions of 57.8%, 62.1%, and 65.9% over the base model. In contrast, the base model exhibits severe deficiencies, especially in Redundant Steps (132 steps), highlighting the necessity of systematic reasoning optimization. These results confirm that ChainPrune's structured semantic pruning—merging semantically equivalent nodes via low-cost semantic similarity—substantially cuts invalid reflections and redundancy while preserving core logical integrity. Further evaluation details are provided in the Appendix.

Analysis and Discussion

Comparison of different variants of DPO in Short-Chosen Preference Optimization

We compared DPO (Rafailov et al. 2023) and SimPO (Meng, Xia, and Chen 2024) in a short-chosen preference optimization setting, where preference pairs (y_w, y_l) were sampled from the reference model and y_w was shorter in token length than y_l. However, the narrow distribution of such datasets and the small edit distance between y_w and y_l often lead these methods to adopt biased strategies. Specifically, we observed a reward synchronization collapse effect: instead of improving y_w relative to y_l, the model simultaneously lowers the probabilities of both, degrading reasoning performance across benchmarks.

As shown in Table 3, incorporating an additional NLL loss term into DPO mitigates this issue, consistently maintaining or improving accuracy on mathematical reasoning tasks while reducing token usage. Notably, DPO+NLL improves the accuracy of both Kimi-1.5 and DAST on AIME24 and MATH500. However, in terms of token efficiency, ChainPrune remains superior. This advantage stems from our data construction pipeline: by merging reasoning paths at the semantic level and selecting Pareto-optimal chains, we obtain reasoning paths with fewer tokens, shorter chains, and complete logical structure.

Analysis of Reasoning Path Impact and Sample Efficiency

We analyzed the shortest correct responses (by token length) across varying difficulty levels (Figure 4). Token length grows proportionally with problem difficulty—harder problems require longer minimal correct responses. Thus, selecting the shortest correct response naturally adapts optimization to question difficulty.

Ablation studies further support this: shortest responses consistently yielded the most concise outputs, with medium-

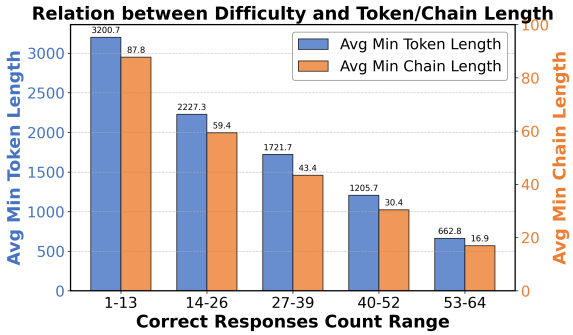

Figure 4: Relationship between difficulty and the token/chain length of the shortest correct response.

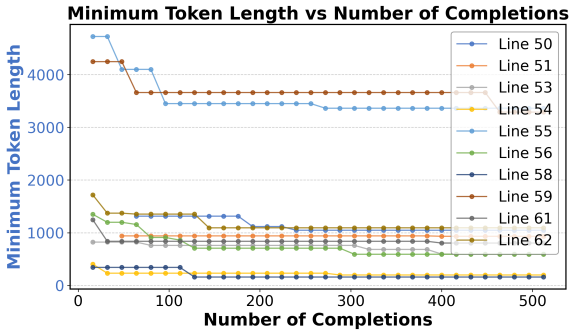

Figure 5: Sample efficiency analysis showing marginal token reduction after extensive sampling.

length responses producing longer outputs, and long responses the longest. This shows a clear linear relationship—shorter chosen samples lead to shorter, equally accurate model outputs. Detailed results are in the Appendix.

However, current RL methods such as DPO and PPO are sample-inefficient: finding sufficiently short correct responses via random sampling is costly. As shown in Figure 5, hundreds of samples yield only marginal token-length gains, making high-quality training data expensive to obtain.

Our method addresses this by merging multiple reasoning paths into a tree structure, enabling efficient generation of high-quality training samples with minimal overhead. As a result, this low-cost approach could be directly integrated into online RL frameworks such as PPO, improving the exploration efficiency for collecting valuable training data.

Related Works

Make Long CoT Short: Researchers have explored multiple directions to compress reasoning paths while maintaining accuracy. These efforts can be broadly categorized into four key approaches. First, RL with length penalties (Team et al. 2025; Luo et al. 2025; Shen et al. 2025a; Hou et al. 2025; Aggarwal and Welleck 2025; Li et al. 2024; Yang, Lin, and Yu 2025) has emerged as an effective strategy to encourage concise reasoning. Methods like O1-Pruner (Luo et al. 2025) optimize both accuracy and brevity by incorporating length constraints into reward functions, while DAST (Shen et al. 2025a) dynamically adjusts reasoning steps based on problem difficulty. Second, SFT with variable-length CoTs (Xia et al. 2025; Yu et al. 2024; Kang et al. 2025; Cui et al. 2025; Munkhbat et al. 2025; Han et al. 2024; Yang et al. 2025) trains models to generate shorter reasoning paths. TokenSkip (Xia et al. 2025) identifies and skips less critical tokens, while C3oT (Kang et al. 2025) leverages LLMs like GPT-4 (Achiam et al. 2023) to compress reasoning steps. Third, prompt-driven efficiency enhancement (Renze and Guven 2024; Xu et al. 2025b; Chen et al. 2024a; Lee, Che, and Peng 2025; Aytes, Baek, and Hwang 2025; Chuang et al. 2025b,a) guides models toward concise reasoning without training. Techniques like Concise CoT (Renze and Guven 2024) use simple instructions, while Break the Chain encourages shortcut reasoning. Finally, latent reasoning (Deng et al. 2023; Shen et al. 2025b; Zhang et al. 2025; Cheng and Van Durme 2024; Liu et al. 2024; Saunshi et al. 2025; Hao et al. 2024) eliminates explicit CoT generation. Implicit-KD (Deng et al. 2023) distills reasoning into hidden states, while Coconut performs reasoning in a continuous latent space.

Guide-based Methods: To accelerate model inference speed, researchers have proposed guide-based reasoning approaches (Yao et al. 2023; Hao et al. 2023; Wang et al. 2025; Xie et al. 2023). Gao et al. (Gao et al. 2024) pioneered the direct integration of conventional speculative decoding techniques with reasoning methodologies. Building upon this foundation, SEED (Wang et al. 2024) introduced a scheduled speculative decoding framework that coordinates multiple parallel small models through a single shared large model, further enhancing efficiency. The SpecSearch framework (Wang et al. 2025) introduces a novel dual-level speculative generator, operating at both coarse-grained thought and fine-grained token levels, achieving accelerated performance without compromising output quality.

Conclusion

Our work systematically addresses the critical challenge of ”pseudo-conciseness” in LLM reasoning optimization, where superficial token reduction fails to eliminate redundant reasoning paths. The proposed ChainPrune method pioneers a semantics-driven compression approach, merging equivalent nodes in reasoning trees to achieve genuine conciseness while preserving logical integrity. Experiments across five reasoning tasks demonstrate ChainPrune’s ability to reduce reasoning steps by 26.8% and tokens by 28.1% without accuracy degradation, while significantly improving reasoning quality, reducing faulty steps, invalid reflections, and redundant steps by 57.8%, 62.1% and 65.9% respectively compared to the base model. These results establish new standards for evaluating reasoning efficiency, emphasizing semantic coherence over mere token compression. While the current study focuses on offline optimization, the proposed approach holds strong potential for online RL training frameworks like PPO to enhance exploration efficiency. Due to time and resource constraints, we have not yet implemented this extension, which represents an important direction for future work.

References

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Aggarwal, P.; and Welleck, S. 2025. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*.

Arora, D.; and Zanette, A. 2025. Training Language Models to Reason Efficiently. *arXiv preprint arXiv:2502.04463*.

Aytes, S. A.; Baek, J.; and Hwang, S. J. 2025. Sketch-of-thought: Efficient llm reasoning with adaptive cognitive-inspired sketching. *arXiv preprint arXiv:2503.05179*.

Browne, C. B.; Powley, E.; Whitehouse, D.; Lucas, S. M.; Cowling, P. I.; Rohlfshagen, P.; Tavener, S.; Perez, D.; Samothrakis, S.; and Colton, S. 2012. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1): 1–43.

Chen, Q.; Qin, L.; Wang, J.; Zhou, J.; and Che, W. 2024a. Unlocking the capabilities of thought: A reasoning boundary framework to quantify and optimize chain-of-thought. *Advances in Neural Information Processing Systems*, 37: 54872–54904.

Chen, X.; Xu, J.; Liang, T.; He, Z.; Pang, J.; Yu, D.; Song, L.; Liu, Q.; Zhou, M.; Zhang, Z.; et al. 2024b. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*.

Cheng, J.; and Van Durme, B. 2024. Compressed chain of thought: Efficient reasoning through dense representations. *arXiv preprint arXiv:2412.13171*.

Chuang, Y.-N.; Yu, L.; Wang, G.; Zhang, L.; Liu, Z.; Cai, X.; Sui, Y.; Braverman, V.; and Hu, X. 2025a. Confident or Seek Stronger: Exploring Uncertainty-Based On-device LLM Routing From Benchmarking to Generalization. *arXiv preprint arXiv:2502.04428*.

Chuang, Y.-N.; Zhou, H.; Sarma, P.; Gopalan, P.; Boccio, J.; Bolouki, S.; and Hu, X. 2025b. Learning to route llms with confidence tokens. *arXiv preprint arXiv:2410.13284*, 3.

Coulom, R. 2006. Efficient selectivity and backup operators in Monte-Carlo tree search. In *International conference on computers and games*, 72–83. Springer.

Cui, Y.; He, P.; Zeng, J.; Liu, H.; Tang, X.; Dai, Z.; Han, Y.; Luo, C.; Huang, J.; Li, Z.; et al. 2025. Stepwise perplexity-guided refinement for efficient chain-of-thought reasoning in large language models. *arXiv preprint arXiv:2502.13260*.

Deng, Y.; Prasad, K.; Fernandez, R.; Smolensky, P.; Chaudhary, V.; and Shieber, S. 2023. Implicit chain of thought reasoning via knowledge distillation. *arXiv preprint arXiv:2311.01460*.

Gao, Z.; Niu, B.; He, X.; Xu, H.; Liu, H.; Liu, A.; Hu, X.; and Wen, L. 2024. Interpretable contrastive monte carlo tree search reasoning. *arXiv preprint arXiv:2410.01707*.

Guo, D.; Yang, D.; Zhang, H.; Song, J.; Zhang, R.; Xu, R.; Zhu, Q.; Ma, S.; Wang, P.; Bi, X.; et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Han, T.; Wang, Z.; Fang, C.; Zhao, S.; Ma, S.; and Chen, Z. 2024. Token-budget-aware llm reasoning. *arXiv preprint arXiv:2412.18547*.

Hao, S.; Gu, Y.; Ma, H.; Hong, J. J.; Wang, Z.; Wang, D. Z.; and Hu, Z. 2023. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*.

Hao, S.; Sukhbaatar, S.; Su, D.; Li, X.; Hu, Z.; Weston, J.; and Tian, Y. 2024. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*.

Hendrycks, D.; Burns, C.; Kadavath, S.; Arora, A.; Basart, S.; Tang, E.; Song, D.; and Steinhardt, J. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Hou, B.; Zhang, Y.; Ji, J.; Liu, Y.; Qian, K.; Andreas, J.; and Chang, S. 2025. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning. *arXiv preprint arXiv:2504.01296*.

Jaech, A.; Kalai, A.; Lerer, A.; Richardson, A.; El-Kishky, A.; Low, A.; Helyar, A.; Madry, A.; Beutel, A.; Carney, A.; et al. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Jain, N.; Han, K.; Gu, A.; Li, W.-D.; Yan, F.; Zhang, T.; Wang, S.; Solar-Lezama, A.; Sen, K.; and Stoica, I. 2024. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*.

Kang, Y.; Sun, X.; Chen, L.; and Zou, W. 2025. C3ot: Generating shorter chain-of-thought without compromising effectiveness. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 24312–24320.

Lee, A.; Che, E.; and Peng, T. 2025. How Well do LLMs Compress Their Own Chain-of-Thought? A Token Complexity Approach. *arXiv preprint arXiv:2503.01141*.

Li, Y.; Yuan, P.; Feng, S.; Pan, B.; Wang, X.; Sun, B.; Wang, H.; and Li, K. 2024. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning. *arXiv preprint arXiv:2401.10480*.

Li, Z.-Z.; Zhang, D.; Zhang, M.-L.; Zhang, J.; Liu, Z.; Yao, Y.; Xu, H.; Zheng, J.; Wang, P.-J.; Chen, X.; et al. 2025. From system 1 to system 2: A survey of reasoning large language models. *arXiv preprint arXiv:2502.17419*.

Liang, X.; Song, S.; Zheng, Z.; Wang, H.; Yu, Q.; Li, X.; Li, R.-H.; Wang, Y.; Wang, Z.; Xiong, F.; et al. 2024. Internal consistency and self-feedback in large language models: A survey. *arXiv preprint arXiv:2407.14507*.

Liao, B.; Xu, Y.; Dong, H.; Li, J.; Monz, C.; Savarese, S.; Sahoo, D.; and Xiong, C. 2025. Reward-Guided Speculative Decoding for Efficient LLM Reasoning. *arXiv preprint arXiv:2501.19324*.

Liu, T.; Chen, Z.; Liu, Z.; Tian, M.; and Luo, W. 2024. Expediting and Elevating Large Language Model Reasoning via Hidden Chain-of-Thought Decoding. *arXiv preprint arXiv:2409.08561*.

Luo, H.; Shen, L.; He, H.; Wang, Y.; Liu, S.; Li, W.; Tan, N.; Cao, X.; and Tao, D. 2025. O1-Pruner: Length-Harmonizing

Fine-Tuning for O1-Like Reasoning Pruning. *arXiv preprint arXiv:2501.12570*.

Ma, X.; Wan, G.; Yu, R.; Fang, G.; and Wang, X. 2025. CoT-Valve: Length-Compressible Chain-of-Thought Tuning. *arXiv preprint arXiv:2502.09601*.

MAA. 2024a. American Invitational Mathematics Examination (AIME). *Mathematics Competition Series*.

MAA. 2024b. American mathematics competitions (AMC 10/12). *Mathematics Competition Series*.

Meng, Y.; Xia, M.; and Chen, D. 2024. Simpo: Simple preference optimization with a reference-free reward. *Advances in Neural Information Processing Systems*, 37: 124198–124235.

Munkhbat, T.; Ho, N.; Kim, S. H.; Yang, Y.; Kim, Y.; and Yun, S.-Y. 2025. Self-training elicits concise reasoning in large language models. *arXiv preprint arXiv:2502.20122*.

Pang, R. Y.; Yuan, W.; He, H.; Cho, K.; Sukhbaatar, S.; and Weston, J. 2024. Iterative reasoning preference optimization. *Advances in Neural Information Processing Systems*, 37: 116617–116637.

Rafailov, R.; Sharma, A.; Mitchell, E.; Manning, C. D.; Ermon, S.; and Finn, C. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36: 53728–53741.

Renze, M.; and Guven, E. 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)*, 476–483. IEEE.

Saunshi, N.; Dikkala, N.; Li, Z.; Kumar, S.; and Reddi, S. J. 2025. Reasoning with latent thoughts: On the power of looped transformers. *arXiv preprint arXiv:2502.17416*.

Shen, Y.; Zhang, J.; Huang, J.; Shi, S.; Zhang, W.; Yan, J.; Wang, N.; Wang, K.; and Lian, S. 2025a. Dast: Difficulty-adaptive slow-thinking for large reasoning models. *arXiv preprint arXiv:2503.04472*.

Shen, Z.; Yan, H.; Zhang, L.; Hu, Z.; Du, Y.; and He, Y. 2025b. Codi: Compressing chain-of-thought into continuous space via self-distillation. *arXiv preprint arXiv:2502.21074*.

Team, K.; Du, A.; Gao, B.; Xing, B.; Jiang, C.; Chen, C.; Li, C.; Xiao, C.; Du, C.; Liao, C.; et al. 2025. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.

Wang, Z.; Wang, J.; Pan, J.; Xia, X.; Zhen, H.; Yuan, M.; Hao, J.; and Wu, F. 2025. Accelerating Large Language Model Reasoning via Speculative Search. *arXiv preprint arXiv:2505.02865*.

Wang, Z.; Wu, J.; Lai, Y.; Zhang, C.; and Zhou, D. 2024. Seed: Accelerating reasoning tree construction via scheduled speculative decoding. *arXiv preprint arXiv:2406.18200*.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35: 24824–24837.

Xia, H.; Li, Y.; Leong, C. T.; Wang, W.; and Li, W. 2025. Tokenskip: Controllable chain-of-thought compression in llms. *arXiv preprint arXiv:2502.12067*.

Xie, Y.; Kawaguchi, K.; Zhao, Y.; Zhao, X.; Kan, M.-Y.; He, J.; and Xie, Q. 2023. Decomposition enhances reasoning via self-evaluation guided decoding. *arXiv preprint arXiv:2305.00633*, 2.

Xu, F.; Hao, Q.; Zong, Z.; Wang, J.; Zhang, Y.; Wang, J.; Lan, X.; Gong, J.; Ouyang, T.; Meng, F.; et al. 2025a. Towards Large Reasoning Models: A Survey of Reinforced Reasoning with Large Language Models. *arXiv preprint arXiv:2501.09686*.

Xu, S.; Xie, W.; Zhao, L.; and He, P. 2025b. Chain of draft: Thinking faster by writing less. *arXiv preprint arXiv:2502.18600*.

Yang, J.; Lin, K.; and Yu, X. 2025. Think When You Need: Self-Adaptive Chain-of-Thought Learning. *arXiv preprint arXiv:2504.03234*.

Yang, W.; Ma, S.; Lin, Y.; and Wei, F. 2025. Towards thinking-optimal scaling of test-time compute for llm reasoning. *arXiv preprint arXiv:2502.18080*.

Yao, S.; Yu, D.; Zhao, J.; Shafran, I.; Griffiths, T.; Cao, Y.; and Narasimhan, K. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36: 11809–11822.

Yeo, E.; Tong, Y.; Niu, M.; Neubig, G.; and Yue, X. 2025. Demystifying Long Chain-of-Thought Reasoning in LLMs. *arXiv preprint arXiv:2502.03373*.

Yu, P.; Xu, J.; Weston, J.; and Kulikov, I. 2024. Distilling system 2 into system 1. *arXiv preprint arXiv:2407.06023*.

Yuan, H.; Zeng, Y.; Wu, Y.; Wang, H.; Wang, M.; and Leqi, L. 2024. A Common Pitfall of Margin-based Language Model Alignment: Gradient Entanglement. *arXiv preprint arXiv:2410.13828*.

Zhang, J.; Zhu, Y.; Sun, M.; Luo, Y.; Qiao, S.; Du, L.; Zheng, D.; Chen, H.; and Zhang, N. 2025. Lightthinker: Thinking step-by-step compression. *arXiv preprint arXiv:2502.15589*.

Appendix

A Statistical Analysis of Reasoning Path Efficiency

To verify the hypothesis that token-level optimization in preference learning often overlooks the compactness of reasoning structures, we conducted a comprehensive analysis of all math questions in the training set. For each question, we examined sampled correct reasoning paths to assess the alignment—or misalignment—between minimal token length and minimal chain length. This analysis quantitatively supports our claim in subsection "Challenges in RL-Optimized Reasoning Path" regarding the issue of pseudo-conciseness.

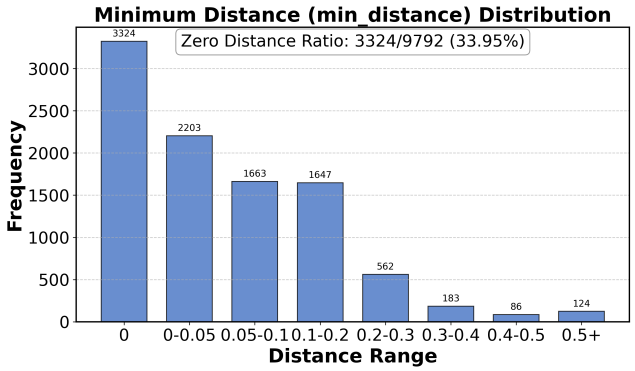

Figure 6: Distribution of `min_distance` across all questions. The metric quantifies the normalized discrepancy between the shortest-token and shortest-chain correct reasoning paths. Only 33.95% of the questions have `min_distance` equal to zero, indicating that the same path minimizes both token length and chain length.

Evaluation Metrics and Key Findings

For each question, we sampled multiple correct reasoning paths and computed two key metrics for each path:

- **Token Length**: the total number of output tokens.
- **Chain Length**: the number of reasoning steps in the logical chain.

We then ranked all paths by each metric to identify:

- $\tau^{\text{min-tok}}$: the path with the shortest token length.
- $\tau^{\text{min-step}}$: the path with the fewest reasoning steps.

To measure the discrepancy between these two optima, we define a normalized distance metric:

$$\text{min\_distance} = \frac{1}{2}\left(\frac{|\tau_{\text{token}}^{\text{min-step}} - \min_T|}{\max_T - \min_T} + \frac{|\tau_{\text{step}}^{\text{min-tok}} - \min_S|}{\max_S - \min_S}\right). \quad (7)$$

where $\tau_{\text{token}}^{\text{min-step}}$ is the token length of the shortest-step path, and $\tau_{\text{step}}^{\text{min-tok}}$ is the step count of the shortest-token path. $\min_T$, $\max_T$, $\min_S$, and $\max_S$ denote the minimum and maximum values over all paths for the respective metric. This metric captures how far apart the two optima are in their non-primary dimensions. A value of min_distance $= 0$ indicates that both minima are achieved by the same path.

As shown in Figure 6, our analysis over the entire training set revealed that only **33.95%** of questions have min_distance $= 0$, i.e., where $\tau^{\text{min-tok}} = \tau^{\text{min-step}}$. This result highlights a significant misalignment between expression-level and structural conciseness in current optimization schemes.

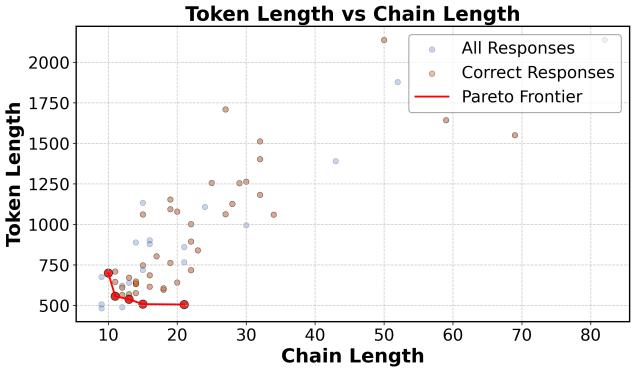

Figure 7: Token length versus chain length for 64 sampled reasoning paths on a representative math question. Each point denotes a unique path. Blue nodes indicate **wrong responses**, visually distinguishing them from correct ones. No single path lies at the lower-left corner, suggesting that none optimally balances both expression-level and structural conciseness.

Pareto Frontier Visualization

To further illustrate the misalignment between structural and expression-level conciseness, we randomly selected a representative math question from the training set and sampled 64 reasoning paths. Each path is visualized in a 2D space, with reasoning chain length on the x-axis and token length on the y-axis, as shown in Figure 7.

Each point represents a unique reasoning path. Importantly, no correct path appears in the bottom-left corner of the plot, indicating that joint minimality is not achieved in both metrics. This visual evidence reinforces our central hypothesis: current reward heuristics that aim to minimize token length often fail to promote reasoning paths that are simultaneously compact in structure and concise in expression.

B Examples of Semantically Equivalent Reasoning Steps in Chosen and Rejected Paths

Our analysis of large reasoning model (LRM) reasoning paths reveals a counterintuitive phenomenon: semantically equivalent intermediate steps coexist in both "chosen" and

Example 1

Instruction: Rachelle uses 3 pounds of meat to make 8 hamburgers for her family. How many pounds of meat does she need to make 24 hamburgers for a neighbourhood picnic?

Chosen: <think>First, determine the amount of meat per hamburger by dividing the total meat used by the number of hamburgers.

Rachelle uses 3 pounds of meat for 8 hamburgers, so the meat per hamburger is: 3 pounds / 8 hamburgers = 0.375 pounds per hamburger.

Next, calculate the total meat needed for 24 hamburgers by multiplying the meat per hamburger by the number of hamburgers: 0.375 pounds/hamburger * 24hamburgers = 9 pounds.

.....

Rejected: <think>First, determine the amount of meat required to make one hamburger by dividing the total meat used by the number of hamburgers: 3 pounds ÷ 8 hamburgers = 0.375 pounds per hamburger.

Next, calculate the total meat needed for 24 hamburgers by multiplying the meat per hamburger by the number of hamburgers: 0.375 pounds / hamburger * 24 hamburgers = 9 pounds.

.....

Figure 8: Example 1: A comparison between the chosen and rejected reasoning paths for a proportional reasoning problem. Although both paths share the same semantic steps—computing meat per hamburger and scaling to 24 hamburgers—the rejected path performs these steps earlier.

"rejected" paths, yet systematically emerge at earlier positions in the latter. This suggests a consistent structural bias in the ordering of reasoning content across preference pairs.

In this section, we present concrete examples that illustrate this behavior and trace it to conventional dataset construction methods, which prioritize minimal token length when selecting "chosen" responses. As shown in Figure 8 and Figure 9, even when both paths reach the correct final answer and share nearly identical intermediate reasoning steps, the path with more delayed semantic content—and thus lower token count per step—is systematically favored.

This bias results in a counterintuitive preference for syntactically fragmented yet token-efficient reasoning, at the cost of structural clarity and efficiency.

C Multi-Reasoning Paths Merging Algorithm

In this section, we provide the implementation details of the multi-reasoning path merging algorithm (Algorithm 1). The algorithm takes as input a set of K candidate reasoning paths $\{\tau_1, ..., \tau_K\}$ generated by a base LLM and constructs a compact, step-level reasoning tree T through hierarchical merging. The goal is to retain all critical reasoning logic while pruning redundancy across paths. The algorithm operates in three main phases:

- **Initialization Phase**: Each path τ_i is first split into a sequence of reasoning steps S_i. The shortest path $S_{\min}$ is selected as the backbone for tree construction, and its steps are used to build an initial linear tree T.
- **Tree Construction Phase**: A chain of nodes is created from $S_{\min}$, where each node represents one reasoning step and is attached sequentially to form a linear tree structure. This backbone captures a minimal but coherent reasoning trajectory.
- **Path Merging Phase**: Each of the remaining paths $S_i \neq S_{\min}$ is incrementally merged into the tree. For each step u in S_i, we compute its embedding and search for semantically similar nodes in the tree using cosine similarity. If the similarity exceeds a threshold θ_{sim}, the step is considered a merge candidate. To ensure logical consistency, we further verify that substituting u with the candidate node v does not significantly increase the predictive entropy of the model. Specifically, the entropy difference ΔH before and after replacement must be within a tolerance threshold ϵ. If both conditions are satisfied, the remaining steps in S_i are attached as a subtree to the matched node. If no valid merge is found, the entire path is appended as a new branch from the root.

The resulting tree structure T reflects shared reasoning segments as internal nodes, divergent reasoning strategies as branches, and unique answers as leaf nodes. This compact representation allows ChainPrune to preserve logical diversity while eliminating redundant computation.

D Theoretical Analysis of different variants of DPO in Short-Chosen Preference Optimization

In this section, we theoretically analyze the decline in reasoning performance observed in the Short-Chosen Preference Optimization scenario for DPO and SimPO. Preference optimization methods like DPO and SimPO, which are margin-based, often lead to synchronized increases or decreases in the log probabilities of the chosen response y_w and the rejected response y_l, denoted as $\log \pi_w$ and $\log \pi_l$. Previous work (Yuan et al. 2024) has found that the gradient entanglement effect causes this synchronization. Specifically:

Example 2

Instruction: Solve for the sum of all possible values of x when $3^{x^2+4x+4} = 9^{x+2}$.

Chosen: <think>Alright, so I have this equation to solve: $3^{x^2+4x+4} = 9^{x+2}$.

Hmm, okay, let me think about how to approach this. I remember that when dealing with exponential equations, it's often helpful to express both sides with the same base. Since 9 is a power of 3, that seems like a good starting point.

First, I'll rewrite 9 as 3^2. So, the right side of the equation becomes $(3^2)^{x+2}$. Using the exponentiation rule $(a^b)^c = a^{bc}$, this simplifies to $3^{2(x+2)}$. Now, the equation looks like this:

$$3^{x^2+4x+4} = 3^{2(x+2)}$$

Since the bases are the same and the equation holds true, the exponents must be equal. Therefore, I can set the exponents equal to each other:

$$x^2 + 4x + 4 = 2(x + 2)$$

Alright, let me simplify this equation step by step. First, distribute the 2 on the right side:

$$x^2 + 4x + 4 = 2x + 4$$

Next, subtract (2x + 4) from both sides to bring all terms to one side:

$$x^2 + 4x + 4 - 2x - 4 = 0$$

Simplify by combining like terms:

$$x^2 + 2x = 0$$

Now, I can factor out an x from the left side:

$$x(x + 2) = 0$$

Setting each factor equal to zero gives the potential solutions:

.....

Rejected: <think>To solve the equation $3^{x^2+4x+4} = 9^{x+2}$, I start by expressing both sides with the same base. Since $9 = 3^2$, the equation becomes:

$$3^{x^2+4x+4} = (3^2)^{x+2}$$

Applying the exponentiation rule $(a^m)^n = a^{mn}$, this simplifies to:

$$3^{x^2+4x+4} = 3^{2(x+2)}$$

Now that both sides have the same base, I can set the exponents equal to each other:

$$x^2 + 4x + 4 = 2(x + 2)$$

Expanding the right side:

$$x^2 + 4x + 4 = 2x + 4$$

Subtracting $2x + 4$ from both sides to set the equation to zero:

$$x^2 + 2x = 0$$

Factoring out x:

$$x(x + 2) = 0$$

Setting each factor equal to zero gives the potential solutions:

.....

Figure 9: Example 2: A comparison of chosen and rejected reasoning paths for an exponential equation. Both paths apply identical mathematical transformations, including rewriting powers of 9, applying exponent rules, and solving a quadratic equation. However, the rejected path introduces semantically equivalent steps earlier in the trajectory.

Algorithm 1 Multi-Reasoning Paths Merging Algorithm

Require:
paths: Set of reasoning paths $\{\tau_1, ..., \tau_n\}$ (<think> τ_i </think>)
θ_{sim}: similarity threshold for semantic merging
ϵ: entropy threshold to ensure logical consistency during merging
Ensure:
Merged tree structure T
Initialization:
Split each path P_i into step sequence $S_i \leftarrow \text{SPLITINTOSTEPS}(\tau_i)$
Select shortest path $S_{\min} \leftarrow \arg\min_{S_i} |S_i|$
origin_depth $\leftarrow |S_{\min}|$
Build initial linear tree T from $S_{\min}$
Build Initial Tree:
for $k = 1$ **to** $|S_{\min}|$ **do**
  Create node v_k with content $S_{\min}[k]$
  Attach v_k as child of v_{k-1} (or root if $k = 1$)
end for
Process Other Paths:
for each path $S_i \neq S_{\min}$ **do**
  merged $\leftarrow$ FALSE
  for $t = 1$ **to** $|S_i|$ **do**
    $u \leftarrow S_i[t]$
    Compute embedding $\mathbf{e}_u$
    Find candidates $C \leftarrow \{v \mid \text{Sim}(\mathbf{e}_u, \mathbf{e}_v) \geq \theta_{\text{sim}}\}$
    if $C \neq \emptyset$ **then**
      Sort C by depth (desc), similarity (desc)
      for each $v^* \in C$ **do**
        Replace u with v^* in S_i and compute entropy ΔH
        if $\Delta H \leq \epsilon$ **then**
          Merge: attach remaining steps $S_i[t+1:]$ as subtree to v^*
          merged $\leftarrow$ TRUE
          break
        end if
      end for
      if *merged* **then**
        break
      end if
    end if
  end for
  if not merged **then**
    Attach S_i as new branch to root
  end if
end for
return T

Gradient Entanglement Mechanism: The optimization objective of DPO aims to widen the gap between the log probability of the chosen response $\log \pi_w$ and that of the rejected response $\log \pi_l$, requiring both an increase in $\log \pi_w$ and a decrease in $\log \pi_l$. The gradient update direction can be expressed as:

$$\Delta\theta \propto d_w \nabla \log \pi_w - d_l \nabla \log \pi_l$$

where, $\nabla \log \pi_w$ and $\nabla \log \pi_l$ are the gradient directions of the log probabilities for y_w and y_l, respectively. d_w and d_l are the derivative weights of the loss function concerning $\log \pi_w$ and $\log \pi_l$ (in DPO, $d_w/d_l = 1$).

The synchronized movement of $\log \pi_w$ and $\log \pi_l$ occurs when the inner product $\langle \nabla \log \pi_w, \nabla \log \pi_l \rangle$ between these gradients becomes large relative to their individual norms. This can manifest in two problematic scenarios:

- Synchronized **Increase** ($\log \pi_w \uparrow, \log \pi_l \uparrow$): Occurs when $\|\nabla \log \pi_l\|^2 \leq \langle \nabla \log \pi_w, \nabla \log \pi_l \rangle \leq \|\nabla \log \pi_w\|^2$. Here, the gradients are so strongly aligned that both probabilities increase, failing to suppress y_l adequately. This is particularly detrimental in safety-critical tasks where harmful responses (y_l) must be actively discouraged.
- Synchronized **Decrease** ($\log \pi_w \downarrow, \log \pi_l \downarrow$): Arises when $\|\nabla \log \pi_w\|^2 \leq \langle \nabla \log \pi_w, \nabla \log \pi_l \rangle \leq \|\nabla \log \pi_l\|^2$. In this case, the model simultaneously forgets both good and bad behaviors, which explains the observed decline in reasoning performance when distilling from high-quality y_w responses.

For DPO ($\frac{d_w}{d_l} = 1$), the ideal divergence condition ($\log \pi_w \uparrow, \log \pi_l \downarrow$) requires:

$$\langle \nabla \log \pi_w, \nabla \log \pi_l \rangle \leq \min(\|\nabla \log \pi_w\|^2, \|\nabla \log \pi_l\|^2)$$

Through its length-normalization design ($\frac{d_w}{d_l} = \frac{|y_l|}{|y_w|}$), SimPO reformulates this condition as a more lenient inequality:

$$\left\langle \frac{\nabla \log \pi_w}{|y_w|}, \frac{\nabla \log \pi_l}{|y_l|} \right\rangle \leq \left\| \frac{\nabla \log \pi_w}{|y_w|} \right\|^2,$$
$$\left\langle \frac{\nabla \log \pi_w}{|y_w|}, \frac{\nabla \log \pi_l}{|y_l|} \right\rangle \leq \left\| \frac{\nabla \log \pi_l}{|y_l|} \right\|^2$$

We conduct an empirical study by sampling 1,000 preference pairs from the dataset and tracking the optimization dynamics of $\log \pi_w$ and $\log \pi_l$ during training. The results, summarized in a bar chart (Figure 10), reveal three possible conditions of gradient entanglement: Condition 1 (synchronized decrease: $\log \pi_w \downarrow, \log \pi_l \downarrow$), Condition 2 (synchronized increase: $\log \pi_w \uparrow, \log \pi_l \uparrow$), and Condition 3 (ideal divergence: $\log \pi_w \uparrow, \log \pi_l \downarrow$). Key observations highlight DPO's dominance of undesirable conditions, with Condition 1 (synchronized decrease) accounting for 75.4% of cases, explaining the reasoning performance decline due to "forgetting" high-quality responses, while Condition 2 (synchronized increase) occurs in 21.2% of cases, reflecting failure to suppress long responses. The ideal Condition 3 is rare (3.4%), underscoring DPO's susceptibility to gradient entanglement.

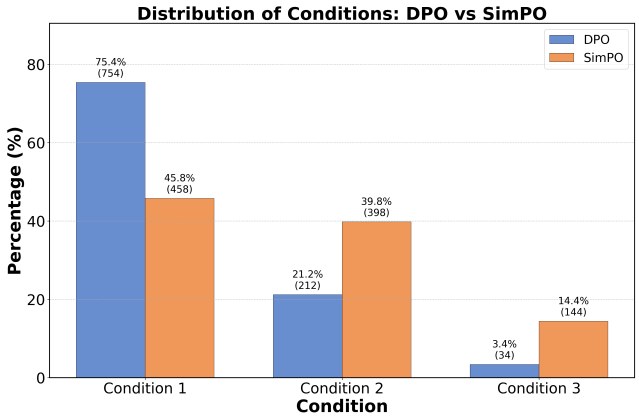

Figure 10: The figure shows three gradient entanglement conditions in DPO/SimPO training: Condition 1 (synchronized decrease: $\log \pi_w \downarrow, \log \pi_l \downarrow$) causing forgetting, Condition 2 (synchronized increase: $\log \pi_w \uparrow, \log \pi_l \uparrow$) failing to suppress bad outputs, and Condition 3 (ideal divergence: $\log \pi_w \uparrow, \log \pi_l \downarrow$); SimPO reduces but doesn't eliminate suboptimal Conditions 1-2 (85.6% combined).

SimPO's length-normalized design partially mitigates this issue by relaxing the divergence inequality, yet it still exhibits an 85.6% combined prevalence of Conditions 1 and 2, indicating residual entanglement. This suggests that merely normalizing gradients by response length is insufficient to fully decouple the dynamics of $\log \pi_w$ and $\log \pi_l$, leaving room for further improvement in disentangling optimization pathways.

DPO combined with NLL Loss, breaks the gradient entanglement effect in standard DPO through explicit regularization. The NLL Loss increases the coefficient d_w for the chosen gradient while leaving the coefficient d_l for the rejected gradient unaffected, thereby relaxing the condition for increasing the log probability of the chosen response. Simultaneously, the incorporation of the NLL term creates a dynamic balance in the optimization objective: it preserves the contrastive learning advantage of DPO (where the chosen response is favored over the rejected one) while explicitly maximizing the probability of the chosen response, thus avoiding the issue where purely margin-based methods might lead to a decrease in the probability of the chosen response.

E LLM-as-a-Judge for Evaluation of Reasoning Model

We present an improved *LLM-as-a-Judge* framework for evaluating the fine-grained quality of reasoning steps in multi-step inference tasks. Unlike prior approaches that relied on fixed heuristic segmentation (`\n\n`) and direct long-context evaluations, our framework enhances precision and scalability through two major innovations: (1) dynamic step segmentation using a large language model as a chain splitter, and (2) a multi-model voting protocol for robust step-level classification.

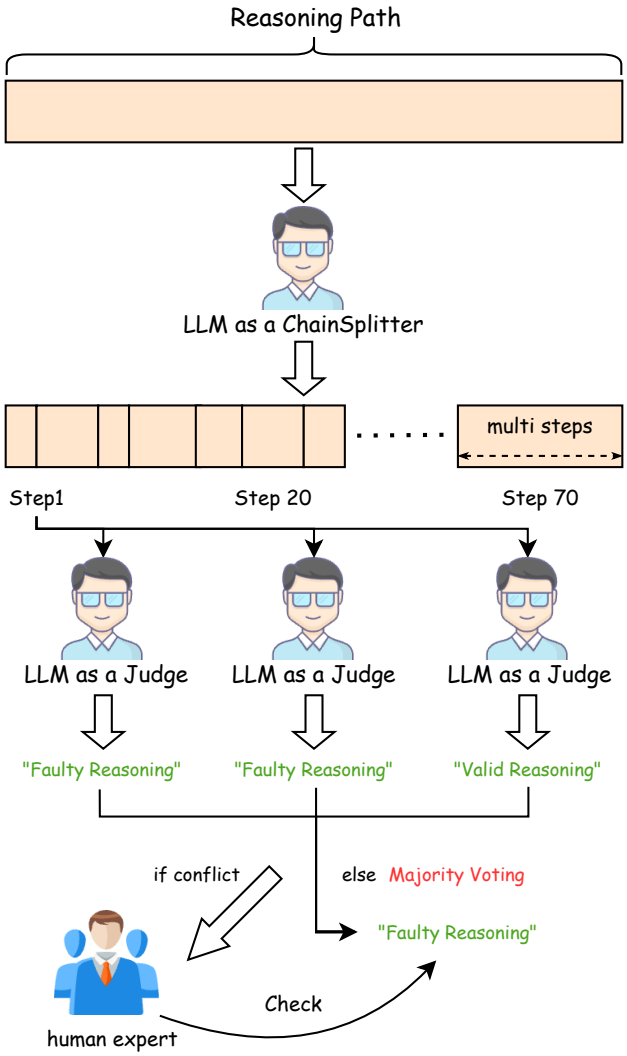

Figure 11: The overall pipeline of the LLM-as-a-Judge framework. The reasoning path is first segmented by GPT-4o acting as a chain splitter. Then, each reasoning step is independently evaluated by three expert models across three categories: faulty reasoning, invalid reflection, and redundant steps. Results are aggregated via majority voting or deferred to human reviewers in cases of high disagreement.

Methods	AIME24			AIME25			AMC23			MATH500		
	ACC	Tokens	Chains	ACC	Tokens	Chains	ACC	Tokens	Chains	ACC	Tokens	Chains
Longest	0.5667	6576	240	0.4167	5555	202	0.9094	4027	137	0.9260	2427	73
Middle	0.5833	6471	235	0.4083	5441	194	0.9141	3958	129	0.9200	2376	67
Shortest	**0.5833**	**5688**	**225**	**0.4250**	**5351**	**189**	**0.9125**	**3771**	**122**	**0.9300**	**2170**	**62**

Table 4: Correlation between chosen sample length and optimized model's output length

As shown in Figure 11, our framework follows a two-stage pipeline consisting of step segmentation and multi-model evaluation. In the first stage, given a full reasoning path, we employ GPT-4o to segment the path into semantically meaningful steps. This LLM-as-a-Chainsplitter strategy avoids the brittleness of rule-based chunking and ensures that each step reflects a coherent and minimal unit of reasoning. In the second stage, each segmented step is independently evaluated by three expert reasoning models—**GPT-o1**, **DeepSeek-R1**, and **Qwen-QwQ**—along three critical dimensions: **faulty reasoning**, **invalid reflection**, and **redundant steps**. The final label for each step is determined via majority voting; in cases of high inter-model disagreement, a human expert is consulted.

The classification criteria are as follows: (1) *Faulty reasoning* refers to logically incorrect or mathematically invalid steps that jeopardize correctness; (2) *Invalid reflection* denotes ineffective or incoherent meta-cognitive steps that fail to improve or critically assess the reasoning process; (3) *Redundant steps* capture repetitive or non-contributory steps that offer no new information. This categorization schema supports precise error attribution and reflects key weaknesses in reasoning traceability.

To ensure label accuracy and interpretability, we design a structured prompt that provides the current step along with up to three preceding and three following steps as context. As shown in Figure 12, the model is instructed to return a JSON object containing the predicted tag and a justification. This context-aware prompt enables evaluators to distinguish, for example, between truly novel steps and those that are redundant or flawed only in relation to their neighbors.

F More Experiment Results

The ablation experiments in Table 4 demonstrated a clear trend: shorter chosen samples consistently led to more optimized model outputs, with both reduced token counts and reasoning chains. Specifically, when comparing responses of varying lengths (shortest, medium (5th-shortest reasoning path), and longest (10th-shortest reasoning path)) for the same questions, the shortest samples produced the most concise outputs (e.g., 2,170 tokens vs. 2,427 for the longest in MATH500) while maintaining or even improving accuracy. This suggests that shorter responses not only enhance efficiency by reducing output length but also streamline reasoning chains. The linear correlation held across datasets, with middle-length samples yielding intermediate results, indicating that response brevity is associated with improved computational efficiency.

LLM-as-a-Judge Prompt

Reasoning Step Classification

Role
You are a professional mathematics evaluator with expertise in assessing logical reasoning processes. Your task is to classify a single reasoning step into one of four predefined categories. The reasoning step is an atomic unit extracted from a student's multi-step problem-solving trace.

Classification Categories:
Choose exactly one of the following categories:
- <Valid Reasoning Step>: The step is logically sound, mathematically correct, and contributes meaningfully to the solution.
- <Faulty Reasoning Step>: The step contains logical, factual, or mathematical errors that undermine the correctness of the reasoning.
- <Invalid Reflection Step>: The step attempts reflection or correction but does so ineffectively or inconsistently, such as continuing flawed reasoning or offering non-constructive commentary.
- <Redundant Reasoning Step>: The step repeats previous reasoning, adds no useful content, or elaborates unnecessarily on previously valid steps.

Input Format:
Problem: {query}
Reasoning Context:
Previous Steps:
1. {step A}
2. {step B}
3. {step C}
Current Step: {response_step}
Next Steps:
1. {step D}
2. {step E}
3. {step F}

Output Requirements:
Return JSON format with the following structure
{
"step_id": <int>,
"content": "Preserve ALL original text"
"tag": "<TAG>",
"reason": "Explain why this tag is appropriate for the current step, based on its content and context."
}

Example Output:
{
"step_id": 5,
"content": "We already found that x = 6 by solving 2x = 12 in the previous step. Therefore, x = 6 is the solution."
"tag": "<Redundant Reasoning Step>",
"reason": "This step repeats the equation solving already completed in step 4 without introducing new insight."
}

Figure 12: The structured prompt design for the LLM-as-a-Judge step-level classification task. The prompt includes the problem statement, the current reasoning step, and up to three preceding and three following steps as context. The model is instructed to classify the target step into one of four categories and output a JSON object including the classification tag and an explanation.