Efficient Reasoning for LLMs through Speculative Chain-of-Thought

Anonymous ACL submission

Abstract

Large reasoning language models such as OpenAI-o1 and Deepseek-R1 have recently attracted widespread attention due to their impressive task-solving abilities. However, the enormous model size and the generation of lengthy thought chains introduce significant reasoning costs and response latency. Existing methods for efficient reasoning mainly focus on reducing the number of model parameters or shortening the chain-of-thought length. In this paper, we introduce Speculative Chain-of-Thought (SCoT), which reduces reasoning latency from another perspective by accelerated average reasoning speed through large and small model collaboration. SCoT conducts thought-level drafting using a lightweight draft model. Then it selects the best CoT draft and corrects the error cases with the target model. The proposed thinking behavior alignment improves the efficiency of drafting and the draft selection strategy maintains the prediction accuracy of the target model for complex tasks. Experimental results on GSM8K, MATH, GaoKao, CollegeMath and Olympiad datasets show that SCoT reduces reasoning latency by 48%∼66% and 21%∼49% for Deepseek-R1-Distill-Qwen-32B and Deepseek-R1-Distill-Llama-70B while achieving near-target-model-level performance.

1 Introduction

Large-scale reasoning language models (e.g., OpenAI-o1 (Jaech et al., 2024) and Deepseek-R1 (DeepSeek-AI et al., 2025)) have exhibited exceptional performance across a broad range of reasoning scenarios. Their superior performance stems from two key factors: massive model scale (e.g., DeepSeek-R1 with 671B parameters) and long chain-of-thought (CoT) (Wei et al., 2023) generation (reaching over 10,000 tokens for complex tasks). However, both factors incur substantial computational overhead, resulting in huge inference costs. In reasoning systems where the model's reasoning process remains implicit, excessive reasoning time induces significant response latency.

Recent works (Paliotta et al., 2025; Liao et al., 2025; Xu et al., 2025b; Zhang et al., 2025) mainly tackle this challenge by developing smaller reasoning models or reducing the generation length of CoTs. Distillation (Paliotta et al., 2025; Liao et al., 2025) serves as a commonly used technique for transferring strong reasoning capabilities from large reasoning models to small ones. Representatively, Deepseek-R1-Distill-Qwen-1.5B (DeepSeek-AI et al., 2025) exhibits robust performance on diverse datasets. Approaches such as StepSkip (Liu et al., 2024) and TokenSkip (Xia et al., 2025) suggest training reasoning models on compressed CoT data to shorten the CoT length for inference. In practical applications, reasoning models need to solve problems with widely varying complexity levels. Fundamentally, employing large reasoning models to handle simple tasks is a waste. On the other hand, using small models alone or compressing the CoT length may compromise performance on complex tasks. Now that we have a powerful and efficient small model, why not prioritize this lightweight alternative and only invoke the large model for deeper thinking when the task exceeds the capability threshold of the small one?

Based on this motivation, we propose Speculative Chain-of-Thought (SCoT), a reasoning framework that reduces reasoning latency through the collaboration of large and small models. The core idea of SCoT is to use a lightweight model to efficiently generate multiple CoT drafts which are then picked and used by the target model. SCoT is inspired by speculative decoding (Stern et al., 2018; Xia et al., 2023; Leviathan et al., 2023), which performs token-level speculation, whereas our approach conducts thought-level speculation. Unlike shortening the length of CoT, SCoT accelerates the thinking process from another dimension, that is, by increasing the speed of CoT generation.

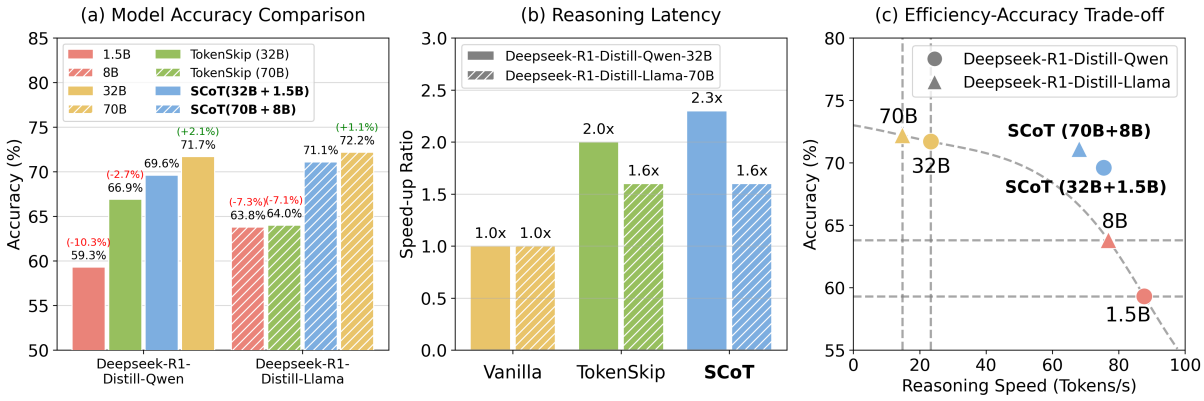

Figure 1: Average reasoning accuracy and efficiency of SCoT with Deepseek-R1-Distill-Qwen-32B and Deepseek-R1-Distill-Llama-B on GSM8K, MATH, GaoKao, CollegeMath and Olympiad. Graph (a): SCoT achieves near-target-model performance level and outperforms TokenSkip on average generation accuracy on the 5 datasets. Graph (b): SCoT demonstrates an average speed-up ratio of 2.3× and 1.6× for the two target models on reasoning latency. Graph (c): Comparison of reasoning speed and accuracy for models of different sizes. Note that all the experiments for testing the reasoning speed are conducted on H800-80G GPUs. SCoT exhibits a better reasoning efficiency-accuracy trade-off through large and small model collaboration.

In Section 3, we introduce thinking behavior alignment to improve drafting efficiency. We also design an error correction mechanism to preserve accuracy in dealing with complex tasks. We construct a training set by 500 samples from GSM8K to efficiently instruct-tuning the target model to improve the capability of draft selection and error detection. We conduct comprehensive experiments in Section 4 to evaluate the effectiveness of SCoT. Figure 1 displays the overview reasoning accuracy and efficiency of SCoT. SCoT reduces reasoning latency by 48%∼66% and 21%∼49% for Deepseek-R1-Distill-Qwen-32B and Deepseek-R1-Distill-Llama-70B on 5 datasets (including the Olympiad-level dataset (He et al., 2024)) while maintaining near-target-model-level performance. It achieves better reasoning accuracy compared with TokenSkip (Xia et al., 2025) under similar speed-up ratios. Adapting to task difficulty and draft quality, the draft selection mechanism decides when the target model should rethink. Simple tasks mainly utilize the draft model for thought chain generation, while complex problems engage the target model more extensively in the reasoning process. On the other hand, the single-pass drafting and selection paradigm of SCoT significantly reduces the model reloading overhead between high-bandwidth memory and on-chip cache inherent in traditional speculative decoding methods, maintaining high efficiency even in long-context scenarios.

To conclude, we introduce an efficient reasoning method in this paper. Our key contributions are:

- We propose a novel efficient reasoning approach that employs a small reasoning model for CoT drafting and the target model for CoT selection and correction. It integrates the computational efficiency of small draft models with the high generation accuracy of large target models.
- The proposed framework demonstrates a difficulty-aware adaptive mechanism where thought chains are primarily generated by the small model for simple tasks, while complex problems trigger the allocation of more thought chains to the target model for reliable outputs.
- Our approach exhibits a better reasoning efficiency-accuracy trade-off in extensive experiments across tasks with varying difficulty levels. It also demonstrates superior acceleration performance compared to standard speculative decoding methods with the same draft models.

2 Related Works

Efficient Reasoning Reasoning LLMs (Jaech et al., 2024; DeepSeek-AI et al., 2025) generate lengthy CoTs to enhance performance, albeit at the cost of increased latency. Some approaches (Ding et al., 2024; Luo et al., 2025; Xia et al., 2025; Xu et al., 2025b; Zhang et al., 2025) speed up the thinking process by reducing the length of the CoT at the token level. Liu et al. (2024) and Xia et al. (2025) use compressed CoTs to fine-tune the reasoning LMs. Hou et al. (2025) prunes long CoTs through reinforcement learning. Xu et al. (2025a) and Aytes

et al. (2025) guide the model to think efficiently through prompting. Soft CoT methods (Saunshi et al., 2025; Xu et al., 2025b; Hao et al., 2024) use latent space representations to replace reasoning tokens, thus reducing the excessive redundancy in the reasoning process. Another mainstream approach (Yu et al., 2024; Paliotta et al., 2025; Liao et al., 2025) for efficient reasoning is to teach smaller models to think. Paliotta et al. (2025) and Liao et al. (2025) transfer reasoning ability from large models to small ones by distillation. Liu et al. (2025) study the impact of quantization for reasoning LLMs. Our approach leverages collaborative inference between large and small models, combining the small model's efficiency with the large model's accuracy to achieve an improved efficiency-accuracy trade-off in reasoning LLMs.

Speculative Decoding Autoregressive language models generate one token in one decoding step, which greatly limits efficiency. Speculative decoding (Stern et al., 2018; Xia et al., 2023; Chen et al., 2023a) adopts lightweight models for drafting and target models for parallel verification to generate multiple tokens in one step, thus achieving lossless acceleration. The draft model can be an independent small model (Leviathan et al., 2023; Chen et al., 2023a; Spector and Re, 2023; Chen et al., 2023b), a submodule of the target model (Cai et al., 2024; Zhang et al., 2023; Li et al., 2024), or an external corpus (He et al., 2023). Draft tokens that match the original output of the target model will be accepted under a strict verification strategy. Most speculative decoding methods perform token-level drafting, while we borrow the idea of speculation to perform thought-level drafting to improve the reasoning efficiency of LLMs.

3 Speculative CoT

This section introduces Speculative Chain-of-Thought (SCoT), an algorithm for accelerating LLM reasoning. SCoT employs a small model to draft reasoning chains while utilizing the target model to select the best draft for generating the final answer. It requires only a minimal amount of instruct tuning. The overview is shown in Figure 3.

3.1 Formulation

For question q in question set Q, the reasoning model M first enters the thinking stage after inputting q. Specifically, M will output a chain-of-thought $T = (y_1, y_2, ..., y_l)$ starting and ending with pre-trained special tokens, where y_i represents the i-th output token and l is length of T. Take Deepseek-R1 (DeepSeek-AI et al., 2025) as an example. The thinking process starts with "<*think*>" and ends with "</*think*>". Next, the model generates the final answer A based on q and T. Our goal is to reduce the inference overhead of T while ensuring the quality of the final answer.

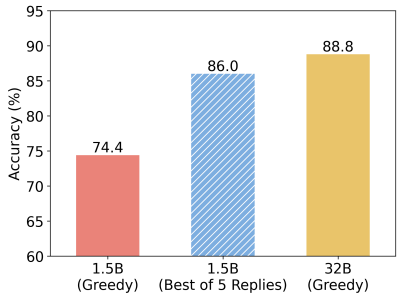

Figure 2: Comparison of reasoning accuracy for different sampling methods on GSM8K.

3.2 Generating Drafts of Thoughts

First, we start with a question: "*Can small models generate high-quality chain-of-thoughts?*". We sampled 500 samples in GSM8K (Cobbe et al., 2021) to evaluate the reasoning accuracy of Deepseek-R1-Distill-Qwen-1.5B and Deepseek-R1-Distill-Qwen-32B (DeepSeek-AI et al., 2025). Figure 2 displays the results. Under greedy sampling, the performance of the 1.5B model is significantly worse than that of the 32B model. However, if we allow the 1.5B model to provide 5 answers under nuclear sampling and select the best one among the 5 replies for each sample, its overall accuracy on GSM8K is close to that of the 32B model.

From the above experiments, we can conclude that the small model has the potential to generate high-quality thought chains, but a stronger verifier is needed to control or select its generated results. Therefore, in order to accelerate the thinking process of reasoning LLMs, we first use a smaller model M_d to draft n chains of thoughts in parallel:

$$[T_d^1, T_d^2, ..., T_d^n] = M_d([\underbrace{q, q, ..., q}_{n}]). \quad (1)$$

We use temperature sampling for the small model thinking to enhance the diversity of drafts of CoTs. Thanks to their smaller parameter size, smaller models can draft reasoning chains significantly faster than large models with the same CoT length, thus speeding up the thinking process.

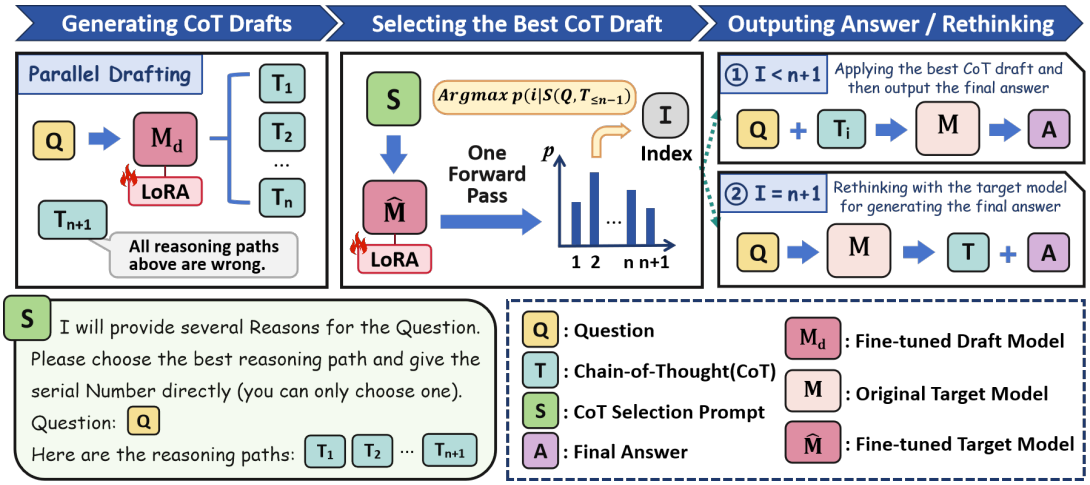

Figure 3: Overview of Speculative Chain-of-Thought. Given a question Q, SCoT first applies a lightweight draft model to generate multiple CoT drafts in parallel. The draft model is fine-tuned with LoRA modules to align the thinking behavior of the target model. It appends a special CoT option for the case where all drafts are wrong. The target model for selecting the best CoT draft is also fine-tuned with LoRA modules for improved accuracy. With the designed prompt template S, only one forward propagation is needed to get the index of the best CoT. Once the best CoT is picked, SCoT directly adopts it for the original target model to generate the final answer. Only if no draft is selected, SCoT will rethink with the target model to ensure the quality of the generated answer.

3.3 Thinking Behavior Alignment

Although small models have higher throughput than large ones, differences in their reasoning patterns may still result in limited, or even reduced, reasoning speed. We observed in experiments that even models of the same series can exhibit significant differences in thinking behavior. The inference speed of Deepseek-R1-Distill-Qwen-1.5B is about 6 times that of Deepseek-R1-Distill-Qwen-32B. However, the average CoT length of the 1.5B model on the GSM8K dataset is more than 4 times that of the 32B model. The CoT of the 1.5B model contains more redundant tokens and repeated reflections. Therefore, it requires training in alignment of thinking behavior to ensure drafting efficiency.

We adopt 1500 samples from the GSM8K training set to make the target model generate the corresponding CoT. The training data $d = (x_1, x_2, ..., x_m, y_1, y_2, ..., y_l)$ is a combination of the question and CoT, where m is the length of the question and $x_{\leq m}$ are input tokens. Then we use LoRA modules (Hu et al., 2022) to instruct-tune the draft model with the cross-entropy loss:

$$\mathcal{L}_{Draft} = -\frac{1}{l}\sum_{i=1}^{l} \log p_{M_d}(y_i|x_{\leq m}, y_{<i}), \quad (2)$$

where $p_{M_d}(\cdot|\cdot)$ represents the output probability distribution of M_d. The original parameters of the draft model are frozen during training. The LoRA modules are adapted to the $\boldsymbol{Q}$ and $\boldsymbol{V}$ matrices in the attention block of each layer. After training, we merge the LoRA module parameters into the weight matrices of the draft model. Therefore, there is no additional overhead during inference.

3.4 Draft Selection and Error Correction

After generating CoT drafts $(T_1, T_2, ..., T_n)$ by the fine-tuned draft model, we design a prompt template S for selecting these thought chains. Figure 3 displays the specific content of this template. For particularly challenging problems where all CoT drafts may be incorrect, we aim for the target model to detect such cases to ensure the accuracy of the final answer. For this purpose, we introduce a special option T_{n+1}, which indicates that all CoT drafts are wrong. Through a single forward propagation, we can efficiently obtain the CoT index with the highest probability from the output distribution of the next token by the target model:

$$index = \arg\max_{i \in \mathcal{V}} P_M(i|S(q, T_{\leq n+1})), \quad (3)$$

where $\mathcal{V} = \{1, 2, ..., n+1\}$. To improve the accuracy of the target model in selecting the correct CoT drafts and detecting errors, we also use the LoRA modules to fine-tune the target model. We use 500 samples from the GSM8K training set to construct the training data. For each sample, we use the draft model to generate n thought chains

and deploy template S as the input. We judge the correctness of the CoT drafts based on the ground truth and construct the label set $\mathcal{Y}$, which is the set of indices for all correct drafts. See Appendix for examples. We design the following loss function to fine-tune the target model:

$$\mathcal{L}_{Target} = \min\{-\log p_M(y|S(q, T_{\leq n+1}))|y \in \mathcal{Y}\}. \tag{4}$$

Likewise, the original model parameters are frozen, and parameters in LoRA modules are merged into the weight matrices of $\boldsymbol{Q}$ and $\boldsymbol{V}$. The fine-tuned target model is only used for CoT selection.

There are two situations when selecting CoT. The first is that the output $index < n + 1$, that is, the target model selects the best one T_{index} among the CoTs generated by the draft model. In this case, we directly let the target model generate the final answer based on q and T_{index}. The second situation is the output $index = n + 1$, which indicates that the problem is too hard and the draft model failed to produce correct reasoning. To maintain performance in the face of these complex problems, we instruct the target model to rethink the question and generate the final answer.

4 Experiments

4.1 Settings

We adopt Deepseek-R1-Distill-Qwen-32B (DeepSeek-AI et al., 2025) and Deepseek-R1-Distill-Llama-70B (abbreviated as D-Qwen-32B and D-Llama-70B) as the target models. We employ Deepseek-R1-Distill-Qwen-1.5B and Deepseek-R1-Distill-Llama-8B as the draft models for them, respectively. We conducte the two groups of experiments under different hardware environments. We use a single A100-PCIE-40GB GPU for the 1.5B model and 4 A100-PCIE-40GB GPUs for the 32B model. While we deploy the 8B model on a single H800-80GB GPU and the 80B model on 4 H800-80GB GPUs. We evaluate the efficiency and accuracy of Speculative CoT on reasoning datasets of various difficulty levels, including GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), GaoKao-En-2023 (Liao et al., 2024), CollegeMath (Tang et al., 2024) and Olympiad (He et al., 2024). As stated in the last section, we use data from GSM8K training set to fine-tune the models with LoRA modules. Note that there is no overlap in the data used for training and evaluation.

For SCoT, we set the rank of LoRA modules to 8 for D-Qwen-1.5B/32B and D-Llama-8B, and we set it to 16 for D-Llama-70B. Learning rate is set to 5e-5 and 1e-4 for the draft models and target models, respectively. The number of CoT drafts n is set to 5. We set the temperature for sampling to 0.6 in CoT drafting process. To avoid running out of memory when selecting the drafts, the maximum draft length is set to 5000. We set the maximum length of CoT to 20480, which is sufficient for most samples in experiments. We compare the reasoning latency and accuracy of SCoT with the vanilla reasoning by the target models.

We also compare the performance of SCoT with TokenSkip (Xia et al., 2025), which trains the target model with compressed CoT data to shorten the mean CoT length. The original CoT data are generated by the original target model. Then it compresses the thought chains through LLMLingua-2 (Pan et al., 2024). We adopt the training set of GSM8K to construct the training data. To compare the accuracy of the two methods at similar acceleration ratios, we set the compression ratio to 0.5 for TokenSkip, which means that about half of the tokens in thought chains are dropped.

4.2 Main Results

Lower Reasoning Latency Table 1 demonstrates the average CoT length and the average CoT latency of each sample on the 5 datasets. l_M and l_{M_d} represent the numbers of tokens generated by the target model and the draft model during the thinking stage, respectively. Since the draft model generates 5 draft chains in parallel, l_{M_d} refers to the length of the longest draft CoT. The average latency t is measured in seconds. For the 32B model, l_M exhibits a wide range of 302 to 8431 across different datasets. Except for MATH where the average CoT length of SCoT ($l_M + l_{M_d}$) is shorter than that of the 32B model, the overall CoT lengths of the two methods are close in the Qwen group. However, SCoT generates more tokens during the reasoning phase in the Llama group, where the speed-up ratio is relatively lower. For SCoT, most of the reasoning tokens are generated by the draft model, and the CoT tokens generated by the target model account for 2.8%∼45.9% of the total CoT tokens. As task difficulty increases, small models exhibit higher error rates, thus necessitating greater involvement from the target model in the reasoning process. The draft selection mechanism dynamically determines whether the target model should

Method	GSM8K		MATH		GaoKao		CollegeMath		Olympiad	
	l_M	l_{M_d}	l_M	l_{M_d}	l_M	l_{M_d}	l_M	l_{M_d}	l_M	l_{M_d}
D-Qwen-32B	302	-	2123	-	3230	-	1051	-	8431	-
SCoT(32B+1.5B)	45	265	151	1855	630	2767	110	1136	3553	4194
D-Llama-70B	326	-	1492	-	2521	-	920	-	5503	-
SCoT(70B+8B)	47	410	70	2395	501	2986	120	1913	2579	4319
	t	r	t	r	t	r	t	r	t	r
D-Qwen-32B	26.5	1.00	225.1	1.00	369.5	1.00	95.3	1.00	1092.2	1.00
SCoT(32B+1.5B)	**11.7**	**2.26**	**77.0**	**2.92**	**160.1**	**2.31**	**43.7**	**2.18**	**575.7**	**1.90**
D-Llama-70B	21.6	1.00	105.5	1.00	188.3	1.00	63.3	1.00	443.6	1.00
SCoT(70B+8B)	**11.0**	**1.96**	**57.8**	**1.83**	**111.0**	**1.70**	**50.4**	**1.26**	**325.0**	**1.36**

Table 1: Comparison of CoT length and reasoning efficiency. We report average CoT length of the target model l_M, average CoT length of the draft model l_{M_d} and average CoT latency of each sample t(sec). Speed-up ratio r is calculated by average thinking time compared with vanilla decoding.

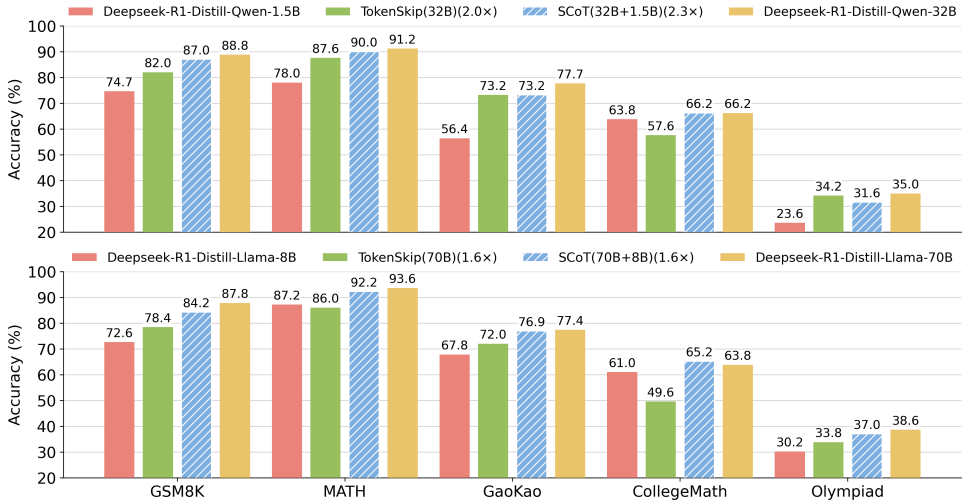

Figure 4: Comparison of accuracy of the final answer with D-Qwen (top) and D-Llama (bottom). The compression ratio for TokenSkip is set to 0.5. The ratio in the legends represents the latency speed-up for TokenSkip and SCoT.

reconsider its output, evaluating both problem complexity and draft generation quality. For simpler tasks, thought chains are predominantly produced by the draft model. Conversely, the proposed mechanism allocates more thought chain generation to the target model when handling complex problems.

The throughput of the draft model is about 5 times that of the corresponding target model for both groups in our experimental environment. The CoT latency of SCoT is mainly composed of the target model reasoning time, draft model reasoning time, and draft selection time. For the GSM8K dataset in the Qwen group, the proportions are approximately 33.9%, 60.2%, and 5.9% respectively. With longer generated CoTs (e.g., on Olympiad), draft selection incurs minimal overhead. Generally, SCoT achieves greater speed-up ratios on simpler tasks where the CoT drafts demonstrate higher quality. It achieves a speed-up ratio ranging from 1.26× to 2.92× compared to vanilla reasoning.

Near-target-model Level Accuracy Figure 4 displays the accuracy of different methods for Deepseek-R1-Distill-Qwen and Deepseek-R1-Distill-Llama on these datasets. Comparative results indicate that SCoT maintains near-target-model performance levels on all datasets (even for the Olympiad-level task), with clearly superior accuracy compared to standalone draft model reasoning. For CollegeMath, SCoT even achieves a better accuracy compared to the target model. Moreover, SCoT outperforms TokenSkip in generation accuracy on most groups except Olympiad with the Qwen model, even with higher speed-up ratios, exhibiting better accuracy-efficiency trade-off.

4.3 Ablation Studies

We conduct an ablation study in this section. We carry out 3 sets of experiments to observe

Method	GSM8K			MATH			GaoKao			CollegeMath			Olympiad		
	s	r'	r	s	r'	r	s	r'	r	s	r'	r	s	r'	r
Deepseek-R1-Distill- Qwen-32B (A100-PCIE-40GB)															
ARD	11.4	1.00	1.00	9.4	1.00	1.00	8.7	1.00	1.00	11.0	1.00	1.00	7.9	1.00	1.00
SD	18.5	1.63	1.87	13.4	1.42	1.41	12.1	1.38	1.32	15.6	1.42	1.34	9.1	1.16	0.59
SCoT	**23.9**	**2.10**	**2.26**	**25.4**	**2.69**	**2.92**	**19.1**	**2.18**	**2.31**	**27.9**	**2.53**	**2.18**	**11.2**	**1.42**	**1.90**
Deepseek-R1-Distill- Llama-70B (H800-80GB)															
ARD	14.5	1.00	1.00	14.1	1.00	1.00	13.4	1.00	1.00	14.5	1.00	1.00	12.7	1.00	1.00
SD	25.1	1.64	1.63	21.2	1.50	1.49	17.6	1.31	1.35	23.2	1.59	**1.61**	12.6	0.99	0.58
SCoT	**37.3**	**2.45**	**1.96**	**41.8**	**2.96**	**1.83**	**28.7**	**2.14**	**1.70**	**37.5**	**2.58**	1.26	**17.4**	**1.37**	**1.36**

Table 2: Comparison of reasoning efficiency with auto-regressive decoding (ARD) and speculative decoding (SD) on the five datasets with the two models. s represents the average throughput during the reasoning phase (including the prefilling phase). For SCoT, only valid tokens (generated by either the draft model or the target model) are included in the calculation of s. r' represents the speed-up ratio of throughput. r stands for the speed-up ratio of average reasoning latency of each sample.

Method	GSM8K	MATH	GaoKao	CollegeMath	Olympiad
SCoT (32B+1.5B)	**87.0**	**90.0**	**73.2**	**66.2**	**31.6**
w Single Draft	85.6	86.6	69.9	63.8	30.0
w/o Tuning M	85.6	**90.0**	69.9	64.8	30.0
w/o Error Correction	85.6	**90.0**	71.4	65.6	30.2

Table 3: Results for ablation study.

Method	GSM8K	MATH	GaoKao	CollegeMath	Olympiad
D-Qwen-1.5B	606	2092	2589	1355	4032
+ TBA	**193**	**1183**	**1487**	**709**	**3554**
D-Llama-8B	585	1979	3223	1285	6474
+ TBA	**258**	**1536**	**2253**	**1254**	**3783**

Table 4: Comparative analysis of average CoT draft length on D-Qwen-1.5B and D-Llama-8B with versus without thinking behavior alignment training (TBA).

the changes in accuracy on the 5 datasets with Deepseek-R1-Distill-Qwen: 1) SCoT with Single Draft: Only one CoT draft is generated for each sample (except for the special draft T_{n+1}), 2) SCoT without Tuning M: We use the original target model instead of the fine-tuned version to select the drafts, 3) SCoT without Error Correction: We remove T_{n+1} and the rethinking strategy. Table 3 shows the results. SCoT exhibits the highest accuracy on all datasets, indicating that both of the three proposed techniques contribute to the reasoning performance. Drafting multiple thought chains increases the upper limit of draft quality, thus enhancing the potential for higher-quality outputs. In preliminary experiments, we found that without fine-tuning, M never selects the "*All reasoning paths are wrong.*" option for any CoT draft candidates. After training, the target model can acquire error correction capability.

To verify the effectiveness of thinking behavior alignment, we compare the average CoT draft length generated by the original draft models and the fine-tuned versions. Table 4 shows the mean CoT length. The CoT length generated by the fine-tuned model is significantly shorter than that of the original model on all datasets for both the two draft models, especially for GSM8K. Therefore, thinking behavior alignment improves the efficiency of the drafting process.

4.4 Comparison with Speculative Decoding

We compare the reasoning efficiency of SCoT with speculative decoding with large and small models collaboration (Leviathan et al., 2023; Kim et al., 2023). Speculative decoding generates several tokens with the draft model and then verifies the candidates with the target model in each decoding step. The maximum draft length is set to 5.

Table 2 displays the results. The throughput of auto-regressive decoding is affected by the length of the sequence. Speculative decoding performs well when the sequence length is short. It achieves a reasoning latency speed-up ratio of up to 1.87 and 1.63 for the Qwen model and the Llama model, respectively. However, for the Olympiad dataset, where the average generation length exceeds 5000 tokens, the drafting overhead grows with sequence length, ultimately causing deceleration for both the target models. Note that speculative decoding may produce outputs that deviate from the original model when using half-precision. In certain cases (e.g., GSM8K dataset with the Qwen model), the shorter generation lengths can lead to $r' < r$. Employing the same draft models, SCoT achieves a higher throughput on all datasets. It shows a speed-up ratio of up to 2.96 for throughput. It also exhibits a better acceleration for reasoning la-

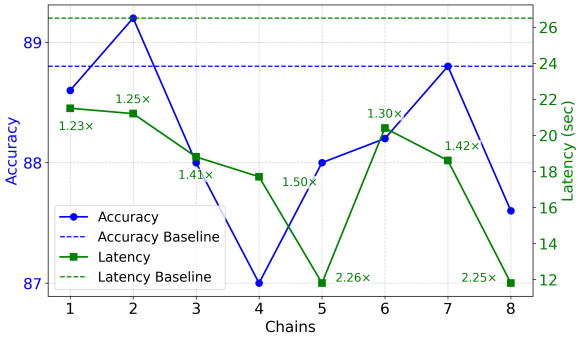

Figure 5: Impact of the number of draft chains.

tency (except for CollegeMath in the Llama group). While speculative decoding requires multiple verifications, SCoT's single draft-verify cycle significantly reduces the frequent model reloads between GPU high-bandwidth memory (HBM) and on-chip cache. Therefore, generation length exhibits a relatively minor impact on SCoT's efficiency (as stated in main results section the speed-up ratio of SCoT is mainly affected by the difficulty of the task), reflecting its good adaptability to long sequence generation scenarios.

4.5 Draft Selection Accuracy Analysis

We study the draft selection and error detection accuracy of the fine-tuned Deepseek-R1-Distill-Qwen-32B on the 500 samples in the test set of GSM8K. We divide these samples into two categories according to their labels. Samples for which the draft model can generate the correct reasoning chain form Class 1 (the label $y \neq n+1$), while others constitute Class 2 (the label $y = n+1$). There are 450 first-class samples out of the 500 samples. The prediction accuracy of Class 1 is 85.1%, and Class 2 is 52%. Since Class 2 samples are more difficult to predict and the training data is limited, the target model demonstrates more accurate prediction for Class 1 samples than Class 2 samples. Our preliminary tests show that direct prediction by the original target model on the second-class samples yields zero accuracy. With a minimal amount of training, the model can correctly predict the labels for about half of the Class 2 samples.

4.6 Impact of Number of Chains

To explore the best number of CoT draft candidates, we evaluate the reasoning accuracy and latency under different numbers of draft thought chains in this section. We conduct an experiment with the Deepseek-R1-Distill-Qwen models on the GSM8K dataset. We tested the performance when drafting 1 to 8 thought chains. Figure 5 displays the results. The blue and green lines represent the accuracy and reasoning latency, respectively. The two dotted lines show the baseline accuracy and latency of Deepseek-R1-Distill-Qwen-32B. Increasing the number of chains increases the probability of including high-quality reasoning paths in the set of drafts. When using 2 draft chains, while the latency reduction remains modest, SCoT surprisingly outperforms the target model in accuracy, demonstrating the remarkable potential of collaborative model generation. A higher speed-up ratio indicates that SCoT accepts more reasoning chains generated by the draft model. SCoT can reduce its reliance on the target model due to the increase in the number of candidates, thus achieving a higher speed-up ratio. However, as the number of chains increases, the difficulty of training the target model to select the correct draft also increases. Meanwhile, due to the inherent randomness in training procedures and the limited scale of training data, it exhibits fluctuations in reasoning accuracy. Moreover, an oversupply of chains complicates error detection for the target model, which may lead to accuracy degradation despite the improved speed. Overall, SCoT demonstrates optimal acceleration performance with 5 thought chains while maintaining acceptable accuracy degradation.

5 Conclusion

In this paper, we proposed a novel reasoning framework called Speculative Chain-of-Thought to alleviate the latency bottleneck of reasoning LLMs. The proposed method leverages the efficiency of small models for parallel CoT drafting and the higher accuracy of the large model for drafting selection and error correction. The draft selection mechanism adaptively decides whether the target model needs to rethink based on the difficulty of the problem and the quality of the draft generation. Thought chains are mostly generated by the draft model for simple tasks. While more thought chains will be allocated to the target model to generate for complex problems. Speculative Chain-of-Thought demonstrates a speed-up ratio of up to 2.92× for the reasoning process while achieving near-target-model-level accuracy on 5 datasets of different difficulties. It exhibits the potential for future work to realize a better accuracy-efficiency trade-off for reasoning LLMs through model collaboration.

Limitations

- SCoT adopts existing pre-trained reasoning models as draft models. However, for some target models, existing pre-trained models of the same family may not be available. In addition, the size of the draft model has a significant impact on the efficiency of reasoning. For example, in the experiments, the speedup ratio of SCoT on the Llama group is lower than that on the Qwen group. This is largely because the 8B draft model is still a bit large for the 70B target model. An overly large draft model may limit acceleration performance or even slow down reasoning. Future work can consider how to train a more effective draft model for any target model.

- SCoT needs to fine-tune the main model and the target model. In this paper, due to resource limitations, we only consider its performance on mathematical reasoning tasks. We used only limited training data. Its performance on other types of reasoning tasks may vary, especially for out-of-domain tasks.

- SCoT changes the original output of the target model, which has a certain impact on the original security of the pre-trained model. In practical applications, it may lead to harmful output. If used in real-world scenarios, the security of the inference framework needs to be reconsidered.

References

Simon A Aytes, Jinheon Baek, and Sung Ju Hwang. 2025. Sketch-of-thought: Efficient llm reasoning with adaptive cognitive-inspired sketching. *arXiv preprint arXiv:2503.05179*.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. 2024. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023a. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*.

Ziyi Chen, Xiaocong Yang, Jiacheng Lin, Chenkai Sun, Jie Huang, and Kevin Chen-Chuan Chang. 2023b. Cascade speculative drafting for even faster llm inference. *arXiv preprint arXiv:2312.11462*.

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*.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Mengru Ding, Hanmeng Liu, Zhizhang Fu, Jian Song, Wenbo Xie, and Yue Zhang. 2024. Break the chain: Large language models can be shortcut reasoners. *Preprint*, arXiv:2406.06580.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. 2024. Training large language models to reason in a continuous latent space. *Preprint*, arXiv:2412.06769.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850, Bangkok, Thailand. Association for Computational Linguistics.

Zhenyu He, Zexuan Zhong, Tianle Cai, Jason D Lee, and Di He. 2023. Rest: Retrieval-based speculative decoding. *arXiv preprint arXiv:2311.08252*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

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

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

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*.

Sehoon Kim, Karttikeya Mangalam, Suhong Moon, Jitendra Malik, Michael W. Mahoney, Amir Gholami, and Kurt Keutzer. 2023. Speculative decoding with big little decoder. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024. Eagle: Speculative sampling requires rethinking feature uncertainty. *arXiv preprint arXiv:2401.15077*.

Huanxuan Liao, Shizhu He, Yupu Hao, Xiang Li, Yuanzhe Zhang, Jun Zhao, and Kang Liu. 2025. SKIntern: Internalizing symbolic knowledge for distilling better CoT capabilities into small language models. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 3203–3221, Abu Dhabi, UAE. Association for Computational Linguistics.

Minpeng Liao, Wei Luo, Chengxi Li, Jing Wu, and Kai Fan. 2024. Mario: Math reasoning with code interpreter output – a reproducible pipeline. *Preprint*, arXiv:2401.08190.

Ruikang Liu, Yuxuan Sun, Manyi Zhang, Haoli Bai, Xianzhi Yu, Tiezheng Yu, Chun Yuan, and Lu Hou. 2025. Quantization hurts reasoning? an empirical study on quantized reasoning models. *Preprint*, arXiv:2504.04823.

Tengxiao Liu, Qipeng Guo, Xiangkun Hu, Cheng Jiayang, Yue Zhang, Xipeng Qiu, and Zheng Zhang. 2024. Can language models learn to skip steps? In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

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

Daniele Paliotta, Junxiong Wang, Matteo Pagliardini, Kevin Y. Li, Aviv Bick, J. Zico Kolter, Albert Gu, François Fleuret, and Tri Dao. 2025. Thinking slow, fast: Scaling inference compute with distilled reasoners. *Preprint*, arXiv:2502.20339.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, and 1 others. 2024. Llmlingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 963–981.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. 2025. Reasoning with latent thoughts: On the power of looped transformers. In *The Thirteenth International Conference on Learning Representations*.

Benjamin Spector and Chris Re. 2023. Accelerating llm inference with staged speculative decoding. *arXiv preprint arXiv:2308.04623*.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. 2018. Blockwise parallel decoding for deep autoregressive models. *Advances in Neural Information Processing Systems*, 31.

Zhengyang Tang, Xingxing Zhang, Benyou Wang, and Furu Wei. 2024. Mathscale: Scaling instruction tuning for mathematical reasoning. *Preprint*, arXiv:2403.02884.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Heming Xia, Tao Ge, Peiyi Wang, Si-Qing Chen, Furu Wei, and Zhifang Sui. 2023. Speculative decoding: Exploiting speculative execution for accelerating seq2seq generation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3909–3925.

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

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

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. 2025b. Softcot: Soft chain-of-thought for efficient reasoning with llms. *Preprint*, arXiv:2502.12134.

Ping Yu, Jing Xu, Jason Weston, and Ilia Kulikov. 2024. Distilling system 2 into system 1. *Preprint*, arXiv:2407.06023.

Jintian Zhang, Yuqi Zhu, Mengshu Sun, Yujie Luo, Shuofei Qiao, Lun Du, Da Zheng, Huajun Chen, and Ningyu Zhang. 2025. Lightthinker: Thinking step-by-step compression. *Preprint*, arXiv:2502.15589.

Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. 2023. Draft & verify: Lossless large language model acceleration via self-speculative decoding. *arXiv preprint arXiv:2309.08168*.

A SCoT for Heterogeneous Draft Model

In some practical scenarios, it may be difficult to obtain a small-scale model of the same structure as a target model. Even if a small-scale model is available, its parameter count may not be suitable for generating speculative cots. Therefore, we explored the effects of adopting an alternative heterogeneous draft model.

We tested on five datasets under the same settings as the main experiment, using llama70b as the target model and qwen1.5b as the draft model. The experimental results are shown in Table 5.

Method	GSM8K	MATH	GaoKao	CollegeMath	Olympiad
D-Qwen-1.5B	74.7	78.0	56.4	63.8	23.6
D-Llama-8B	72.6	87.2	67.8	61.0	30.2
D-Llama-70B	87.8	93.6	77.4	63.8	38.6
SCoT (70B+8B)	84.2	92.2	76.9	65.2	37.0
SCoT (70B+1.5B)	84.2	91.2	72.5	66.0	33.2

Table 5: Performance of SCoT with heterogeneous draft model.

The experimental results suggest that the performance of the draft model mainly affects the upper limit of Scot. If the performance of the draft model is limited, its performance on some more difficult tasks (e.g., Olympiad) will be restricted. For simpler tasks, the performance of the draft model has minimal impact on SCOT's effectiveness, where smaller draft models prove more beneficial for inference acceleration. Overall, SCoT performs well even with a heterogeneous draft model, which breaks through the reliance of standard speculative decoding on model alignment.

B Impact on Length of the Final Answer

We study the impact of SCoT on the length of the final answer. Table 6 displays the answer length on the 5 datasets. In addition to reducing the latency of the reasoning phase, SCoT also reduces the length of the generated final answer compared to the target model and even brings additional inference efficiency improvements in the Qwen group. For the Llama group, SCoT generates final answers with lengths comparable to those produced by the original target model.

Method	GSM8K	MATH	GaoKao	CollegeMath	Olympiad
D-Qwen-32B	314	632	948	416	2952
SCoT (32B+1.5B)	270	358	610	357	2179
D-Llama-70B	238	256	453	312	840
SCoT (70B+8B)	212	310	568	314	1068

Table 6: Average length of the final answer.

C Sample Presentation

We display two samples of constructed training data for draft selection in Figure 6 and Figure 7. The red text is the prompt template we designed, and the blue text is the input question.

Input: "I will provide several Reasons for the Question. Please choose the best reasoning path and give the serial Number directly (you can only choose one). \n\nQuestion: The rug is 5 feet wider than the chair. The couch is 2 feet longer than twice the width of the rug. If the chair is 3 feet wide. How many feet long is the couch?\n\nHere are the reasoning paths:\n\nReason 1: First, I determine the width of the rug. Since the rug is 5 feet wider than the chair and the chair is 3 feet wide, the rug's width is 3 feet plus 5 feet, which equals 8 feet.\n\nNext, I calculate the length of the couch. The couch is 2 feet longer than twice the rug's width. Twice the rug's width is 2 times 8 feet, which is<n>16 feet</n>. Adding 2 feet to this gives a total length of 18 feet for the couch.\n\nReason 2: First, identify the given information. The chair is 3 feet wide, the rug is 5 feet wider than the chair, and the couch is 2 feet longer than twice the width of the rug.\n\nNext, calculate the width of the rug by adding 5 feet to the width of the chair. Then, determine the length of the couch by taking twice the width of the rug and adding 2 feet to it.\n\nFinally, perform the arithmetic operations to find the exact length of the couch.\n\nReason 3: First, I know that the chair is 3 feetabcdefghijklmnop.\n\nNext, since the rug is 5 feet wider than the chair, I calculate the rug's width by adding 5 feet to the chair's width.\n\nThen, the couch is 2 feet longer than twice the rug's width. I first find twice the rug's width and then add 2 feet to it to determine the couch's length.\n\nFinally, by following these steps, I can find the length of the couch.\n\nReason 4: First, I'll note the given information: the width of the chair is 3 feet. The rug is 5 feet wider than the chair, so the width of the rug is 3 feet plus 5 feet, which equals 8 feet.\n\nNext, the couch is 2 feet longer than twice the width of the rug. Since the rug is 8 feet wide, twice that width is 16 feet. Adding 2 feet to this gives a total length of 18 feet for the couch.\n\nTherefore, the couch is 18 feet long.\n\nReason 5: BPM: Read the problem and identify the given information and what is being asked. The rug is 5 feet wider than the chair. The chair is 3 feet wide. The couch is 2 feet longer than twice the width of the rug. We need to find the length of the couch.\n\nFirst, let's figure out the width of the rug. The rug is 5 feet wider than the chair. Since the chair is 3 feet wide, the rug's width should be 3 + 5 = 8 feet.\n\nNext, the couch is 2 feet longer than twice the width of the rug. So, twice the rug's width is 2 * 8 = 16 feet. Adding 2 feet to that gives 16 + 2 = 18 feet.\n\nTherefore, the couch is 18 feet long.\n\nReason 6: All reasoning paths above are wrong.\n\n< | Assistant | >Number of the best reasoning path: "
Label: [1, 4, 5]

Figure 6: The selection template is in red and question is in blue.

Input: "I will provide several Reasons for the Question. Please choose the best reasoning path and give the serial Number directly (you can only choose one). \n\nQuestion: Steve loves playing video games. His parents get him a console along with 5 games for his birthday. He saves up enough money to buy 1 game per month for a year, and then the following year he starts buying 2 games a month. For the third year he buys 4 games a month as he has a new part-time job that makes him more money. He also gets 5 games for Christmas every year. How many games does Steve have after 3 years?\n\nHere are the reasoning paths:\n\nReason 1: To determine how many games Steve has after 3 years, I'll break down his monthly purchases and annual gifts separately.\n\nIn the first year, Steve buys 1 game per month, totaling 12 games.\n\nIn the second year, he increases his purchase rate to 2 games per month, resulting in 24 games.\n\nIn the third year, his purchase rate rises to 4 games per month, adding 48 games.\n\nAdditionally, Steve receives 5 games at each Christmas, adding 5 games annually.\n\nBy the end of 3 years, Steve has accumulated a total of 108 games.\n\nReason 2: First, I'll determine how many games Steve buys in the first year. He saves enough to buy 1 game each month for 12 months, totaling 12 games.\n\nNext, in the second year, he increases his monthly purchases to 2 games per month. Over 12 months, that amounts to 24 games.\n\nIn the third year, his buying rate increases to 4 games per month. Again, over 12 months, that's 48 games.\n\nAdditionally, Steve receives 5 games as Christmas each year, so that's 5 games per year for 3 years, totaling 15 games.\n\nFinally, I'll add up all the games: 12 (first year) + 24 (second year) + 48 (third year) + 15 (Christmas gifts) to find the total number of games Steve has after 3 years.\n\nReason 3: First, identify the initial number of games Steve has. He starts with 5 games from his birthday.\n\nNext, in the first year, he buys 1 game each month. There are 12 months in a year, so he buys 12 games. Adding this to his initial 5 games gives him 17 games after the first year.\n\nIn the second year, he increases his purchase rate to 2 games per month. Over 12 months, this results in 24 additional games, bringing the total to 41 games.\n\nIn the third year, his purchasing rate doubles again to 4 games per month. This year, he buys 48 games, bringing the total to 89 games after three years.\n\nAdditionally, Steve receives 5 games as Christmas each year. Over three years, this amounts to 15 more games.\n\nFinally, add the Christmas games to the previous total to get the final number of games Steve has after 3 years.\n\nReason 4: First, I need to determine how many games Steve buys each year.\n\nIn the first year, he buys 1 game per month for 12 months, totaling 12 games.\n\nIn the second year, he increases his buying rate to 2 games per month, so that's 24 games.\n\nIn the third year, he increases his rate further to 4 games per month, resulting in 48 games.\n\nAdditionally, Steve receives 5 new games at each Christmas, so that's 5 games annually.\n\nAdding all these together, the total number of games Steve has after 3 years is 12 + 24 + 48 + 5 = 89 games.\n\nReason 5: First, Steve buys 1 game per month for the first year, which is 12 games. \n\nIn the second year, he increases his monthly game purchases to 2 games, totaling 24 games.\n\nIn the third year, his purchasing rate doubles again to 4 games per month, totaling 48 games.\n\nAdditionally, Steve receives 5 games at the end of each year as Christmas gifts, adding 15 games over the three years.\n\nFinally, by the end of the third year, Steve has a total of 84 games.\n\nReason 6: All reasoning paths above are wrong.\n\n< | Assistant | >Number of the best reasoning path: "
Label: [6]

Figure 7: The selection template is in red and question is in blue.