Self-attention-based Graph-of-Thought for Math Problem Solving

Ruiqiao Bai[1], Xue Han[1,*], Shuo Lei[1], Junlan Feng[1,*], Yanyan Luo[1], Chao Deng[1]
[1]JIUTIAN Team, China Mobile Research Institute, Beijing, China
[*]Corresponding Authors
Emails: {bairuiqiao,hanxueai,leishuo,fengjunlan,luoyanyan,dengchao}@chinamobile.com

Abstract

Applying Large Language Models (LLM) to solve math problems is one of the hottest research topics at present. Traditional Chain-of-Thought-based methods typically generate the reasoning path in a chain structure, leading to unnecessary interference caused by non-zero self-attention among weakly related reasoning steps. Such a setting also differs from humans' typical graph-structured reasoning habit (with an inter-step relationship graph in mind). To solve the problem, this paper proposes a novel decoding method for Transformer-based LLM, named Self-attention-based Graph-of-Thought (SaGoT). SaGoT constructs a thought graph simultaneously as an LLM inference (based on a newly defined inter-step self-attention indicator), and generates reasoning steps with a novel graph-structured self-attention mechanism. It is a significant contribution for SaGoT to enable an LLM's graph-like reasoning ability by modifying its inner working operations, compared to SOTA prompting methods that are ex-post, rely on huge LLMs and redundant reasoning step generation to form a graph (inefficient & non-human-like). In addition, SaGoT is a training-free technique that can be seamlessly incorporated into pre-trained Transformer-based LLMs. Our experimental results have shown that SaGoT could significantly enhance mathematical reasoning accuracy without the reliance on huge computationally over-expensive LLMs. It also avoids SOTA methods' performance degradation issues when the LLM is too small to comprehend complex prompts. Moreover, SaGoT integrates intrinsic interpretability into the LLM's reasoning procedure, intuitively assisting humans in understanding how an LLM views the relationships among its reasoning steps, and why the LLM succeeds or fails.

1 Introduction

A Large Language Model (LLM) is a deep learning model capable of handling various natural language tasks (such as text classification and question answering) and has a large number of parameters. At present, LLMs employed in academia and industry are commonly based on the Transformer architecture, whose core technology is the self-attention mechanism (Vaswani et al., 2017). In addition to improving model performance, an advantage of the self-attention mechanism is that it can help people interpret an LLM's inner working procedure by showing how the model distributes weights among tokens (Vig, 2019; Wang et al., 2023b; Vig and Belinkov, 2019).

Applying LLMs to solve math problems is one of the hottest research areas at present, and the Chain-of-Thought (CoT) technology has been widely applied to such a task (see Section 2.1). In a nutshell, CoT requires the LLM to output texts describing the reasoning steps before generating the final answer (Wei et al., 2022).

However, traditional CoT-based methods usually generate reasoning steps in a chain structure (see Figure 1(a)). This might prevent humans from knowing whether the LLM correctly understands and utilizes the relationship among its reasoning steps (as depicted in Figure 1(b), a human-like deduction trajectory is a graph instead of a chain). In addition, some non-zero self-attention values among weakly related reasoning steps might disturb subsequent model outputs, resulting in lower accuracy of the final answer (as verified by our experiments in Section 5).

Recently, scholars have noticed that combining CoT with the graph structure improves LLMs' responses to complex questions (see Section 2.2). However, there is still a lack of methods that efficiently mirror humans' ability (without generating redundant reasoning steps/paths that might be wrong) to directly come up with a new reasoning step based on a graph structure in mind, which depicts the relationships among existing reasoning steps (Zhang et al., 2023). Besides,

Findings of the Association for Computational Linguistics: ACL 2025, pages 6112–6125
July 27 - August 1, 2025

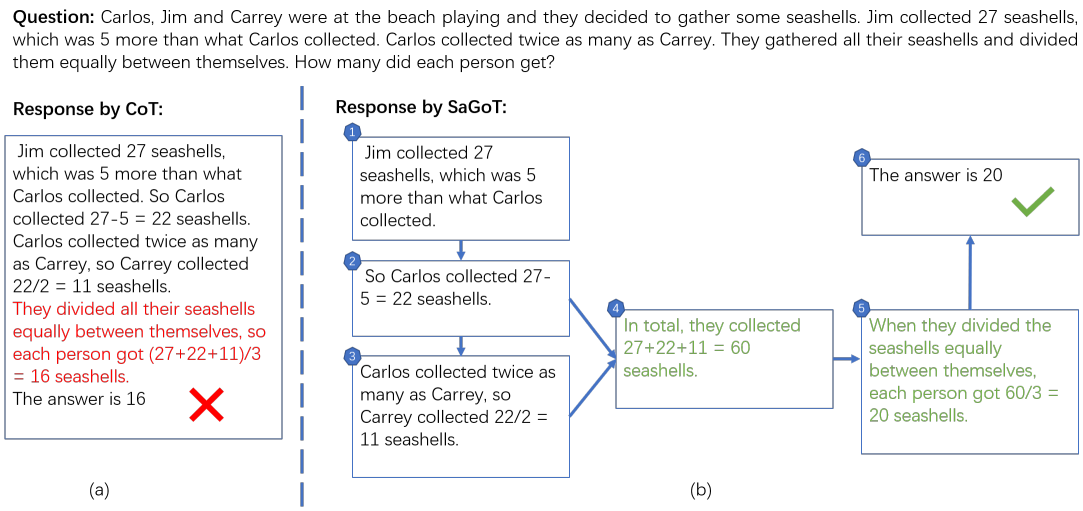

Figure 1: Comparison of CoT and SaGoT.

some techniques require additional model training that's resource-consuming (Cao, 2024; Ning et al., 2024), and there is an ignorance of employing self-attention rectification for graph construction. On the other hand, possessing a graph-based reasoning ability is essential for clearly teasing out the relationships among intermediate reasoning steps, and self-attention is a valuable entry point for achieving such a goal, as it involves the LLM's inner working mechanism and potentially solves the problem from a fundamental perspective of the decoding technique.

To fill the research gap, this paper proposes a novel training-free decoding method for Transformer-based LLMs to solve math problems, namely the Self-attention-based Graph of Thought (SaGoT), which constructs a thought graph simultaneously as an LLM inference based on a newly defined inter-step self-attention indicator, and generates reasoning steps with a novel thought graph-based self-attention mechanism. The thought graph could effectively reflect how LLM views the relationships among its reasoning steps, and thus could better assist humans in interpreting an LLM's inner working mechanisms compared to traditional CoT (see Figure 1). It is a great novelty and significant contribution for SaGoT to enable an LLM to follow a human-like graph-structured reasoning trajectory by modifying the LLM's inner working operations (i.e. integrating intrinsic interpretability), compared to solely prompting-based methods that are ex-post (Cao 2024; Zhang et al. 2024a,b; Yao et al. 2023b), rely on huge LLMs such as GPT-4 (Achiam et al., 2023), and have to generate redundant reasoning paths/steps to form a graph (which is inefficient and non-human-like, see Section 2.2 for details). For instance, the Tree-of-Thought (ToT) method integrates multiple reasoning paths to build a graph, while each path is still generated via a chain-based reasoning structure by the LLM (Yao et al., 2023b). However, a single reasoning path is usually enough for a human being to establish a reasoning graph (Zhang et al., 2023). To solve the problem, SaGoT is designed to significantly differ from existing approaches, and drive an LLM to be more human-like by enabling a single-path-based graph-structured reasoning procedure. Our experimental results have shown that SaGoT could consistently enhance the LLM's accuracy in solving math problems compared to SOTA methods, without additional model training or reliance on huge LLMs that are computationally over-expensive. It avoids prompting-based reasoning methods' performance degradation issue when the LLM is too small to comprehend/achieve the task stated in the complex prompts used for graph construction. Moreover, SaGoT successfully integrates intrinsic interpretability into the LLM's reasoning procedure. The thought graph generated could intuitively assist human interpretation of how an LLM views the relationships among its reasoning steps, and trace why the LLM succeeds or fails in question answering.

In the following paper, a literature review of related work is presented in Section 2, Section 3 illustrates the SaGoT method systematically, and Section 4 introduces our experimental setting. Results and thorough discussions of the experiments

are provided in Section 5. We conclude our analysis and point out the limitations in Sections 6 and 7.

2 Related Work

2.1 Chain-of-Thought

CoT assists the LLM to decompose and solve complex math problems step by step, so as to enhance the correctness of the final answer (Wei et al., 2022). There is a series of LLM-based math problems solving techniques derived from this idea. For example, the Least-to-Most Prompting (Zhou et al., 2023), Plan-and-Solve Prompting (Wang et al., 2023c), and DialCoT-S-PPO methods (Han et al., 2023) all require LLMs to split the math problem into sub-questions. Besides, the BRIDGE method decomposes the math problem solution into substeps, generates the corresponding equations, and then calls the Python package sympy for equation-solving (Wang et al., 2023a).

2.2 Combining Chain-of-Thought with Graph

A few SOTA techniques combine CoT with the graph structure, whereas there are still demerits associated with those works. HoT (Yao et al., 2023a), Graph-Guided Reasoning (Park et al., 2024), and the GoT approach proposed by Yao et al. (2024) treat words or phrases as nodes of a graph. However, these methods usually do not take a full reasoning step as a single node, and thus the graphs they construct could hardly depict inter-step relationships. Some studies (e.g., Besta et al., 2024, Lei et al., 2023 and Xiao and Liu, 2023) provide theoretical frameworks for the LLM's graphical reasoning process. However, when it comes to solving various types of math problems in practice, they haven't offered sufficient detailed instructions supporting the realization of some functional modules described in the general frameworks. Technologies like MindMap (Wen et al., 2024), ToG (Sun et al., 2024), DGoT (Ning et al., 2024), and Rex-GoT (Zheng et al., 2024) are designed specifically for missions such as querying the information stored in a knowledge graph, commonsense multi-choice question answering, or abstract generation. Consequently, obstacles exist when adapting them to math problems solving. There are also prompting methods employing huge LLMs such as GPT-4 (Achiam et al., 2023) to generate multiple reasoning paths and integrate them as a graph (Cao, 2024; Yao et al., 2023b), or generate extra reasoning steps and discard/revise possibly wrong ones based on additional verification (Zhang et al., 2024a,b). Nevertheless, such settings could be inefficient and non-human-like, as a person could directly come up with a relationship graph among reasoning steps in a single trajectory. In addition, as the prompting techniques do not change the inner working mechanism of LLM, within each reasoning path they utilize, independent steps might still interfere with each other due to the original self-attention mechanism (Cao 2024; Zhang et al. 2024a,b; Yao et al. 2023b). The limitations of existing work give us a strong motivation to develop a new technique that enables an LLM to pursue a human-like graph-based reasoning procedure by modifying the LLM's self-attention mechanism.

3 Methodology

Figure 2 presents an overview of the SaGoT method, which consists of 5 procedures: Input Prompt Construction, Initial Node Generation, Subsequent Node Generation, Graph-structured Self-attention Construction, and Process Circulation & Termination. The following sections elaborate on each procedure.

3.1 Input Prompt Construction

First, construct a prompt (denoted by P) with multiple sets of math problems and their step-by-step answer exemplars, as well as the math problem to be solved, and take it as the input to the LLM. The exemplars could be identical to those employed for the CoT technique (labeled by humans), and a bedrock of SaGoT is that the CoT technique with the same prompt could lead to an LLM's accuracy enhancement, as it justifies the effectiveness of the exemplars used. Figure A1 shows an example of the prompt template. Under a greedy search algorithm, the task of generating the ith reasoning step s_i by an LLM given the input prompt and all previously generated reasoning steps $(s_1, ..., s_{i-1})$ could be defined by Equation 1:

$$\max_{s_i} p(s_i|P, s_1, ..., s_{i-1}) \rightarrow s_i \quad (1)$$

where $p(s_i|.)$ is the probability that s_i is output by the LLM.

3.2 Initial Node Generation

Second, conduct a real-time check while the LLM is generating tokens one by one: When the checked

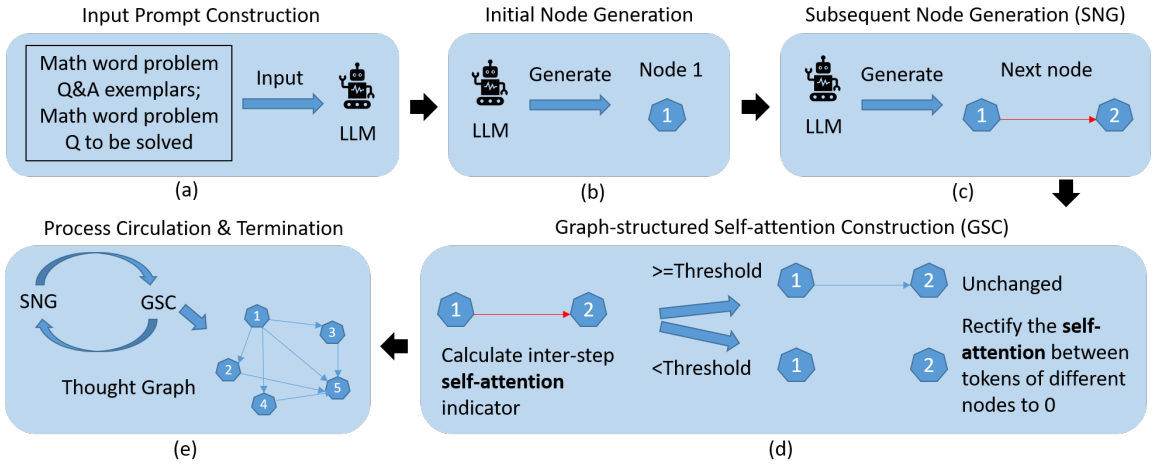

Figure 2: Method overview of SaGoT.

token contains the end-of-sentence symbols (including line breaks, colons, periods, question marks and semicolons), the generation gets paused, and all tokens output so far would be spliced as the first reasoning step (see Figure 2(b)). This first reasoning step s_1 corresponds to the initial node v_1 in the thought graph. The thought graph right after the generation of s_i is represented as $G_i = (V_i, E_i)$, where V_i and E_i are the set of nodes and edges in the graph, respectively. Hence, the Initial Node Generation procedure could be formulated as follows:

$$\max_{s_1} p(s_1|P) \rightarrow s_1 \rightarrow G_1 \tag{2}$$

where $G_1 = (V_1, E_1)$, $V_1 = \{v_1\}$ and $E_1 = \emptyset$.

3.3 Subsequent Node Generation

Then, allow the LLM to restart generating subsequent tokens, and continue checking whether the token contains the end-of-sentence symbols. If yes, pause and splice the generated tokens (except those belonging to existing reasoning steps) as a new reasoning step. This reasoning step s_i corresponds to a new node v_i connected to all previous nodes v_j $(1 \leq j < i)$ by directed lines, resulting in an updated version of the thought graph (denoted by G_i). Each directed line e_{ji} points from a previous node v_j to the new node v_i. Figure 2(c) depicts the procedure, which could be formulated as:

$$\max_{s_i} p(s_i|P, s_1, ..., s_{i-1}, G'_{i-1}) \rightarrow s_i \rightarrow G_i \tag{3}$$

where $G'_{i-1} = (V'_{i-1}, E'_{i-1})$ is equivalent to G_1 when i equals 2, otherwise, it refers to the thought graph after the Graph-structured Self-attention Construction procedure (see Section 3.4) in the previous round of circulation (see Section 3.5); $G_i = (V_i, E_i)$, $V_i = V'_{i-1} \cup \{v_i\}$ and $E_i = E'_{i-1} \cup \{e_{ji}|1 \leq j < i\}$. Please note that $p(s_i|P, s_1, ..., s_{i-1}, G'_{i-1})$ is not equivalent to $p(s_i|P, s_1, ..., s_{i-1})$, as the former one is calculated based on our graph-structured self-attention mechanism demonstrated in Section 3.4.

The lines formed here only infer that, when this procedure is executed, the inter-step self-attention indicator (see Section 3.4) between corresponding reasoning steps is non-zero. Those lines might be removed later as the self-attention values get rectified to zero.

3.4 Graph-structured Self-attention Construction

In this procedure, we first calculate the inter-step self-attention indicator $\alpha(j, i)$ between the reasoning step s_i and each of the previous reasoning steps s_j by the following function:

$$\alpha(j, i) = \max_{J,I,l,h} A^l_h(J, I) \tag{4}$$

where J and I are tokens in s_j and s_i, respectively, and $A^l_h(J, I)$ refers to the self-attention calculated by the hth self-attention head in Transformer layer l, between the Query of token I and the Key of token J (see Appendix A for explanations on the terminology and traditional self-attention mechanism).

When $\alpha(j, i)$ is smaller than a predefined threshold $\theta \in [0, 1]$ (could be determined by grid-searching using a validation set), the self-attention

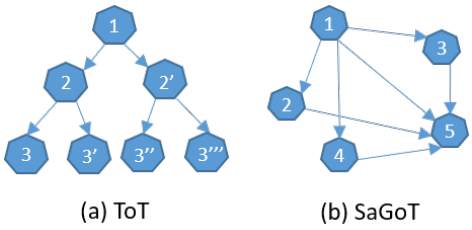

Figure 3: Comparison between ToT and SaGoT (Each node represents a reasoning step, the number in the node represents the sequence of the step in their reasoning path, the arrow in subfigure (a) indicates the step generation path in ToT, the arrow in subfigure (b) shows the inter-step self-attention relationship preserved by SaGoT).

between each token in s_i and each token in s_j (without the self-attention between tokens of the same node) would be rectified as zero (see Figure 2(d)):

$$A_h^l(J,I) = \begin{cases} 0 & \text{if } \alpha(j,i) < \theta \\ A_h^l(J,I) & \text{otherwise} \end{cases} \quad (5)$$

Consequently, directed lines between corresponding nodes e_{ji} in the thought graph G_i would be pruned (as $\alpha(j,i)$ would be corrected to zero simultaneously), resulting in a rectified thought graph G_i' (see Figure 3(b) for an example):

$$G_i \rightarrow G_i' = (V_i', E_i') \quad (6)$$

Denote the set of lines pruned as PE_i, then $V_i' = V_i$ and $E_i' = E_i \setminus PE_i$. In practice, the thought graph construction could be realized by an inter-token self-attention mask, together with a list of tuples recording the start and end token positions of reasoning steps.

When generating each new reasoning step by SaGoT, each token in the step could be sampled following the LLM's original sampling mode (e.g. greedy search), while the token-wise probability distribution used by the sampling procedure is generated based on the graph-structured self-attention mechanism. Specifically, the thought graph could be realized by an inter-token self-attention mask, together with a list of tuples recording the start and end token positions of reasoning steps. When producing each new reasoning step, the step's associated inter-token self-attention values before rectification could already diverge from those calculated following the traditional self-attention system. This is because the graph-structured inter-step self-attention among former steps would influence the Value vectors of tokens in those steps, and further affect the self-attention distribution of a latter step. In other words, under SaGoT's designation, the initial inter-token self-attention intensities before rectification are already calculated based on a graph-structured system, instead of the traditional self-attention mechanism.

3.5 Process Circulation & Termination

Finally, as shown in Figure 2(e), circularly execute the Subsequent Node Generation procedure and the Graph-structured Self-attention Construction procedure, until the LLM generates the end-of-text token (i.e. the token marking the end of an LLM's intended output). Besides, in all procedures stated above, whenever the end-of-text token is generated, the reasoning step and thought graph could be formulated taking the end-of-text token as the end-of-sentence symbol instead, and there is no need to carry out other procedures anymore.

In general, the thought graph put forward in this paper (see Figure 3(b)) could rectify the LLM reasoning procedure simultaneously as an LLM output a single reasoning path. It is highly distinct from the graph structure used in existing methods such as ToT (see Figure 3(a)) that rely on generating extra reasoning paths/steps and additional verification, and better mirrors the human's graph-structured reasoning habit.

4 Experimental Setup

As SaGoT targets open-source small-sized LLMs, the Qwen2-1.5B base model (a decoder-only Transformer-based LLM) has been employed in our experiments (QwenTeam, 2024). The base models of Qwen2-7B (QwenTeam, 2024), Llama3-8B (Grattafiori et al., 2024) and Llama2-13B (Touvron et al., 2023) have also been used to examine the SaGoT's performance when the model scales up. More detailed model hyperparameter settings are illustrated in Appendix B. We utilize the open framework OpenCompass 2.0 (OpenCompass, 2023) for performance evaluation, and datasets employed include GSM8K (Cobbe et al., 2021) and MathBench-A (Liu et al., 2024). GSM8K is a dataset of English grade school math problems and answers created by humans, including over 1.3k test samples. MathBench-A (annotated as MB) comprises 1.5k math questions covering five stages (Arithmetic, Primary, Middle, High, College) and two languages (English and Chinese, denoted as E and C) (Liu

Method	GSM8K	MB-Primary-E	MB-Primary-C	MB-Middle-E	MB-High-E	MB-College-E
Vanilla	19.6	33.3	24.0	9.3	20.0	16.0
CoT	58.3	36.0	36.0	32.0	37.3	26.7
ToT	38.2	37.3	25.3	16.0	18.7	14.7
GraphReason	22.0	33.3	41.3	5.3	8.0	14.7
SGT-Avg	58.4	36.0	36.0	32.0	37.3	21.3
- without exemplars	19.6	32.0-33.3	21.3-24.0	9.3	20.0	12.0
SaGoT	**59.9**	**36.0-37.3**	**40.0**	**36.0**	**41.3**	**26.7-29.3**
- without exemplars	19.7	32.0-34.7	17.3	8.0	21.3	13.3-20.0

Table 1: Accuracy (%) comparison on test datasets (The bold values highlight the results of our method. Value ranges exist since multiple self-attention thresholds might lead to the same validation accuracy, while their results on the test dataset vary).

et al., 2024). As mentioned in Section 3.1, a prerequisite of SaGoT is that the CoT technique with the same prompt could lead to accuracy enhancement on the target task. Due to the incompatibility between Qwen2-1.5B and the exemplars provided by OpenCompass 2.0, CoT would lead to performance degradation on partial datasets (see Appendix C for corresponding results and explanations). Hence, in Section 5, we focus on datasets whose corresponding CoT exemplars function normally. To determine the inter-step self-attention threshold by grid-searching using a validation set, we randomly split each of the original datasets into a new validation dataset and a new test dataset of approximately the same size (referred to as the validation and test datasets in the following paper).

5 Result and Discussion

5.1 Comparison with SOTA Methods

We compare the performance of SaGoT with Vanilla, CoT (Wei et al., 2022) and SOTA graph-related reasoning methods ToT (Yao et al., 2023b), GraphReason (Cao, 2024) and SGT-Avg. Vanilla refers to the naive approach without adding exemplars to the input prompt template. ToT and GraphReason are graph-based reasoning methods that construct the graph by sampling the reasoning step/path multiple times (Yao et al., 2023b; Cao, 2024). SGT-Avg is developed by us for testing the effect of changing the maximum value-based inter-step self-attention indicator used by SaGoT to an average value-based indicator. For each dataset, our experiments on CoT, ToT, GraphReason, SGT-Avg and SaGoT use the same exemplars guiding the model to generate step-wise answers, except for slight format differences due to the algorithms' default settings.

Table 1 presents the comparison results on test datasets. It should be noted that GSM8K and MB-Primary allow open-ended responses, while queries of MB-Middle/High/College are single-choice questions, so the task difficulty might not solely depend on the grade level. The results manifest that SaGoT could further improve the correctness compared to SOTA methods. Specifically, ToT requires the LLM to be capable of accurately evaluating the correctness of each reasoning step for its tree-branch pruning procedure (Yao et al., 2023b). This might lead to reliance on huge powerful LLMs such as GPT-4 (Achiam et al., 2023) to ensure its performance enhancement effect. On the other hand, SaGoT could release this constraint, leading to a higher accuracy even with a lightweight LLM, without the need for high computational burdens. Although GraphReason and SaGoT achieve comparable results on MB-Primary-C, GraphReason requires the LLM to generate 30 reasoning paths for each question, which could be inefficient (Cao, 2024). SaGoT also outperforms SGT-Avg. A possible reason is that compared to the maximum value-based indicator (employed by SaGoT), there are higher chances for uninformative tokens in long reasoning steps to drag down and disturb the average inter-token self-attention value (employed by SGT-Avg) from accurately reflecting inter-step relationships.

Table 2 summarizes the disadvantages of more SOTA graph-based methods compared to SaGoT, including those not considered in the quantitative comparison above (as their codes might not be open-source, or not readily applicable to the tasks of this study). In general, a significant novelty and contribution of SaGoT, is that it enables an LLM to achieve a human-like single-path-based graph-of-thought by rectifying the LLM's inner working mechanism, compared to prompting methods that are inefficient and fail to modify an LLM fundamentally (Cao 2024; Zhang et al. 2024a,b; Yao et al.

Method	Disadvantages compared to SaGoT
Yao et al., 2023a; Park et al., 2024; Yao et al., 2024	Could not take each reasoning step as a graph node for inter-step relationship characterization.
Besta et al., 2024; Lei et al., 2023; Xiao and Liu, 2023; Wen et al., 2024; Sun et al., 2024; Ning et al., 2024; Zheng et al., 2024	Insufficient to support the specific task of math problems solving, for reasons such as they haven't offered sufficient detailed instructions supporting the realization of some functional modules described in the general frameworks, or the technologies are tailor-made for other tasks.
Cao 2024; Zhang et al. 2024a,b; Yao et al. 2023b	Inefficient graph construction based on extra reasoning trajectories/steps generation and additional verifiers. Within each reasoning trajectory, unrelated steps might disturb each other through self-attention.

Table 2: Disadvantages of SOTA graph-based methods compared to SaGoT.

2023b). Besides, the self-attention-based thought graph (see Figure 3(b)) could efficiently rectify the LLM reasoning procedure simultaneously as an LLM outputs a single reasoning path, and assist better human interpretation of the LLM's inner reasoning rationales.

5.2 Ablation Study on CoT Exemplars

An ablation study has been conducted to examine the effect of removing the CoT exemplars from the prompt of SaGoT and SGT-Avg. As shown in Table 1 (the self-attention thresholds used by each pair of methods with and without exemplars are consistent), such a prompt with exemplars could also play a crucial role in elevating the performance. This makes sense as the CoT exemplars guide the LLM's step-wise answers, which is the bedrock for calculating the inter-step self-attention indicator. It should be noted that the CoT exemplars used in our experiments are not tailor-made for our method, but the ones uniformly employed by OpenCompass 2.0. Hence, the requirement of SaGoT on the CoT exemplar isn't strict (as long as they enable CoT to outperform Vanilla).

5.3 Influence of the Self-attention Threshold

Table 3 presents SaGoT's performance when the self-attention threshold varies from 0 to 1, based on the overall dataset (for a higher statistical significance). SaGoT with the threshold 0 is equivalent to CoT, as there would be no self-attention rectified. In other words, CoT could be viewed as a special case of SaGoT. Underlined values highlight the thresholds when SaGoT outperforms CoT. The results reveal that low and high thresholds are sub-optimal, and the thresholds leading to the best performance typically range from 0.3 to 0.8. In addition, the higher smoothness of the accuracy changing pattern on GSM8K as the threshold varies is consistent with our expectations. Specifically,

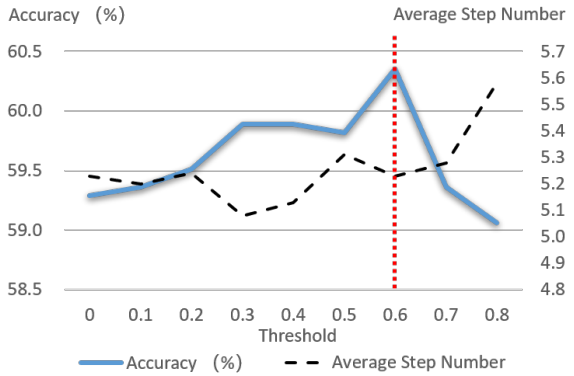

Figure 4: SaGoT with different thresholds on GSM8K.

the number of samples in the original test dataset in GSM8K is 1319, while each of the MathBench-A sub-datasets shown in Table 3 contains 150 samples. Hence, our experimental results indicate that a higher number of samples would contribute to a higher statistical stability, and potentially contribute to a better hyperparameter selection result. On the other hand, the validation dataset employed by SaGoT is still much smaller than the training dataset typically employed in training-based methods, and thus saves researchers' data collection efforts. More analysis on the validation sample size is presented in Appendix D.

To better visualize a typical case of the accuracy-changing patterns, partial results on GSM8K, whose number of questions is the highest among datasets (thus might be less influenced by randomness), are presented in Figure 4 and the vertical line marks the peak. The average reasoning step numbers corresponding to different thresholds are also calculated to support an in-depth analysis. As the threshold increases from 0, more redundant inter-step self-attention linkages get pruned by SaGoT, and thus, there could be a correctness improvement. When the threshold is too large, almost all inter-step self-attention linkages get pruned, and

Threshold	0 (CoT)	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
GSM8K	59.3	59.4	59.5	59.9	59.9	59.8	**60.3**	59.4	59.1	56.1	35.8
MB-Primary-E	40.7	40.7	40.7	40.7	40.7	40.7	**41.3**	40.7	**41.3**	38.0	30.7
MB-Primary-C	35.3	35.3	35.3	36.0	36.0	36.0	35.3	36.7	**38.7**	38.0	28.0
MB-Middle-E	30.0	30.0	27.3	28.0	32.7	30.7	33.3	**34.7**	29.3	22.0	14.0
MB-High-E	34.0	36.7	36.0	32.7	33.3	**39.3**	34.0	36.7	26.7	23.3	20.7
MB-College-E	26.0	26.0	26.0	**27.3**	22.7	21.3	26.7	26.0	20.0	25.3	24.7

Table 3: Accuracy (%) comparison of SaGoT with different thresholds on overall datasets (The bold values signify the highest accuracy. Underlined values highlight the thresholds when SaGoT outperforms CoT).

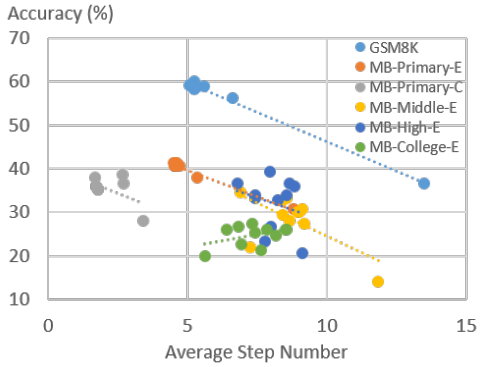

Figure 5: Accuracy and average step number of SaGoT with different thresholds across datasets.

the ignorance of some informative inter-step relationships might result in a performance decrement. Besides, we are surprised to find that the accuracy-changing patterns could be opposite to the average reasoning step number. The negative correlation can also be observed in most of the other datasets, and the only positive correlation observed on the MB-Colledge-E dataset is weak (see Figure 5). This indicates that, unlike some SOTA techniques employed by reasoning-focused models that are blamed for requiring LLMs to overthink and generate an overlong reasoning path (Chen et al., 2024), SaGoT might not have to rely on largely extending the reasoning trajectory to enhance the final accuracy. This could be a merit of SaGoT, as producing a tremendous amount of steps would be resource-consuming, and a concise answer might be preferred by humans. A more in-depth comparison regarding the computational complexity of SOTA prompting techniques and SaGoT is presented in Appendix E, which supports SaGoT's efficiency.

5.4 Effect of Model Scaling

Figure 6 presents the highest accuracy enhancement achievable on the overall datasets by adjusting the self-attention threshold under different model sizes (Qwen2-1.5B, Qwen2-7B, Llama3-8B, Llama2-13B). The results indicate that SaGoT could still manage to enhance the LLM performance when the model scales up, which further strengthens its effectiveness. In addition, MB-Middle/High/College contains single-choice questions, while the other datasets consist of open-answer questions. The experimental results suggest that, compared to open-answer questions, it might be easier for SaGoT to achieve a higher maximum accuracy enhancement on single-choice question tasks. Besides, the result achieved by Qwen2-7B is comparable to SOTA methods such as GraphReason (Cao, 2024) with GPT-3.5 (which achieves 85.7% on GSM8K when asking the LLM to generate the reasoning path 30 times for a single question).

5.5 Thought Graph for Better Interpretation

In Figure 1, we showcase how SaGoT could help enhance the LLM's reasoning accuracy while contributing to better human interpretations of LLMs' inner reasoning mechanisms. The left sub-figure presents the results of traditional CoT, while the right one visualizes the thought graph produced by SaGoT (threshold = 0.6). As the final answers could be extracted from the last sentence/step shown in the figure, the rest of the texts are omitted. Apparently, the thought graph of SaGoT could intuitively depict how the LLM understands the relationships among its reasoning steps, while the CoT output fails. Additionally, we could observe an interesting phenomenon that, for the specific case shown in Figure 1, after removing redundant self-attention linkages among former reasoning steps by SaGoT, the LLM seems to have a 'clearer mind' to separate the latter reasoning procedure into sub-steps (i.e. the green texts), instead of mixing multiple reasoning steps into a single one (i.e. the red texts) that leads to calculation errors. We believe that, with the assistance of SaGoT, there might be other insightful findings on the reasoning mecha-

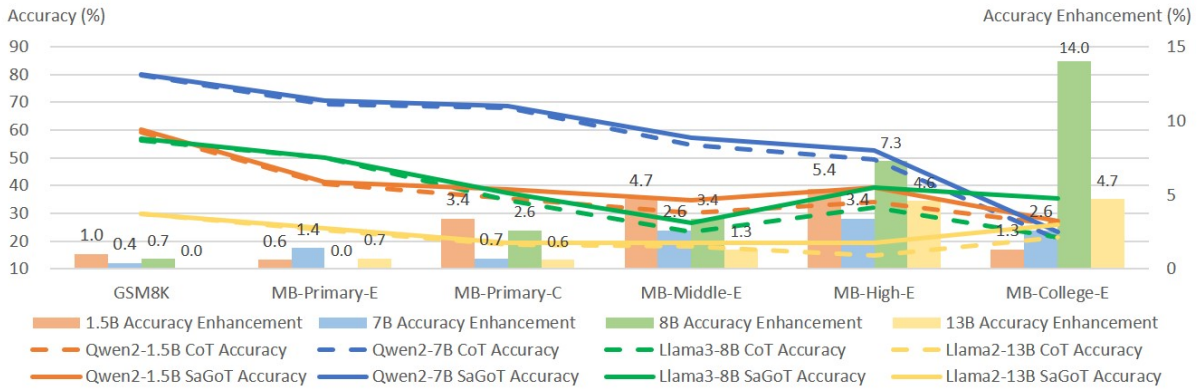

Figure 6: Model scaling effect on overall datasets.

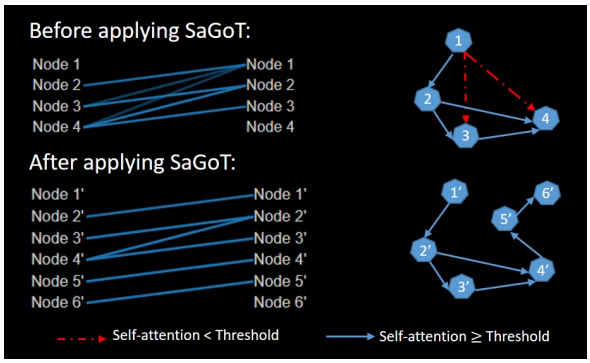

Figure 7: Impacts on inter-step self-attention.

nisms of LLMs.

Figure 7 visualizes the possible impacts of SaGoT on the inter-step self-attention distribution. Left sub-figures show inter-step self-attention distributions and right sub-figures present corresponding thought graphs. Within each left sub-figure, left nodes correspond to Queries and right nodes correspond to Keys, and a line with a deeper color indicates a greater self-attention intensity. It should be noted that, rectification of self-attention among former reasoning steps by SaGoT could result in changes in the specific content of the following reasoning steps, the subsequent self-attention strength distribution, and also the total reasoning step number.

We also present a bad case analysis in Appendix F for the condition when SaGoT fails in the question answering. Typically, the reason is that when the LLM is not capable of generating relatively feasible reasoning steps with the CoT technique, further rectifying self-attention by SaGoT among reasoning steps with overwhelmingly erroneous steps might not help a lot. However, with the thought graph generated by SaGoT, users could still more clearly understand how an error in the early reasoning step influences subsequent steps, and trace why an LLM fails from a more in-depth perspective.

6 Conclusion

This paper proposes a novel decoding method named SaGoT enabling LLMs to perform human-like graph-structured mathematical reasoning. Experimental results show that SaGoT could effectively enhance an LLM's mathematical reasoning accuracy while exempting the reliance on huge LLMs that are resource-consuming. It is a training-free technique with a thought graph generated that could assist humans in interpreting how an LLM views the relationships among its reasoning steps.

In the future, a potential research direction is to adapt SaGoT to LLM with the Mixture-of-Experts structure by enabling different Experts to generate different thought graphs, which might focus on different aspects of the inter-step relationships. It could be meaningful for SaGoT to assist humans in interpreting how the Experts diverge in their reasoning approaches. Moreover, although SaGoT is designed to be a training-free technique that enables the LLM to have a graph-structured reasoning ability by its inherent working mechanism, it could also shed light on the LLM training technique by integrating the graph-structured self-attention mechanism into the model training procedure. Specifically, graph-structured training mechanisms might make use of human-annotated inter-reasoning relationships to guide the inter-step self-attention formulation. We would like to encourage further investigations into that.

7 Limitation

SaGoT is designed for math problem solving, so it might require modifications when applied to other reasoning tasks such as code generation. However, it is also a merit of SaGoT that it could potentially be generalized to other reasoning tasks. Take the Python code generation task as an example, as the codes are typically wrapped as function blocks, future work might develop a multi-level SaGoT technique, which could include a low-level graph depicting relationships among code lines within a function, and a high-level graph describing interactions among functions. This might be an interesting research direction in the future. Besides, the existing implementation of the KV cache is incompatible with SaGoT, and an engineering adjustment is encouraged. In addition, in our experiments, we define the symbol '.' as the English period, whereas it sometimes represents the decimal point or the abbreviation. This might be solved by refining the period recognition rules in the future.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

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

Lang Cao. 2024. Graphreason: Enhancing reasoning capabilities of large language models through a graph-based verification approach. *arXiv preprint arXiv:2308.09267*.

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

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

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Chengcheng Han, Xiaowei Du, Che Zhang, Yixin Lian, Xiang Li, Ming Gao, and Baoyuan Wang. 2023. Dialcot meets ppo: Decomposing and exploring reasoning paths in smaller language models. *arXiv preprint arXiv:2310.05074*.

Bin Lei, Chunhua Liao, Caiwen Ding, et al. 2023. Boosting logical reasoning in large language models through a new framework: The graph of thought. *arXiv preprint arXiv:2308.08614*.

Hongwei Liu, Zilong Zheng, Yuxuan Qiao, Haodong Duan, Zhiwei Fei, Fengzhe Zhou, Wenwei Zhang, Songyang Zhang, Dahua Lin, and Kai Chen. 2024. Mathbench: Evaluating the theory and application proficiency of llms with a hierarchical mathematics benchmark. *arXiv preprint arXiv:2405.12209*.

Xinyu Ning, Yutong Zhao, Yitong Liu, and Hongwen Yang. 2024. Dgot: Dynamic graph of thoughts for scientific abstract generation. *arXiv preprint arXiv:2403.17491*.

OpenCompass. 2023. Opencompass: A universal evaluation platform for foundation models. https://github.com/open-compass/opencompass. Accessed: 2024-07-25.

Jinyoung Park, Ameen Patel, Omar Zia Khan, Hyunwoo J Kim, and Joo-Kyung Kim. 2024. Graph elicitation for guiding multi-step reasoning in large language models. *arXiv preprint arXiv:2311.09762*.

Pytorch. 2023. torch.nn.functional.scaled_dot_product_attention. https://pytorch.org/docs/stable/generated/torch.nn.functional.scaled_dot_product_attention.html. Accessed: 2024-07-25.

QwenTeam. 2024. Qwen2 technical report. Technical report, Alibaba Group.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Heung-Yeung Shum, and Jian Guo. 2024. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. *arXiv preprint arXiv:2307.07697*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Jesse Vig. 2019. A multiscale visualization of attention in the transformer model. *arXiv preprint arXiv:1906.05714*.

Jesse Vig and Yonatan Belinkov. 2019. Analyzing the structure of attention in a transformer language model. *arXiv preprint arXiv:1906.04284*.

Dingzirui Wang, Longxu Dou, Wenbin Zhang, Junyu Zeng, and Wanxiang Che. 2023a. Exploring equation as a better intermediate meaning representation for numerical reasoning. *arXiv preprint arXiv:2308.10585*.

Lean Wang, Lei Li, Damai Dai, Deli Chen, Hao Zhou, Fandong Meng, Jie Zhou, and Xu Sun. 2023b. Label words are anchors: An information flow perspective for understanding in-context learning. *arXiv preprint arXiv:2305.14160*.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023c. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. *arXiv preprint arXiv:2305.04091*.

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

Yilin Wen, Zifeng Wang, and Jimeng Sun. 2024. Mindmap: Knowledge graph prompting sparks graph of thoughts in large language models. *arXiv preprint arXiv:2308.09729*.

Changnan Xiao and Bing Liu. 2023. Conditions for length generalization in learning reasoning skills. *arXiv preprint arXiv:2311.16173*.

Fanglong Yao, Changyuan Tian, Jintao Liu, Zequn Zhang, Qing Liu, Li Jin, Shuchao Li, Xiaoyu Li, and Xian Sun. 2023a. Thinking like an expert: Multimodal hypergraph-of-thought (hot) reasoning to boost foundation modals. *arXiv preprint arXiv:2308.06207*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023b. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

Yao Yao, Zuchao Li, and Hai Zhao. 2024. Beyond chain-of-thought, effective graph-of-thought reasoning in language models. *arXiv preprint arXiv:2305.16582*.

Beichen Zhang, Kun Zhou, Xilin Wei, Xin Zhao, Jing Sha, Shijin Wang, and Ji-Rong Wen. 2023. Evaluating and improving tool-augmented computation-intensive math reasoning. *Advances in Neural Information Processing Systems*, 36.

Yifan Zhang, Jingqin Yang, Yang Yuan, and Andrew Chi-Chih Yao. 2024a. Cumulative reasoning with large language models. *arXiv preprint arXiv:2308.04371*.

Yifan Zhang, Yang Yuan, and Andrew Chi-Chih Yao. 2024b. On the diagram of thought. *arXiv preprint arXiv:2409.10038*.

Li Zheng, Hao Fei, Fei Li, Bobo Li, Lizi Liao, Donghong Ji, and Chong Teng. 2024. Reverse multi-choice dialogue commonsense inference with graph-of-thought. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19688–19696.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. 2023. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.

A Appendix A

In a typical Transformer layer, each input token is mapped to a Query vector, a Key vector and a Value vector (Vaswani et al., 2017). The output corresponding to each input token would be calculated as the weighted sum of Value vectors in the layer. The weight of each Value is called the self-attention intensity, calculated based on the Query and the Key. Specifically, in a decoder-only LLM, the self-attention intensity A, which is the weight of token y's Value when calculating the output corresponding to input token x, can be calculated using the following formula (Vaswani et al., 2017):

$$A = \begin{cases} 0, & x\ is\ before\ y \\ softmax(\frac{QK^T}{\sqrt{d}}), & otherwise \end{cases} \quad (7)$$

where $softmax(.)$ represents the $Softmax$ function, Q represents the Query corresponding to x, K represents the Key corresponding to y, and d represents the length of the Query or Key (both are of equal length). Figure A2 shows an example of the self-attention mechanism of a decoder-only LLM.

Besides, many LLMs employ the multi-head self-attention mechanism, enabling the existence of multiple self-attention heads in a single Transformer layer (Vaswani et al., 2017). Each self-attention head can result in a distinct set of Query, Key and Value vectors.

B Appendix B

Compared to the default model hyperparameters, max_new_tokens is reduced to 1024 to prevent unnecessary token generation, do_sample and use_cache are set as $False$ to avoid the influence

Question: math word problem exemplar 1
Let's think step by step.
Answer:
step-by-step answer exemplar 1

multiple math word problem question and step-by-step answer exemplars, following the format above

Question: math word problem to be answered
Let's think step by step.
Answer:

Figure A1: An example of the prompt template (OpenCompass, 2023). (Black words are fixed in the template, and blue words would be substituted by corresponding exemplars or the question to be answered by the LLM.)

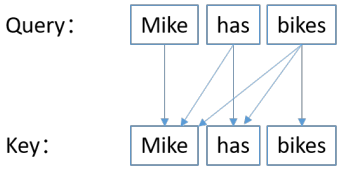

Figure A2: Self-attention mechanism in a decoder-only LLM. (Each square in the figure represents a token; the upper squares stand for tokens from which the Query is derived, and the lower squares correspond to the Key. Only non-zero self-attention intensity would be represented by the directed line in this figure, whose arrow points from the token that produces the Query to the token that produces the Key.)

of randomness and incompatibility with SaGoT's design, respectively. Besides, when calculating the inter-token self-attention used for generating the customized self-attention mask required by SaGoT, we find that in Qwen2-1.5B's first Transformer layer and Qwen2-7B's last Transformer layer, directly multiplying the Query and Key vectors sometimes results in an output exceeding the maximum limit. Qwen2-1.5/7B employs the $scaled\_dot\_product\_attention$ function to prevent the problem, which has three different implementations to be automatically selected following complex rules based on inputs, and is subjected to change (Pytorch, 2023). To avoid such complexity, the self-attention in the corresponding Transformer layer is excluded from the inter-step self-attention indicator calculation for those specific models.

C Appendix C

As mentioned in the paper, a pre-requisite of SaGoT's application, is that the CoT technique with the same prompt could enhance the model performance compared to Vanilla on the targeting task, and this section presents and analyzes conditions when such a pre-requisite is not fulfilled.

On the MB-Arithmetic-E, MB-Middle-C, MB-High-C and MB-College-C sub-datasets under MathBench-A, the Vanilla method performs even better than CoT on Qwen2-1.5B (see Table C1). Upon a closer look, it is found that OpenCompass 2.0 employs the same exemplars for MB-Arithmetic-E and MB-Primary-E, whose format is similar to questions in MB-Primary-E, but quite different from MB-Arithmetic-E. The difference might misdirect an LLM's stepwise answering when it's applied to the task of MB-Arithmetic-E. MB-Middle-C, MB-High-C and MB-College-C share the same set of exemplars (distinct from other tasks) provided by OpenCompass 2.0, and we suspect the anomaly might be caused by the inherent Chinese capability of Qwen2-1.5B, which might not fully comprehend those exemplars.

We can observe that even when CoT malfunctions, SaGoT might still eliminate the accuracy reduction effect. This could be an advancement of SaGoT. Given the high reliance on CoT across the academic and industrial fields, there are occasions when improper CoT exemplars lead to LLM performance degradation unexpectedly, while our experimental results show that the integration of SaGoT might alleviate this negative effect.

Method	MB-Arithmetic-E	MB-Middle-C	M-High-C	M-College-C
Vanilla	44.7	49.3	32.0	29.3
CoT	41.3	29.3	25.3	12.0
ToT	22.0	20.0	13.3	12.0
GraphReason	34.0	4.0	13.3	12.0
SGT-Avg	42.7	29.3	28.0	13.3
- without exemplars	42.0	49.3	25.3	28.0
SaGoT	**42.0**	**29.3**	**26.7**	**18.7**
- without exemplars	42.0-42.7	48.0-49.3	29.3	29.3

Table C1: Accuracy (%) comparison on test datasets when CoT performs worse than Vanilla (The bold values highlight the results of our method. Value ranges exist since multiple self-attention thresholds might lead to the same validation accuracy, while their results on the test dataset vary).

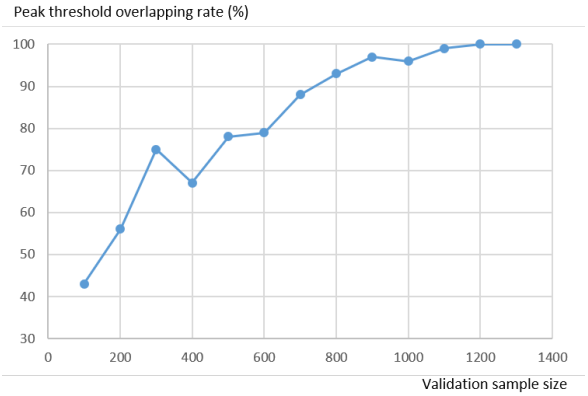

Figure D1: Impacts of validation sample size on the peak threshold overlap rate for the GSM8K dataset.

D Appendix D

Figure D1 visualizes the impact of validation sample size on the rate of obtaining the best threshold used by SaGoT on the GSM8K dataset. Specifically, for each sample size ranging from 100 to 1300, the validation dataset is randomly sampled 100 times. Then we calculate the rate that the threshold selected based on the validation set is the same as the actual threshold corresponding to the peak value (which is 0.6 for the GSM8K dataset). As shown in the figure, the peak threshold overlapping rate could reach approximately 80% when the sample size is around 500.

E Appendix E

To take into consideration various prompting methods that do not modify the LLM's inherent self-attention mechanism, we would like to examine a general condition: Without loss of generality, assume that a prompting method X and SaGoT require the LLM to generate N and M reasoning steps separately to finish the reasoning procedure. The average number of tokens in each step is n and m for method X and SaGoT, separately. As there would be no inter-token self-attention computation pruned by method X, the computational complexity caused by the self-attention module would be $O((N*n)^2)$. Please note that method X might require other models to validate the step correctness, so the actual computational complexity might be higher than this value. As for SaGoT, assume that there are K inter-step self-attention linkages that get pruned in the thought graph. Hence, there would be $K*m^2$ inter-token self-attention pruned, resulting in a final computational complexity $O((M*m)^2-K*m^2)$. In many cases, state-of-the-art prompting-based reasoning methods would rely on largely enhancing the number of tokens generated, while SaGoT does not, resulting in $N*n > M*m$, so $O((N*n)^2) > O((M*m)^2-K*m^2)$, the computational complexity of SaGoT would be lower.

F Appendix F

Figure F1 presents an example when SaGoT could not lead to a correct answer, and Figure F2 visualizes the inter-step self-attention distribution before and after applying SaGoT on that case. It could be seen that there are overwhelming errors in the reasoning steps generated by CoT (see the red text on the left sub-figure), and in that case, further rectifying the relationships among them by SaGoT might hardly lead to a correct answer. However, SaGoT could still help researchers to more clearly observe how the error in the initial steps is transmitted across reasoning steps, and influence the final answer.

For a normal LLM, the noises in self-attention relationships among early reasoning steps might lead to errors when generating the subsequent reasoning steps, and this is exactly why we apply the SaGoT algorithm to alleviate this issue. If there are still

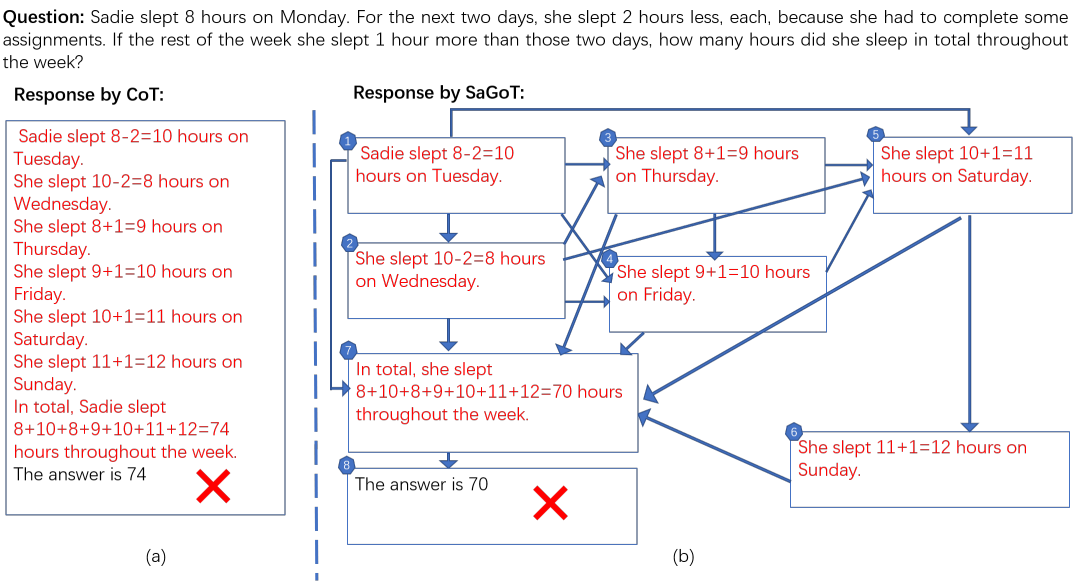

Figure F1: Comparison of CoT and SaGoT in a bad case.

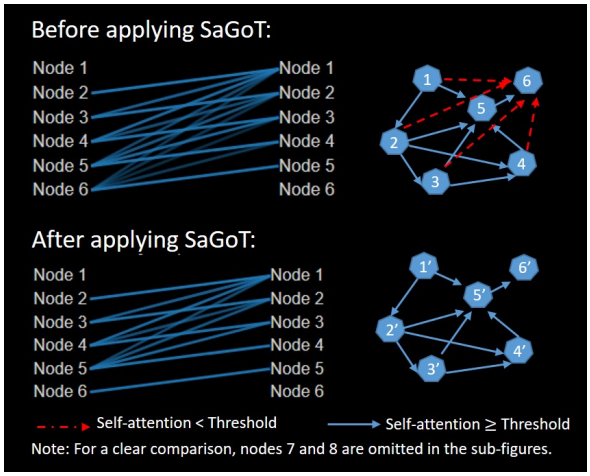

Figure F2: Impacts on inter-step self-attention on a bad case.

errors in the relationship among early steps after the application of SaGoT, an approach to further mitigate this, is to adjust the CoT exemplars to ensure that the CoT method with the same exemplars is capable of better enhancing the LLM reasoning accuracy. The rationale behind this approach, is that better CoT exemplars could lead to fewer errors made in the LLM's initial reasoning steps, which is the bedrock for a correct recognition of the inter-step relationship among earlier reasoning steps.