Decoding on Graphs: Faithful and Sound Reasoning on Knowledge Graphs through Generation of Well-Formed Chains

Kun Li[♡*], **Tianhua Zhang**[♡*], **Xixin Wu**[♡],
Hongyin Luo[◇], **James Glass**[◇], **Helen Meng**[♡]
[♡]The Chinese University of Hong Kong, Hong Kong SAR, China
[◇]Massachusetts Institute of Technology, Cambridge MA, USA
kunli@se.cuhk.edu.hk, thzhang@link.cuhk.edu.hk

Abstract

Knowledge Graphs (KGs) can serve as reliable knowledge sources for question answering (QA) due to their structured representation of knowledge. Existing research on the utilization of KG for large language models (LLMs) prevalently relies on subgraph retriever or iterative prompting, overlooking the potential synergy of LLMs' step-wise reasoning capabilities and KGs' structural nature. In this paper, we present DoG (**D**ecoding **o**n **G**raphs), a novel framework that facilitates a deep synergy between LLMs and KGs. We first define a concept, *well-formed chain*, which consists of a sequence of interrelated fact triplets on the KGs, starting from question entities and leading to answers. We argue that this concept can serve as a principle for making faithful and sound reasoning for KGQA. To enable LLMs to generate well-formed chains, we propose *graph-aware constrained decoding*, in which a constraint derived from the topology of the KG regulates the decoding process of the LLMs. This constrained decoding method ensures the generation of well-formed chains while making full use of the step-wise reasoning capabilities of LLMs. Based on the above, DoG, a training-free approach, is able to provide faithful and sound reasoning trajectories grounded on the KGs. Experiments across various KGQA tasks with different background KGs demonstrate that DoG achieves superior and robust performance. DoG also shows general applicability with various open-source LLMs[1].

1 Introduction

Large language models (LLMs) have shown impressive performance across various natural language processing tasks (Brown et al., 2020; Ouyang et al., 2022; Achiam et al., 2023; Dubey et al., 2024). Despite this, LLMs still suffer from the lack of knowledge and are prone to hallucinations for some knowledge-intensive scenarios (Yin et al., 2023; Hong et al., 2023). One direction to addressing this limitation is to supplement LLMs with external knowledge (Lewis et al., 2020; Li et al., 2022; Asai et al., 2024). Among the external knowledge sources, knowledge graphs (KGs) lend themselves to knowledge-intensive tasks like question-answering, due to their structured and explicit representations of knowledge (Sun et al., 2024a; Jiang et al., 2023; Baek et al., 2023).

Many researchers have studied the augmentation of KGs as knowledge sources for LLMs when handling question answering (QA) tasks (Sun et al., 2024a; Luo et al., 2024; Markowitz et al., 2024; He et al., 2024; Wang et al., 2023). These KGQA approaches can be mainly categorized into the following two threads. (1) Luo et al. (2024), He et al. (2024) and Mavromatis and Karypis (2024) rely on specialized subgraph retrievers to retrieve question-focused subgraphs from a complete KG, which are then fed into LLMs for predicting answers. Such approaches shift the burden of finding answers on KGs to the subgraph retrievers, while the LLMs merely select the most possible answer from the highly concentrated subgraphs and are not deeply involved in the graph reasoning process. This only makes limited use of LLMs' reasoning capabilities. Moreover, training a specialized subgraph retriever requires substantial labeled data, and the trained retriever may struggle with out-of-domain scenarios (a retriever trained on one KG typically fails to adapt effectively to other KGs with disparate taxonomies, *e.g.*, Freebase (Bollacker et al., 2008) *vs.* Wikidata (Vrandeić and Krötzsch, 2014), §4). (2) To address these shortcomings, an alternative thread of works directly engages LLMs in the reasoning process on KGs (Sun et al., 2024a; Markowitz et al., 2024; Sun et al., 2024b), through iteratively prompting LLMs to generate a series of operations on KGs (*e.g.*, neighbor entities explo-

[*] Equal contribution.
[1]The code is available here.

ration, path decision, and *etc.*), finally reaching to an answer. Such approaches, however, place significant demands on the LLMs' abilities, since the instructions intended for some complex operations are often difficult for small-sized LLMs (with < 10 billion parameters) to understand and execute, as evidenced by the fact that the backbone LLMs used in these works all have tremendous sizes, *e.g.*, GPT-4, Llama 2-70b.

Recently, research on chain-of-thought (CoT, Wei et al., 2022, Feng et al., 2023) has established that LLMs can solve some complex questions through the generation of step-by-step reasoning trajectories. Meanwhile, the structured nature of KGs, where related facts are connected (directly or indirectly) through paths comprising sequences of relations and entities between them, facilitates sound reasoning trajectories on the graphs. In that sense, a tight coupling between LLMs' step-wise reasoning capability and KGs can be a promising solution to KGQA. Motivated by this, we explore a perspective different from the aforementioned two threads of works: *let LLMs reason directly on KGs through generation of reasoning steps*. Specifically, given a KG as input, the LLM is required to reason on KGs by generating sequential and interrelated reasoning steps, with each step being anchored by a fact triplet on the KGs. This manner can make full use of LLMs' textual understanding and generation capabilities for discovering suitable paths from question to answer, leading to general applicability to different KGs. Moreover, predicting the next token during generation is easier than following complex instructions for small LLMs.

Specifically, we first define a concept, ***well-formed chain***, to serve as a principle for making faithful and sound reasoning on KGs. A well-formed chain, starting from query entities and leading to an answer, is composed of a sequence of interconnected triplets as the reasoning steps, and each step of triplet can only grow from all previously visited triplets. Thanks to this properties, for a question, a well-formed chain offers a reasoning trajectory that is sound and faithful to the KG; Moreover, it will also naturally narrow down the search scopes for inferring each reasoning step. Therefore, in principle, by generating well-formed chains, LLMs will derive high-quality reasoning trajectories from query entities to answer candidates. However, under a training-free setting, it is almost impossible to ensure the generation of such chains simply by some conventional techniques like in-context learning or prompting (Wei et al., 2021). This mainly stems from the obscure structural information after the linearization of graphs in input, and the difficulties of instilling the notion of well-formed chain into LLMs through prompting alone. To overcome this, we further propose ***graph-aware constrained decoding***, to impose a constraint upon LLMs' decoding process. The constraint is induced from a local and query-centric subgraph, and as the reasoning process proceeds, the subgraph progressively expands in accordance with both the principle of well-formed chains and the topology of the source KG. By restricting the space of output tokens, this hard constraint is able to strictly regularize the LLM's generation to be a well-formed chain. Combining well-formed chains and graph-aware constrained decoding together, our approach, Decoding on Graphs (DoG), enables LLMs to produce reasoning trajectories that are sound and faithful to the given KGs.

We evaluate our approach on three KGQA datasets with various open-source LLMs. The experimental results demonstrate that DoG effectively guarantees the generation of well-formed chains, thereby leading to higher accuracy. The key contributions of this work are: (1) We define well-formed chains as a principle for faithful and sound reasoning on KGs; (2) To achieve reasoning with well-form chains, we propose graph-aware constrained decoding for LLMs; (3) The experiments show that our approach exhibits the best performances on three KGQA benchmarks under a training-free setting.

2 Decoding on Graphs

2.1 Task Formulation

Given a KG $\mathcal{G}$ which can be represented by a set of relation triplets $t = (e, r, e^{'})$, where $e, e^{'}, r$ denote the head entity, tail entity, and corresponding relation respectively, the task of KGQA is to answer a natural language question q through finding the supporting information embedded in $\mathcal{G}$.

While prior works predominantly rely on specialized subgraph retrievers or iterative LLM-prompting to discover helpful information on KGs, we propose DoG to make LLMs to generate well-formed chains on KGs (§2.2) by enforcing graph-aware constrained decoding (§2.3) and beam search execution (§2.4), finally leading to better reasoning trajectories and answer candidates.

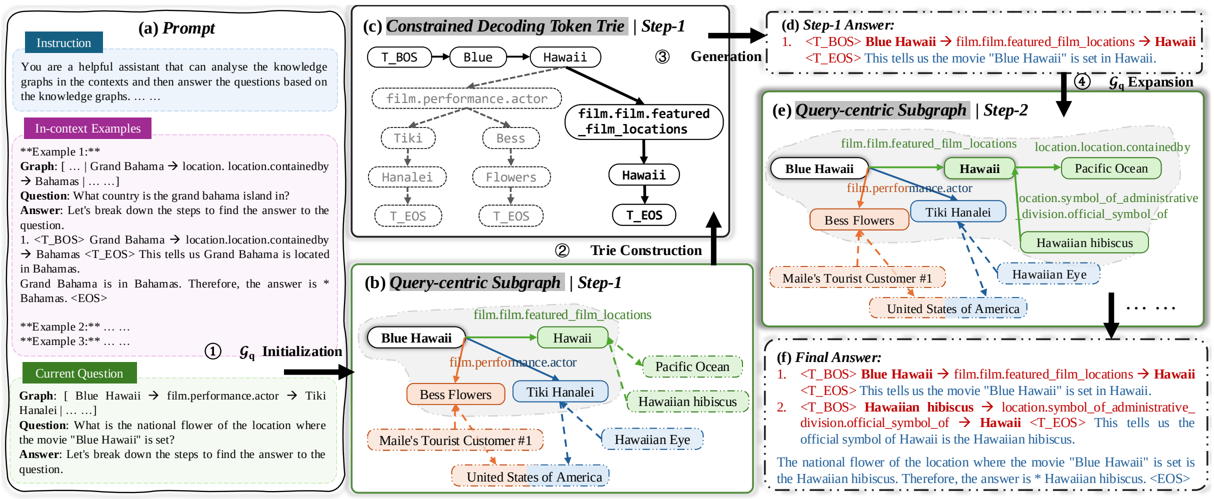

(a) Prompt
Instruction
You are a helpful assistant that can analyse the knowledge graphs in the contexts and then answer the questions based on the knowledge graphs. … …
In-context Examples
Example 1:
Graph: [… | Grand Bahama → location. location.containedby → Bahamas | … …]
Question: What country is the grand bahama island in?
Answer: Let's break down the steps to find the answer to the question.
1. <T_BOS> Grand Bahama → location.location.containedby → Bahamas <T_EOS> This tells us Grand Bahama is located in Bahamas.
Grand Bahama is in Bahamas. Therefore, the answer is * Bahamas. <EOS>
Example 2: … …
Example 3: … …
① $\mathcal{G}_q$ Initialization
Current Question
Graph: [Blue Hawaii → film.performance.actor → Tiki Hanalei | … …]
Question: What is the national flower of the location where the movie "Blue Hawaii" is set?
Answer: Let's break down the steps to find the answer to the question.
(c) Constrained Decoding Token Trie | Step-1
T_BOS
Blue
Hawaii
film.performance.actor
Tiki
Bess
Hanalei
Flowers
T_EOS
T_EOS
film.film.featured_film_locations
Hawaii
T_EOS
③ Generation
② Trie Construction
(b) Query-centric Subgraph | Step-1
film.film.featured_film_locations
Blue Hawaii
Hawaii
film.perrformance.actor
Pacific Ocean
Bess Flowers
Tiki Hanalei
Hawaiian hibiscus
Maile's Tourist Customer #1
Hawaiian Eye
United States of America
(d) Step-1 Answer:
1. <T_BOS> Blue Hawaii → film.film.featured_film_locations → Hawaii <T_EOS> This tells us the movie "Blue Hawaii" is set in Hawaii.
④ $\mathcal{G}_q$ Expansion
(e) Query-centric Subgraph | Step-2
film.film.featured_film_locations
location.location.containedby
Blue Hawaii
Hawaii
Pacific Ocean
film.perrformance.actor
location.symbol_of_administrative_division.official_symbol_of
Bess Flowers
Tiki Hanalei
Hawaiian hibiscus
Maile's Tourist Customer #1
Hawaiian Eye
United States of America
… …
(f) Final Answer:
1. <T_BOS> Blue Hawaii → film.film.featured_film_locations → Hawaii <T_EOS> This tells us the movie "Blue Hawaii" is set in Hawaii.
2. <T_BOS> Hawaiian hibiscus → location.symbol_of_administrative_division.official_symbol_of → Hawaii <T_EOS> This tells us the official symbol of Hawaii is the Hawaiian hibiscus.
The national flower of the location where the movie "Blue Hawaii" is set is the Hawaiian hibiscus. Therefore, the answer is * Hawaiian hibiscus. <EOS>

Figure 1: An example workflow of DoG with beam size of 1. The input consists of the instruction, three in-context learning examples and the current question, with the full prompt detailed in Tab. 6. The 2-hop input graph is illustrated in (b) and (e), including entities and relations in both solid and dotted lines. Starting from the query entity (white node), the query-centric subgraph $\mathcal{G}_q$ (grey area) is initialized by adding all triplets associated with the query entity, represented with solid lines in (b). The corresponding trie for constrained decoding is shown in (c), maintaining a set of valid tokens $w \in W_{val}$ for each position within Step-1. DoG chooses `(Blue Hawaii -> film.film.featured_film_locations -> Hawaii)` as Step-1 triplet, branch highlighted in **bold** within (c). Followed by unconstrained generation, Step-1 result is in (d) with well-formed chain in red and standard decoding in blue. The process advances to Step-2 in (e), where all triplets outside $\mathcal{G}_q$ that involve the two visited entities with **boldface** are added. The final answer is provided in (f).

2.2 Well-formed Chain

Based on a KG $\mathcal{G}$ and a question q, we aim to generate a well-formed chain, which is a sequence of fact triplets $T = \{t_1, t_2, ..., t_{|T|}\}$ and would lead to an answer.[2] Here we give the definition of well-formed chain. A chain of triplets is well-formed chain if

1. all the triplets along the chain exist on the KG, formally, $t_i \in \mathcal{G}$ for $1 \leq i \leq |T|$;

2. either head or tail entity of each triplet has been visited in the previous triplets or referred to as a query entity in the question q.

A well-formed chain is desirable for reasoning of KGQA in that (1) hallucination can be avoided during the reasoning process due to Property 1; (2) the search scopes for planning each reasoning step can be narrowed down, as only the neighbors of the already visited entities instead of the whole KG should be explored, according to Property 2.

We use in-context learning (Wei et al., 2021) for guiding the LLM to output a chain of triplets (we do not say well-formed chain here, as it is difficult to achieve this solely using in-context learning). The prompt that contains three in-context examples with the desired output format is demonstrated in Tab. 6. We transform the whole graph into a linearized form to accommodate text-form input.

Nevertheless, it is difficult to ensure the generation of well-formed chains by solely using in-context learning (§5.1). This is because, given the linearized form of a large-sized KG, LLMs may struggle to comprehend the structure of the graph, as well as the requirement of well-formed chains. The challenge will be even more pronounced when a chain of multiple triplets is required to solve multi-hop questions. Next, we apply graph-aware constrained decoding to open-source LLMs to guarantee the generation of well-formed chains.

2.3 Graph-Aware Constrained Decoding

As revealed by Wang and Zhou (2024), chain of thoughts (CoT) reasoning paths can be effectively elicited from LLMs by simply altering the decoding process. Inspired by this, DoG regulates the decoding process of LLMs with KG topology as a constraint for generating well-formed chains.

Query-centric Subgraph To effectively model the constraint, for question q, we maintain a local and query-centric subgraph $\mathcal{G}_q$ throughout the reason-

[2] A useful and well-formed chain does not necessarily include the answer as an entity, but contains all the information necessary for deriving the answer.

ing process. This subgraph is initialized as the set of triplets that contain the query entity e_q, formally, $\mathcal{G}_q = \{(e, r, e^{'}) \mid e_q \in \{e, e^{'}\}\}$. At the i-th step of the reasoning process, only those triplets on the query-centric subgraph, $t \in \mathcal{G}_q$, are allowed to be generated by the LLM. As the reasoning process proceeds, the subgraphs gradually expands in alignment with both the principle of well-formed chains and the topology of the source KG. Specifically, once $t_i = (e_i, r_i, e_i^{'})$ is generated as the i-th step result (as the LLM considers t_i is more possible than any other triplets on $\mathcal{G}_q$), all the triplets outside $\mathcal{G}_q$ that have e_i/$e_i^{'}$ as head/tail entity, will be incorporated into $\mathcal{G}_q$, as.

$$\mathcal{G}_q \leftarrow \mathcal{G}_q \cup \{(e, r, e^{'}) \in \mathcal{G} \mid \{e, e^{'}\} \cap \{e_i, e_i^{'}\} \neq \emptyset\}. \tag{1}$$

An example is shown in Fig. 1 (b) to (e): the initial query-centric subgraph (the shaded region in (b)) includes only the triplets containing "*Blue Hawaii*" (as head or tail entity) , which is the query entity in the question; The first step of the reasoning process generates a new entity "*Hawaii*", and the query-centric subgraph subsequently incorporates all the triplets containing "*Hawaii*", leading to an extended subgraph denoted by the shaded region in (e). Such a progressive expansion is essential because it enables $\mathcal{G}_q$ to precisely encompass all the possibilities of the next triplet as the chain continues.

It is easy to infer that constraining the generation of each reasoning step to the triplets on $\mathcal{G}_q$ can *ensure the well-formedness of generated triplet chains*. Note that query-centric subgraph described here are not the graphs used in input. The input incorporates a complete KG, of which the query-centric subgraph is a subgraph.

Constrained Decoding To force the output of each step to be a triplet on $\mathcal{G}_q$, we impose a dynamic constraint on the output vocabulary during LLMs' decoding, and this constraint only allows the output of those valid tokens $w \in W_{val}$. The set of valid tokens W_{val} contains only those tokens that, when combined with the prefix already generated, form a valid triplet on $\mathcal{G}_q$. However, even at the same step, W_{val} would change for different token positions. To update this constraint on-the-fly, we use a trie to keep track of W_{val}. For example, at the first step, the trie is constructed as in Fig.1(c) based on $\mathcal{G}_q$ in Fig.1(b). When the prefix generated so far within the first step is "*Blue Hawaii*", we have W_{val} = {"*film.performance.actor*", "*film.film.featured_film_location*"}, the set of the child nodes of "*Hawaii*" in the trie structure[3]. W_{val} is updated continuously according to the path from the root to the leaves. See App.D for more details on our implementation of the trie.

When generating the i-th token at each reasoning step, given W_{val}, the constraint is executed with modifying the logits of the output vocabulary as

$$logit_w = \begin{cases} logit_w, & w \in W_{val}, \\ -\infty \quad , & \text{otherwise}, \end{cases} \tag{2}$$

where $logit_w$ represents token w's logit score predicted by the LLM, which is then fed into a softmax function to obtain the final output distribution. Noteworthily, we do not modify the parameters or operating mechanism of LLMs, instead, we merely exclude those invalid generation. Consequently, the pre-trained LLMs' inherent reasoning capacity is maintained while being regularized by the KG topology to generate well-formed reasoning chains. For instance, in above case, between the two available tokens in W_{val}, the LLM is more likely to generate "*film.film.featured_film_location*" which is more relevant to the question.

On the other hand, unconstrained generation for intermediate analyses and final conclusion, *e.g.*, the text in blue in Fig. 1 (f), is also necessary. Therefore, the decoding constraint is lifted once a complete triplet is generated, allowing unconstrained generation of these content; When $< T\_BOS >$ (placeholder as the beginning of a triplet) is generated during the unconstrained generation, suggesting the LLM considers exploring the next triplet, the decoding constraint will take effect again; The generation of $< EOS >$ during the unconstrained generation will terminate the whole reasoning.

2.4 Beam Search for DoG

When generating a triplet chain, a vanilla practice is to produce only one triplet for each step[4]. This however may cause error propagation — inferring an irrelevant (though valid) triplet will negatively affect subsequent reasoning. To alleviate this problem, we further integrate DoG with triplet-level beam search execution, to make it possible to consider multiple valid triplets at each reasoning step.

[3]For the sake of simplicity in the demonstration, subtokens is not considered here.

[4]After the logit manipulation in Eq.2, greedy search, beam search or other sampling methods can be used to sample tokens

The complete approach is formalized in Algo. 1, involving a token-level beam search and a triplet-level beam search. We use beam search as the token-level sampling method as it can return multiple sequences (triplets here)[5]. The triplet-level beam search corresponds to the selection of triplet chains. Specifically, for each reasoning step, it retains bs (the hyperparameter for beam search) triplet chains with the highest chain scores. The chain score S, which measures the confidence of a chain, is the cumulative sum of all triplet scores in a chain. The triplet score s_t for a triplet t can be derived as

$$s_t = \log P_{\text{LLM}}(t|Q), \tag{3}$$

where Q denotes the so-far generation, including the input prompt and previous reasoning steps.

With beam search execution, for each reasoning step, DoG is able to explore multiple triplets in parallel and then retain the most plausible chains.

3 Experiment

3.1 Datasets and Knowledge Graphs

We evaluate DoG for KGQA on three benchmarks: WebQuestionSP (**WebQSP**, Yih et al., 2016), Complex WebQuestion (**CWQ**, Talmor and Berant, 2018) and **2Wikimultihop** (Ho et al., 2020). WebQSP and CWQ are two widely-used KGQA benchmarks with Freebase (Bollacker et al., 2008) as the background knowledge graph, containing up to 2-hop and 4-hop questions respectively. For each question in WebQSP, we extract all the entities within a 2-hop distance from the query entities on Freebase, and these entities along with the corresponding relations will constitute a KG. For CWQ, we do this with 4-hop distance. However, the resulting KGs are prohibitively large. We further utilize a lightweight text embedding model,`stella_en_400M_v5` [6], to rank the triplets based on their semantical similarities to the questions, and finally retain the top-120 triplets, leading to the final KGs used for evaluation.

2Wikimultihop is a more challenging benchmark with a large proportion of multi-hop questions that require different types of reasoning, like comparison and composition. Besides, some instances on the dataset offer the annotation of the ground truth reasoning path, allowing the assessment on the generated triplet chains. For each instance, 10 relevant passages are provided as context information. Following Li and Du (2023) and Fang et al. (2024), we instruct an LLM, `Gemma-2-9b-it` (Riviere et al., 2024), to extract the underlying knowledge graphs from the passages with in-context learning. We do not filter the resulting graphs as they are often moderate-sized. Note that the original passages are no longer used in input when taking evaluation. See App. B for more details on the graph construction.

Dataset	# instances	average graph size (# triplets)	% multi-hop question (>1 hop)
WebQSP	1542	119.71	2.01
CWQ	2617	119.64	28.58
2Wikimultihop	6964	64.58	100.00

Table 1: Data statistics of three processed datasets. The hop number of a question is measured by the length of the shortest path from the query entity to the answer question, which may be shorter than the ground truth path (which is unavailable for WebQSP, CWQ and part of 2Wikimultihop).

Due to the incompleteness of the source KG and the imperfection of the graph construction process, some instances may have a graph that excludes the gold answer. Finally, we filter out these instances[7], as we want to focus on LLMs' ability of reasoning on graphs. We found that the LLMs would rely on their internal knowledge to make a prediction if no plausible answer on the given KGs. The statistics of the processed datasets are shown in Tab. 1.

3.2 Implementation and Evaluation Metrics

To ascertain the general applicability of our approach, we apply DoG to three open-source LLMs, `Llama-3.1-8B-it`(Dubey et al., 2024), `Gemma-2-9b-it`(Riviere et al., 2024) and `Qwen-2.5-7b-it`(Qwen, 2024), where `-it` means instruction-tuned versions. We implement the decoding method using Huggingface (Wolf et al., 2020) framework. Notably, we refrain from fine-tuning of these models. Following previous works (Sun et al., 2024a; Baek et al., 2023; Jiang et al., 2023), we report the performances for all benchmarks using $Hits@1$, which measures the pro-

[5]One can also use sampling method like top-k/p (Holtzman et al., 2020) for the same purpose, but greedy search is not allowed here as it only returns at most one sequence.

[6]`https://huggingface.co/dunzhang/stella_en_400M_v5`

[7]For excluding these instances, we compare the names of all the entities name on the graph and the gold answer simply using text matching. However, this method can not exclude those false positive instances in which the answer entity appears on the KG but the relation paths from query entity to answer entity are not relevant to the corresponding questions.

portion of instances whose top-1 prediction is the ground truth answer.

3.3 Baselines

We compare DoG with four types of baselines. First, to validate the effectiveness of each design in DoG, we involve *Vanilla Baselines*: (1) **Direct Answering** is achieved with in-context learning, and the prompt is similar to the one used in DoG except that the triplet chains are omitted to encourage LLMs to give predictions directly. Further, we ablate the graph-aware constrained decoding component in DoG and obtain (2) **CoT**, which can be considered as a vanilla chain-of-thought (Wei et al., 2022) method. Secondly, to highlight the benefits of training-free DoG, we consider two baselines that train in-domain retrievers over KGs (*Specialized Retrievers*). (3) **RoG** (Luo et al., 2024) jointly trains LLMs for the generation of relation paths and the reasoning over valid retrieved KG paths. (4) **GNN-RAG** (Mavromatis and Karypis, 2024) trains a GNN retriever to select relevant KG reasoning paths. We reproduce the performance on our processed datasets using the two retrievers trained by the authors on CWQ and WebQSP. The retrieval results are then verbalized into the same LLM reasoners listed in §3.2 with Direct Answering. Thirdly, we show the effectiveness of decoding-based DoG over two *Iterative LLM-Prompting* approaches which heavily rely on the instruction-following abilities of LLMs and typically base their decisions on a single hop at a time. (5) **ToG** (Sun et al., 2024a) treats LLMs as an agent to iteratively execute beam search on KG by exploring relevant facts hop-by-hop. It first selects significant relations and then uses selected relations to guide entity exploration. (6) **Tree-of-Traversals** (Markowitz et al., 2024) is a zero-shot approach that augments LLMs with the interface of KGs. It repeatedly expands the local subgraph with an action state machine and tree search algorithm. We use the authors' open-source codes and their default settings for reproduction. Lastly, we compare the structure-decoding DoG over the *structure-training* method (7) **StructLM** (Zhuang et al., 2024), which is designed to train generalist models with a large amount of structured data. The results are reproduced using the model checkpoint and prompt provided by the authors.

We reproduce the results for each proposed baseline, as the settings in their original papers are distinct, especially on the choices of LLMs (e.g., **ToG** uses GPT4/ChatGPT and Llama2-70B-Chat as the backbone model) and the graph pre-processing. We thus re-implement these results under consistent settings for fair comparison: we use the same Llama 8B/Gemma 9B/Qwen 7B instruction-tuned models and the same input graphs for different approaches.

4 Main Results

Tab. 2 shows the results of `DoG` across three benchmarks. Some general patterns are manifest across different foundation LLMs:

Effectiveness of reasoning with triplet chains `Direct Answering` shows considerable performance on WebQSP, corroborating the findings by Dai et al. (2024) that pre-trained LLMs possess the ability of comprehending graph-structured data to some extent. However, the performance drops dramatically for CWQ and 2Wikimultihop which have a larger proportion of multi-hop questions (see Tab. 1). In comparison, `CoT` exhibits notably better performance on the two datasets, indicating the benefits brought by the reasoning trajectories in the form of triplet chains.

Effectiveness of graph-aware constrained decoding Even with identical input prompts, `DoG` (beam size = 1) outperforms `CoT` with notable gaps, verifying the effectiveness of graph-aware constrained decoding. The detailed analysis for the improvement is provided in §5.1. It is also noteworthy that `DoG` surpasses `StructLM` which underwent extensive pre-training on structured data, making graph-aware constrained decoding a more affordable yet better-performing alternative to the pre-training on structural data.

Effectiveness of beam search execution `DoG` shows improving performance with larger beam sizes in most cases. The improvements are prominent on CWQ and 2Wikimultihop while WebQSP sees limited gains, implying that the beam search execution could benefit multi-hop questions more.

Superiority of explicit reasoning over specialized retrievers The graph retrievers of `GNN-RAG` considerably boosts `Direct Answering`'s performances on CWQ, and even outperforms `CoT`. We found that the retriever always returns a highly concentrated subgraph which only contains a few relation paths towards the plausible answers, greatly reducing the difficulty of finding answers for LLMs. However, there are obvious degradations in the performance on 2Wikimultihop, highlighting the

	Approach	Llama 3.1-8B WebQSP	Llama 3.1-8B CWQ	Llama 3.1-8B 2Wikimultihop	Gemma 2-9B WebQSP	Gemma 2-9B CWQ	Gemma 2-9B 2Wikimultihop	Qwen 2.5 -7B WebQSP	Qwen 2.5 -7B CWQ	Qwen 2.5 -7B 2Wikimultihop
Vanilla Baselines	Direct Answering	87.55	67.10	54.58	84.95	60.68	58.85	88.72	67.41	50.92
	CoT	89.88	72.14	80.92	88.07	67.48	80.00	92.22	71.42	79.97
Iterative LLM Prompting	ToG	83.59	52.66	64.03	78.02	44.33	64.27	81.52	51.39	60.21
	Tree-of-Traversals	79.18	53.92	68.42	84.82	66.54	75.00	85.78	57.70	73.41
Specialized Retrievers	RoG	83.98	64.12	-	81.00	58.20	-	84.50	65.11	-
	GNN-RAG	87.42	74.89	42.00	83.53	68.51	26.78	86.96	71.38	37.54
Structure Training	StructLM* (**Mistral-7B**)	91.31	72.98	58.20	-	-	-	-	-	-
Ours	DoG(bs=1)	91.05	75.55	83.99	90.27	70.77	82.30	**92.67**	73.75	82.98
	DoG(bs=2)	90.99	76.08	83.54	**91.57**	72.56	83.49	92.60	73.71	83.31
	DoG(bs=3)	**91.38**	**76.16**	**84.06**	91.37	**74.10**	**84.06**	**92.67**	**74.17**	**84.16**

Table 2: Performance comparison of different methods on the three KGQA benchmarks. The best performance is highlighted in **bold**. All results are obtained using the same input graphs. Scores for `StructLM`* are reproduced using the open-sourced model (Zhuang et al., 2024), which is obtained by performing continual pre-training on `Mistral-7B-Instruct-v0.2`, using large-scale structural data. `RoG` cannot be applied on 2Wikimultihop, as its checkpoint trained on Freebase fails to generate relation labels of Wikidata, the background KG of 2Wikimultihop.

specially trained retrievers' inability to adapt to out-of-domain scenarios. Because the relation labels on 2Wikimultihop mainly follow the taxonomy of Wikidata, quite divergent from that of Freebase on which `GNN-RAG` and `RoG` are based. In contrast, `DoG` shows consistently enhanced performance across three benchmarks. `DoG` utilizes the pre-trained abilities of LLMs and textual information from verbalized KGs to make explicit reasoning, rendering it considerably robust to different KGQA tasks with diverse background KGs.

Superiority of direct generation over iterative prompting Although both `DoG` and iterative LLM-prompting are training-free methods, `DoG` shows superior performance, especially on CWQ and 2Wikimultihop. Iterative prompting places high demands on the LLMs' abilities, as our manual check found that the small LLMs used in this experiment often struggle on handling the instructions for some complex operations over KGs. To substantiate this, we further apply `Qwen-2.5-14b-it` to both `ToG` and `Tree-of-Traversals` and evaluate on CWQ (seen in App. C.2). With the 14B foundation model, they obtain obvious improvement over their counterparts with 7B models (ToG: 61.94 vs. 51.39, and Tree-of-Traversals: 63.55 vs. 57.70). `DoG` finds answers through the straightforward generation of a well-defined reasoning trajectory and thereby works well with small-scaled LLMs. Moreover, for both `ToG` and `Tree-of-Traversals`, only a very small portion of the grounded KG is visible to the LLMs when deciding operations over the KG, which may bring adverse effects to the path-planning process for finding answers. More discussion on this is in §5.2.

(a) Triplet-F1 ↑

Model	CoT 2-hop	CoT >2-hop	DoG(bs=1) 2-hop	DoG(bs=1) >2-hop	Ground-Truth ≥2-hop
Llama	75.77	66.80	79.85	79.93	100.00
Gemma	67.39	68.07	70.73	68.79	100.00
Qwen	75.35	66.19	80.41	74.61	100.00

(b) % ill triplet ↓

Model	CoT 2-hop	CoT >2-hop	DoG(bs=1) 2-hop	DoG(bs=1) >2-hop	Ground-Truth ≥2-hop
Llama	13.44	16.67	0.00	0.00	0.00
Gemma	13.41	11.26	0.00	0.00	0.00
Qwen	12.64	19.81	0.00	0.00	0.00

Table 3: Analysis of `CoT` and `DoG` (beam size = 1) on 2Wikimultihop for (a) Triplet-F1 and (b) % ill triplet.

5 Analysis

5.1 Effect of Graph-Aware Constrained Decoding

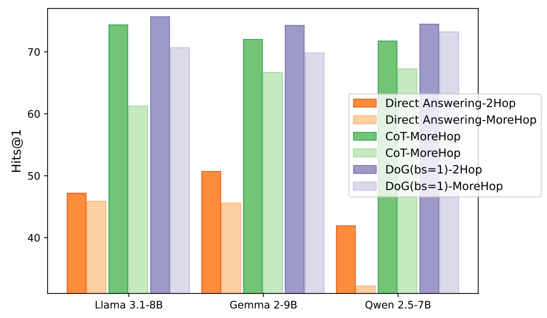


Figure 2: Performance of `Direct Answering`, `CoT` and `DoG` (beam size = 1) on 2Wikimultihop with 2-hop and >2 hop instances.

2Wikimultihop dataset provides the annotation of ground truth reasoning paths for some of the instances, with each step represented as a relation triplet. There are 4,843 such instances on our pro-

	Llama 3.1-8B			Gemma 2-9B			Qwen 2.5 -7B		
	WebQSP	CWQ	2Wikimultihop	WebQSP	CWQ	2Wikimultihop	WebQSP	CWQ	2Wikimultihop
DoG w/ local visibility	91.05	74.70	80.00	90.40	72.30	82.06	86.32	68.48	76.22
DoG w/ full visibility	90.99	76.08	83.54	91.57	72.56	83.49	92.60	73.71	83.31

Table 4: Performance of DoG (beam size = 2) with different levels of graph visibility.

	WebQSP		CWQ	
Model	DoG	DoG + GNN	DoG	DoG + GNN
Llama	90.99	90.66	76.08	84.72
Gemma	91.57	89.56	72.56	80.09
Qwen	92.60	90.14	73.71	84.37

Table 5: Performance of DoG (beam size = 2) with the integration of specialized retriever released by GNN-RAG.

cessed dataset. We split those instances into two parts based on the number of reasoning steps[8]. The evaluation on both parts for Direct Answering, CoT and DoG (the beam size is 1 for a fair comparison) is shown in Fig. 2.

As shown in Fig. 2, DoG demonstrates more pronounced improvements over CoT for >2-hop questions, compared to 2-hop questions. For an in-depth investigation on the gaps, we examine the accuracy of the triplet chains predicted by both methods. Here we define two metrics: (1) $Triplet-F1$ which represents the F1-score between predicted triplets and ground-truth triplets, (2) $\%\ ill\ triplet$ which measures the percentage of predicted triplets that break the well-formedness of a chain (§2.2). As shown in Tab. 3, all the ground-truth reasoning paths on 2Wikimultihop are well-formed chains ($\%ill\ triplet = 0$), implying that well-formed chains can aid in making accurate sequential reasoning on KGs. Accordingly, it can be observed that DoG consistently exhibits superior $Triplet-F1$ than CoT, attributable to DoG's complete elimination of ill triplets. Moreover, for >2-hop questions, the differences in both metrics between the two methods are more significant (except for the cases of Gemma). DoG's graph-aware constrained decoding effectively roots out the occurrence of ill triplets for both splits of questions, and, in turn, benefits the generation of high-quality triplet chains for multi-hop questions.

5.2 Effect of Global Visibility of Graphs

DoG incorporates a complete KG as input for global visibility of the graph. In this section, we attempt to alter this setting such that for each step, only the local query-centric graph (§2.3) instead of the complete KG is fed to the LLMs. The comparisons are illustrated in Tab. 4. With local visibility of graphs, DoG's performance incurs modest degradation. The visibility of global KG structure allows the LLMs to access information long distance from query entities, which perhaps helps plan better reasoning chains at early steps. Meanwhile, the local query-centric graph would be updated as the reasoning chain proceeds, preventing the Key-Value cache from being reused. DoG with local visibility thus has a noticeably slower inference speed.

5.3 Integration with Specialized Retriever

We have seen that the subgraph retriever of GNN-RAG can enhance the performances of Direct Answering on CWQ. In this section, we integrate DoG with this retriever by feeding the retrieved subgraphs instead of the complete KGs into DoG (beam size = 2). As presented in Tab. 5, with the concentrated subgraphs retrieved by GNN-RAG, DoG gets further improved on CWQ. This suggests that DoG sometimes could be adversely affected by noise from the complete KG and providing DoG with higher-quality KGs would lead to better performances. Nevertheless, training a specialized retriever is expensive, and the retriever may struggle with out-of-domain scenarios and return subpar subgraphs. In contrast, DoG maintains robust performances across KGs under various conditions.

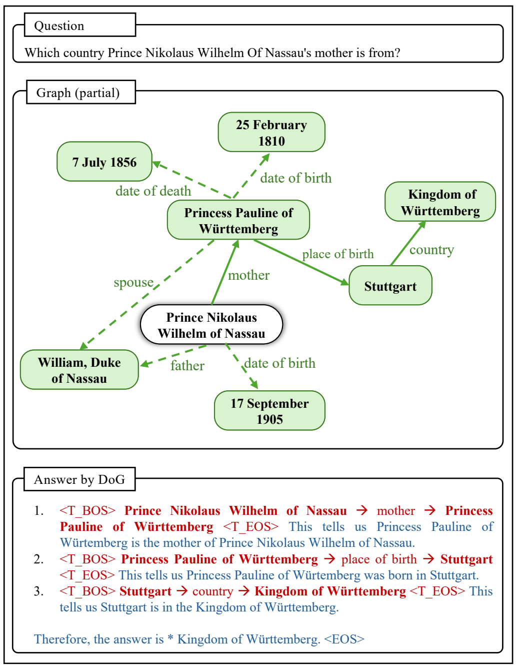
5.4 Case Study

Fig. 3 presents a 3-hop question from 2Wikimultihop dataset, along with the answer by DoG. DoG first deduces the correct reasoning path within the knowledge graph and subsequently derives the final conclusion based on this inferred path.

6 Related Works

Previous attempts on solving knowledge graph question answering (KGQA) often involve semantic parsing (Lan et al., 2022; Yu et al., 2023) that transforms questions into logical queries (e.g., SPARQL) to be executed over KG for answers

[8]The first part contains the questions with 2 hops, the second one is with >2 hops. We do not further split the second path in a finer-grained scheme as there is only a small number of >3-hop questions.



Which country Prince Nikolaus Wilhelm Of Nassau's mother is from?

1. <T_BOS> **Prince Nikolaus Wilhelm of Nassau** → mother → **Princess Pauline of Württemberg** <T_EOS> This tells us Princess Pauline of Würtemberg is the mother of Prince Nikolaus Wilhelm of Nassau.
2. <T_BOS> **Princess Pauline of Württemberg** → place of birth → **Stuttgart** <T_EOS> This tells us Princess Pauline of Würtemberg was born in Stuttgart.
3. <T_BOS> **Stuttgart** → country → **Kingdom of Württemberg** <T_EOS> This tells us Stuttgart is in the Kingdom of Württemberg.

Therefore, the answer is * Kingdom of Württemberg. <EOS>

Figure 3: A case of 3-hop question from 2Wikimultihop. The white node on the KG represents the query entity. The answer is given by DoG (bs=1, `Llama-3.1-8B-it`), with well-formed chain (constrained decoding) in red and standard decoding in blue.

(Lan and Jiang, 2020; Ye et al., 2022). These methods may require ground-truth logical queries for training, and model-generated queries might be non-executable due to syntax or semantic errors (Yu et al., 2023; Luo et al., 2024; Mavromatis and Karypis, 2024). Recently, researchers have been exploring the joint use of KGs and large language models (LLMs) for KGQA, leveraging the advanced capabilities of LLMs for enhanced reasoning (Salnikov et al., 2023; Wu et al., 2023; He et al., 2024). Various approaches (Jin et al., 2024; Sun et al., 2024b) *iteratively prompt* LLMs to synergize with KGs to select relevant information step by step (Ren et al., 2024). ToG (Sun et al., 2024a) prompts LLMs to explore relevant facts hop-by-hop using beam search on KG. Tree-of-Traversals (Markowitz et al., 2024) is a zero-shot reasoning algorithm motivated by Tree-of-Thoughts (Yao et al., 2023). Instead of focusing on one hop at a time, DoG uses the entire question graph as input, enabling reasoning over complete knowledge. Additionally, these prompting approaches overly rely on the instruction-following abilities of LLMs, posing challenges when applied to compact LLMs. Some studies (He et al., 2024) focus on training *specialized retrievers* to extract key information from KG and the retrieved knowledge as input to LLMs. RoG (Luo et al., 2024) and GNN-RAG (Mavromatis and Karypis, 2024) train in-domain LLM and GNN retrievers respectively. StructLM (Zhuang et al., 2024), on the other hand, trains generalist models over large amount of structured data to augment the structured knowledge grounding capabilities of LLMs. However, fine-tuning large scale models is computationally expensive, and specialized training suffers from limited generalization capabilities. Conversely, DoG employs a general graph-aware decoding approach, eliminating the need for external retrievers or specialized training.

7 Conclusion

We present DoG, a framework that integrates LLMs' reasoning capabilities with KGs' structural knowledge in a tightly coupled manner. The graph-aware decoding component of DoG effectively regulates the decoding process of LLMs with the topology of the KG, to enable the generation of well-formed chains, thereby leading to reasoning trajectories that are sound and faithful to the KG. Experiments demonstrate that DoG exhibits consistently robust performances across different KGs.

Acknowledgement

This study was supported by the Centre for Perceptual and Interactive Intelligence (CPII) Ltd., a CUHK-led InnoCentre under the InnoHK initiative of the Innovation and Technology Commission of the Hong Kong Special Administrative Region Government.

Limitations

While DoG demonstrates notable performance on KGQA task through training-free graph-aware constrained decoding, there are some limitations to consider. First, unlike specialized subgraph retrievers and iterative LLM-based prompting approaches, DoG processes the entire question graph as input to facilitate faithful and sound reasoning over knowledge graphs (KGs) LLMs directly by LLMs. This, however, requires a larger context window in LLMs to accommodate both the graph and in-context examples. Secondly, although DoG requests the same number of forward passes as the vanilla chain-of-thoughts (CoT) prompting due to the use of Key-Value cache, it is slower in practice. This slowdown occurs because the query-centric subgraph must be updated after each reasoning step. Nevertheless, analysis in App. C.1 shows our method achieves a better balance between time and performance. Further work on engineering solutions could be implemented to enhance the efficiency of LLMs when reasoning over KGs. Lastly, our experiments and evaluations have been limited to English-based benchmarks. To ensure the robustness and reliability, it is important to extend these assessments to other languages, as this will help identify any language-specific issues or inconsistencies.

Ethics Statement

The proposed method, DoG, is a general graph-aware decoding approach designed to enhance reasoning capabilities over public knowledge graphs (KGs), thereby reducing the reliance on the prompting engineering of LLMs. While we do not expect DoG iteself to introduce new areas of risk, it is essential to consider the broader ethical impacts and acknowledge the potential risks associated with existing pre-trained large language models (LLMs) and KGs. (1) Both LLMs and KGs may inherently contain social biases or factual inaccuracies due to the nature of how they are constructed from real-world data. There is a risk that DoG may generate biased or incorrect outputs with such backbone models for KG reasoning. (2) We have not conducted an extensive safety analysis to assess the performance of DoG under conditions involving misleading or deceptive knowledge graphs.

References

OpenAI Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, et al. 2023. Gpt-4 technical report.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*.

Jinheon Baek, Alham Fikri Aji, and Amir Saffari. 2023. Knowledge-augmented language model prompting for zero-shot knowledge graph question answering. In *Proceedings of the 1st Workshop on Natural Language Reasoning and Structured Explanations (NLRSE)*, pages 78–106, Toronto, Canada. Association for Computational Linguistics.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, page 1247–1250, New York, NY, USA. Association for Computing Machinery.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Xinbang Dai, Yuncheng Hua, Tongtong Wu, Yang Sheng, Qiu Ji, and Guilin Qi. 2024. Large language models can better understand knowledge graphs than we thought.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, et al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Jinyuan Fang, Zaiqiao Meng, and Craig Macdonald. 2024. Trace the evidence: Constructing knowledge-grounded reasoning chains for retrieval-augmented generation. *ArXiv*, abs/2406.11460.

Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. 2023. Towards revealing the mystery behind chain of thought: A theoretical perspective. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Gaole He, Yunshi Lan, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Improving multi-hop knowledge base question answering by learning intermediate supervision signals. In *WSDM*.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V. Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2024. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. *Preprint*, arXiv:2402.07630.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Ruixin Hong, Hongming Zhang, Hong Zhao, Dong Yu, and Changshui Zhang. 2023. Faithful question answering with Monte-Carlo planning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3944–3965, Toronto, Canada. Association for Computational Linguistics.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Xin Zhao, and Ji-Rong Wen. 2023. StructGPT: A general framework for large language model to reason over structured data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9237–9251, Singapore. Association for Computational Linguistics.

Bowen Jin, Chulin Xie, Jiawei Zhang, Kashob Kumar Roy, Yu Zhang, Zheng Li, Ruirui Li, Xianfeng Tang, Suhang Wang, Yu Meng, and Jiawei Han. 2024. Graph chain-of-thought: Augmenting large language models by reasoning on graphs. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 163–184, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Yunshi Lan, Gaole He, Jinhao Jiang, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2022. Complex knowledge base question answering: A survey. *Preprint*, arXiv:2108.06688.

Yunshi Lan and Jing Jiang. 2020. Query graph generation for answering multi-hop complex questions from knowledge bases. In *Annual Meeting of the Association for Computational Linguistics*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Kun Li, Tianhua Zhang, Liping Tang, Junan Li, Hongyuan Lu, Xixin Wu, and Helen Meng. 2022. Grounded dialogue generation with cross-encoding re-ranker, grounding span prediction, and passage dropout. In *Proceedings of the Second DialDoc Workshop on Document-grounded Dialogue and Conversational Question Answering*, pages 123–129, Dublin, Ireland. Association for Computational Linguistics.

Ruosen Li and Xinya Du. 2023. Leveraging structured information for explainable multi-hop question answering and reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 6779–6789, Singapore. Association for Computational Linguistics.

Linhao Luo, Yuan-Fang Li, Reza Haf, and Shirui Pan. 2024. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *The Twelfth International Conference on Learning Representations*.

Elan Markowitz, Anil Ramakrishna, Jwala Dhamala, Ninareh Mehrabi, Charith Peris, Rahul Gupta, Kai-Wei Chang, and Aram Galstyan. 2024. Tree-of-traversals: A zero-shot reasoning algorithm for augmenting black-box language models with knowledge graphs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12302–12319, Bangkok, Thailand. Association for Computational Linguistics.

Costas Mavromatis and George Karypis. 2024. Gnn-rag: Graph neural retrieval for large language model reasoning. *Preprint*, arXiv:2405.20139.

Riccardo Orlando, Pere-Lluís Huguet Cabot, Edoardo Barba, and Roberto Navigli. 2024. ReLiK: Retrieve and LinK, fast and accurate entity linking and relation extraction on an academic budget. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 14114–14132, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

Qwen. 2024. Qwen2.5: A party of foundation models.

Xubin Ren, Jiabin Tang, Dawei Yin, Nitesh Chawla, and Chao Huang. 2024. A survey of large language models for graphs. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, page 6616–6626, New York, NY, USA. Association for Computing Machinery.

Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, L'eonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ram'e, Johan Ferret, Peter Liu, Pouya Dehghani Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, Nino Vieillard, et al. 2024. Gemma 2: Improving open language models at a practical size. *ArXiv*, abs/2408.00118.

Mikhail Salnikov, Hai Le, Prateek Rajput, Irina Nikishina, Pavel Braslavski, Valentin Malykh, and Alexander Panchenko. 2023. Large language models meet knowledge graphs to answer factoid questions. In *Proceedings of the 37th Pacific Asia Conference on Language, Information and Computation*, pages 635–644, Hong Kong, China. Association for Computational Linguistics.

Nasim Shirvani-Mahdavi, Farahnaz Akrami, Mohammed Samiul Saeef, Xiao Shi, and Chengkai Li. 2023. Comprehensive analysis of freebase and dataset creation for robust evaluation of knowledge graph link prediction models. In *The Semantic Web – ISWC 2023*, pages 113–133, Cham. Springer Nature Switzerland.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel Ni, Heung-Yeung Shum, and Jian Guo. 2024a. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *The Twelfth International Conference on Learning Representations*.

Lei Sun, Zhengwei Tao, Youdi Li, and Hiroshi Arakawa. 2024b. ODA: Observation-driven agent for integrating LLMs and knowledge graphs. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 7417–7431, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 641–651, New Orleans, Louisiana. Association for Computational Linguistics.

Denny Vrandeić and Markus Krötzsch. 2014. Wikidata. *Communications of the ACM*, 57:78 – 85.

Keheng Wang, Feiyu Duan, Sirui Wang, Peiguang Li, Yunsen Xian, Chuantao Yin, Wenge Rong, and Zhang Xiong. 2023. Knowledge-driven cot: Exploring faithful reasoning in llms for knowledge-intensive question answering. *Preprint*, arXiv:2308.13259.

Xuezhi Wang and Denny Zhou. 2024. Chain-of-thought reasoning without prompting. *ArXiv*, abs/2402.10200.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2021. Finetuned language models are zero-shot learners. *ArXiv*, abs/2109.01652.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing. *Preprint*, arXiv:1910.03771.

Yike Wu, Nan Hu, Sheng Bi, Guilin Qi, Jie Ren, Anhuan Xie, and Wei Song. 2023. Retrieve-rewrite-answer: A kg-to-text enhanced llms framework for knowledge graph question answering. *Preprint*, arXiv:2309.11206.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Xi Ye, Semih Yavuz, Kazuma Hashimoto, Yingbo Zhou, and Caiming Xiong. 2022. RNG-KBQA: Generation augmented iterative ranking for knowledge base question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6032–6043, Dublin, Ireland. Association for Computational Linguistics.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206, Berlin, Germany. Association for Computational Linguistics.

Zhangyue Yin, Qiushi Sun, Qipeng Guo, Jiawen Wu, Xipeng Qiu, and Xuanjing Huang. 2023. Do large language models know what they don't know? In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8653–8665, Toronto, Canada. Association for Computational Linguistics.

Donghan Yu, Sheng Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Yiqun Hu, William Yang Wang, Zhiguo Wang, and Bing Xiang. 2023. DecAF: Joint decoding of answers and logical forms for question answering over knowledge bases. In *The Eleventh International Conference on Learning Representations*.

Alex Zhuang, Ge Zhang, Tianyu Zheng, Xinrun Du, Junjie Wang, Weiming Ren, Wenhao Huang, Jie Fu, Xiang Yue, and Wenhu Chen. 2024. StructLM: Towards building generalist models for structured knowledge grounding. In *First Conference on Language Modeling*.

A Prompts

Tab. 6 presents the complete prompt used for DoG over three KGQA benchmarks. Three in-context learning examples with the desired output format are listed.

B Graph Construction

For Complex WebQuestion (CWQ, Talmor and Berant, 2018) and WebQuestionSP (WebQSP, Yih et al., 2016), we use the Freebase data provide by He et al. (2021) as the source graphs. There are CVT nodes (Shirvani-Mahdavi et al., 2023) in Freebase knowledge graph for modeling n-ary relationships, the CVT nodes do not have real meaning.

```
You are a helpful assistant that can analyse the knowledge graphs in the contexts and then answer the questions based on the knowledge graphs.
The answers should give the grounded reasoning chains and think step by step, and the reasoning chains should be logically complete but have as fewer steps as possible. Do not include information irrelvant to the question.


**Example 1:**

Context: [ Bahamas -> location.country.first_level_divisions -> Grand Cay | Grand Bahama -> location.location.containedby -> Bahamas | Bahamas -> location.location.contains -> Grand Cay | Bahamas -> location.location.contains -> Grand Bahama | Grand Cay -> location.location.containedby -> Bahamas | Bahamas -> location.country.first_level_divisions -> East Grand Bahama | Bahamas -> location.country.first_level_divisions -> West Grand Bahama | Grand Bahama -> location.location.contains -> Grand Bahama International Airport | Bahamas -> location.location.contains -> East Grand Bahama | Bahamas -> location.location.contains -> West Grand Bahama | East Grand Bahama -> location.location.containedby -> Bahamas | Bahamas -> location.location.contains -> Grand Bahama International Airport | Grand Bahama -> location.location.people_born_here -> Hubert Ingraham | Grand Cay -> location.administrative_division.first_level_division_of -> Bahamas | Bahamas -> location.country.administrative_divisions -> Cat Island, Bahamas | Bahamas -> location.country.administrative_divisions -> Long Island | West Grand Bahama -> location.location.containedby -> Bahamas | Bahamas -> location.country.capital -> Nassau | Bahamas -> location.country.administrative_divisions -> Inagua | Bahamas -> location.country.administrative_divisions -> Exuma | Grand Bahama International Airport -> location.location.containedby -> Bahamas | Grand Bahama -> location.location.people_born_here -> Juan Lewis | Grand Bahama -> location.location.contains -> West End Airport ]

Question: What country is the grand bahama island in?

Answer: Let's break down the steps to find the answer to the question.

1. < Grand Bahama -> location.location.containedby -> Bahamas > This tells us Grand Bahama is located in Bahamas.

Grand Bahama is in Bahamas. Therefore, the answer is * Bahamas.


**Example 1:**

Context: [ William Shakespeare -> people.person.profession -> Playwright | William Shakespeare -> people.person.profession -> Poet | William Shakespeare -> base.kwebbase.kwtopic.has_sentences -> By the time these works were published in 1609, Shakespeare was an acknowledged master of drama and an established country gentleman. | William Shakespeare -> people.person.profession -> Actor | William Shakespeare -> people.person.profession -> Author | William Shakespeare -> people.person.profession -> Lyricist | In the 21 years between 1592 and 1613, Shakespeare produced more than 30 plays. -> base.kwebbase.kwsentence.previous_sentence -> Above all, his humanity spanned all classes and circumstances ]

Question: What did William Shakespeare do for a living?

Answer: Let's break down the steps to find the answer to the question.

1. < William Shakespeare -> people.person.profession -> Playwright > This tells us William Shakespeare is was playwright.
2. < William Shakespeare -> people.person.profession -> Poet > This tells us William Shakespeare was a poet.

William Shakespeare was a playwright, and poet. Therefore, the answer is * playwright, and * poet.


**Example 3:**

Context: [ Carlton the Bear -> sports.mascot.team -> Toronto Maple Leafs | Toronto Maple Leafs -> sports.sports_team.team_mascot -> Carlton the Bear | Carlton the Bear -> common.topic.notable_types -> Mascot | Mascot -> type.type.properties -> Team | Toronto Maple Leafs -> sports.sports_team.previously_known_as -> Toronto St. Patricks | Team -> type.property.master_property -> Team Mascot | Toronto Maple Leafs -> sports.sports_team.previously_known_as -> Toronto Arenas | m.0crt465 -> sports.sports_league_participation.team -> Toronto Maple Leafs | Toronto St. Patricks -> sports.defunct_sports_team.later_known_as -> Toronto Maple Leafs | Toronto Maple Leafs -> sports.sports_team.sport -> Ice Hockey | Toronto St. Patricks -> sports.sports_team.sport -> Ice Hockey | Toronto Arenas -> sports.defunct_sports_team.later_known_as -> Toronto Maple Leafs | Toronto -> sports.sports_team_location.teams -> Toronto Maple Leafs | Toronto Maple Leafs -> sports.sports_team.location -> Toronto ]

Question: What is the sport played by the team with a mascot known as Carlton the Bear?

Answer: Let's break down the steps to find the answer to the question.

1. < Carlton the Bear -> sports.mascot.team -> Toronto Maple Leafs > This tells us Carlton the Bear is the mascot of the team Toronto Maple Leafs.
2. < Toronto Maple Leafs -> sports.sports_team.sport -> Ice Hockey > This tells us Toronto Maple Leafs plays Ice Hockey.

Carlton the Bear is the mascot of the team Toronto Maple Leafs which plays Ice Hockey. Therefore, the answer is * Ice Hockey.


**Example 4:**

Context: [ {graph} ]

Question: {question}

Answer: Let's break down the steps to find the answer to the question.
```

Table 6: Prompt for DoG on KGQA task. Three in-context examples with the desired output format are listed.

To remove CVT nodes on our dataset, we preprocess the KG to convert n-ary relationships to binary relationships by concatenating the edge labels by "-" . For each example, we apply a text embedding model, `stella_en_400M_v5` [9] to encode the question and all the triplet in the source graph into embeddings respectively. The input of a triplet is in the form of (`{head_entity}`, `{relation}`, `{tail_entity}`). We sort all the triplets in descending order of cosine similarity with the question embedding. A new graph G is initialized as an empty set, and we gradually add the top-ranked triplet to this new graph. To maintain the connectivity of G, each time a top-ranked triplet t is added, all the intermediate triplets along the path from the query entity to the head/tail entity of t are also added to G. This process repeats until the size of G reaches 120.

2Wikimultihop dataset provides 10 context passages for each question. We first utilize an entity linking tool, `relik-entity-linking-large` (Orlando et al., 2024) [10], to identify distinct entities mentioned in the passages and link them to a unique identifier in Wikidata. Given the passage and all the entities identified in the passage, we instruct an LLM, `Gemma-2-9b-it`(Riviere et al., 2024) to extract all the relation triplets using in-context learning (Wei et al., 2021). The prompt is detailed in Tab. 7. Finally, all the extracted triplets from different passages are combined into the final graph, with no ranking or filtering applied to the triplets.

C Additional Experiments

C.1 Time complexity analysis

Approach	Average time cost (second per example)
Direct Answering	2.82
CoT	12.74
ToG	15.74
Tree-of-Traversals	37.37
DoG (beam size = 1)	14.56

Table 8: Average time used for running the first 100 instances on WebQSP.

Here we provide the average running time over the first 100 instances on WebQSP testset. Direct Answering, CoT, and DoG are all implemented with Huggingface framework, while ToG and Tree-of-Traversal are implemented by prompting the LLMs deployed locally using vLLM which provides better inference acceleration.

As shown in Tab.8, Direct Answering runs most quickly as it directly outputs answers which are often a short sentence or entities. CoT and DoG take more time as they both output a complete reasoning trajectory before the final answer, instead of a short answer. The slightly longer time our method takes compared to CoT is due to the update of the decoding constraint after each reasoning step. Nonetheless, the constraint effectively makes the LLMs output well-formed chains, in turn, leading to significantly better performance. We believe the additional time cost of the constrained decoding is well worth it. Besides, the two iterative prompting based methods, ToG and Tree-of-Traversals, have the slowest speed as they both prompt the LLMs multiple times with different inputs.

C.2 Experiments on larger-scale LLMs

Approach	Hits@1
Direct Answering	71.15
CoT	75.51
RoG	65.53
GNN-RAG	69.62
ToG	63.55
Tree-of-Traversals	61.94
DoG (beam size = 1)	77.65

Table 9: Results of various methods with Qwen2.5-14B-Instruct.

We apply Qwen2.5-14B-Instruct to various evaluated methods, and the results are shown in Tab. 9. With a larger foundation model, almost all the methods except GNN-RAG receive some improvement over their 7B-sized counterparts. Notably, DoG still has the best performance.

D Implementation of the Trie for Graph-Aware Constrained Decoding

We use a dictionary structure to implement a trie. Given a query-centric subgraph (§2.3), the function `build_trie()` in the code block of Tab. 10 is used to build the dictionary. Then, as demonstrated in the function `find_valid_tokens()`, for each iteration of token generation, with the so-far generation at the current reasoning step as the key, we can efficiently get the set of valid tokens by performing a lookup in the dictionary.

[9] https://huggingface.co/dunzhang/stella_en_400M_v5

[10] https://huggingface.co/sapienzanlp/relik-entity-linking-large

Given the documents and some entities within the documents, extract all the relation triplets between any pairs of the entities.

Document 1:
Title: The Return of Dr. Fu Manchu
The Return of Dr. Fu Manchu is a 1930 American pre-Code film directed by Rowland V. Lee. It is the second of three films starring Warner Oland as the fiendish Fu Manchu, who returns from apparent death in the previous film," The Mysterious Dr. Fu Manchu"(1929), to seek revenge on those he holds responsible for the death of his wife and child.

Entities:The Return of Dr. Fu Manchu\n1930\nUnited States\nPre-Code Hollywood\nRowland V. Lee\nWarner Oland\nFu Manchu\nThe Mysterious Dr. Fu Manchu\n1929

Relation triplets: The Return of Dr. Fu Manchu->country->United States\nThe Return of Dr. Fu Manchu->director->Rowland V. Lee\nThe Return of Dr. Fu Manchu->movement->Pre-Code Hollywood\nThe Return of Dr. Fu Manchu->publication date->1930\nThe Return of Dr. Fu Manchu->cast member->Warner Oland\nThe Vengeance of Fu Manchu->cast member->Warner Oland\nThe Mysterious Dr. Fu Manchu->cast member->Warner Oland\nThe Mysterious Dr. Fu Manchu->country->United States\nThe Mysterious Dr. Fu Manchu->publication date->1929

Document 2:
Title: Now, Voyager
Now, Voyager is a 1942 American drama film starring Bette Davis, Paul Henreid, and Claude Rains, and directed by Irving Rapper. The screenplay by Casey Robinson is based on the 1941 novel of the same name by Olive Higgins Prouty. Prouty borrowed her title from the Walt Whitman poem" The Untold Want", which reads in its entirety, In 2007," Now, Voyager" was selected for preservation in the United States National Film Registry by the Library of Congress as being" culturally, historically, or aesthetically significant." The film ranks number 23 on" AFI's 100 Years ... 100 Passions", a list of the top love stories in American cinema. Film critic Steven Jay Schneider suggests the film continues to be remembered due not only to its star power, but also the" emotional crescendos" engendered in the storyline.

Entities: "1942\nUnited States\nDrama (film and television)\nBette Davis\nPaul Henreid\nClaude Rains\nIrving Rapper\nCasey Robinson\n1941\nNow, Voyager (novel)\nOlive Higgins Prouty\nL. Fletcher Prouty\nWalt Whitman\n2007\nNow, Voyager\nNational Film Registry\nLibrary of Congress\nCity Lights\nAmerican Film Institute\nAFI's 100 Years ... 100 Passions\nSteven Jay Schneider"

Relation triplets: Now, Voyager->country->United States\nNow, Voyager->director->Irving Rapper\nNow, Voyager->genre->Drama (film and television)\nNow, Voyager->publication date->1942\nNow, Voyager->cast member->Bette Davis\nNow, Voyager->cast member->Paul Henreid\nNow, Voyager->cast member->Claude Rains\nNow, Voyager->screenplay writer->Casey Robinson\nNow, Voyager->source material->Now, Voyager (novel)\nNow, Voyager (novel)->publication date->1941\nNow, Voyager (novel)->author->Olive Higgins Prouty\nNow, Voyager->preserved by->National Film Registry\nNow, Voyager->ranked in->AFI's 100 Years ... 100 Passions\nNow, Voyager->analyzed by->Steven Jay Schneider\nNational Film Registry->maintained by->Library of Congress

Document 3:
Title: {Title}
{Content}

Entities:{Entity List}

Relation triplets:

Table 7: Prompt for relation extraction on 2Wikimultihop.

```python
def build_trie(sub_graph: list[list[int]]) -> dict[int, dict]:
    # sub_graph: the list that contains the token ids of all the triplet, triplets
    are in the form of "<T_BOS> <Triplet> <T_EOS>".
    trie = dict()
    for triplet in sub_graph:
        for suffix in [triplet[i:] for i in range(len(triplet))]:
            node = trie
            for token in suffix:
                if token not in node:
                    node[token] = dict()
                node = node[token]

    return trie

def find_valid_tokens(trie: dict[int, dict], prefix: list[int]) -> list[int]:
    # prefix: the so-far genetation at at the current reasoning step
    v = trie
    for k in prefix:
        v = v.get(k, {})
    return list(v.keys())
```

Table 10: Code for implementation of the trie for graph-aware constrained decoding.

Algorithm 1 DoG with beam search

```
Input: Pre-trained $LLM$, knowledge graph $\mathcal{G}$, input prompt $Q$ (containing $\mathcal{G}$ and question $q$), query
    entity $e_q$, beam size $bs$, maximum number of steps $T$
 1: $\mathcal{G}_q \leftarrow \{(e, r, e^{'}) \in \mathcal{G} \mid e_q \in \{e, e^{'}\}\}$ ▷ initialize query-centric subgraph
 2: $\mathcal{P} \leftarrow \{(Q, \mathcal{G}_q, 0)\}$ ▷ initialize candidate pool with candidates comprised of input, a query-centric
    subgraph and a chain score
 3: for $t = 1, ..., T$ do
 4:     // Triplet-level beam search
 5:     $\mathcal{P}^{'} \leftarrow \emptyset$
 6:     for $i = 1, ..., |\mathcal{P}|$ do
 7:         $Q, \mathcal{G}_q, S \leftarrow \mathcal{P}_i$
 8:         $(r_1, s_1), ..., (r_{bs}, s_{bs}) \sim LLM(\cdot|Q, \mathcal{G}_q)$ ▷ sample $bs$ triplets along with the triplet scores from
            $LLM$ based on input $Q$ and constraint $\mathcal{G}_q$, using token-level beam search
 9:         for $j = 1, ..., bs$ do
10:             $\mathcal{G}_q^{'}, \leftarrow Update(\mathcal{G}_q, \mathcal{G}, r_j)$ ▷ update $\mathcal{G}_q$ for $r_j$ as in Eq. 1
11:             $\mathcal{P}^{'} \leftarrow \mathcal{P}^{'} \cup \{(concate(Q, r_j), \mathcal{G}_q^{'}, S + s_j)\}$ ▷ update the input with the latest reasoning step,
                and add the triplet score to the chain score
12:         end for
13:     end for
14:     $\mathcal{P} \leftarrow Top(\mathcal{P}^{'}, bs)$ ▷ retain the top-$bs$ candidates in $P^{'}$ based on their chain scores
15: end for
16: return $\mathcal{P}_1$
```