DCMKC: A Dual Consistency Matching Approach for Multi-hop Question Answering in LLMs

Xinyi Wang[1], Yiping Song[1]*, Chang Liu[1], Tingjin Luo[1], Bo Liu[2], Zheng Xie[1], Minlie Huang[3]

[1]National University of Defense Technology
[2]Academy of Military Sciences
[3]The CoAI Group, Department of Computer Science and Technology, Tsinghua University

Correspondence: songyiping@nudt.edu.cn

Abstract

Reasoning based on chains of thought (CoTs) enables large language models (LLMs) to solve problems by thinking step by step and becomes the mainstream solution for Question-Answering (QA) tasks. Knowledge graph (KG)-enhanced CoT technology helps correct factual errors or predict reasoning direction. Existing KG-enhanced methods find relevant information in KGs "within" each reasoning step of CoTs. However, in some cases, logical connections "between" reasoning steps may be missing or wrong, leading to broken reasoning chains and wrong reasoning direction. To solve the above problem, we argue that the errors between reasoning steps require collaborative verification and mining of multiple triplets and multiple paths in KG. So we propose the **DCMKC** (Dual Consistency Matching for KG and CoT) method, aiming to maintain semantic and structural consistency between KG and CoT. The main idea is to convert CoTs and KGs into two granularity-aligned graphs, transforming multi-hop reasoning and KG matching into iterative matching and modification of two graphs. In each iteration, DCMKC matches the KG reasoning chains with CoTs based on semantic similarity and judges the structural consistency between them. Then it modifies CoTs using the matched chains. After iterations, the CoTs and KG reasoning chains reach high semantic and structural consistency, which is theoretically and experimentally demonstrated by kernel and spectral methods. The two kinds of chains are then used to generate the final answers. Experimental results show that our method outperforms baselines on multiple datasets, especially on multi-answer questions, with up to 5.1% improvement over the baseline. Our code is available at https://github.com/suyun417/DCMKC.

1 Introduction

The Question-Answering (QA) task is an important benchmark for evaluating the ability of large language models (LLMs) to understand and process natural language (Shailendra et al., 2024). We can evaluate the performance of LLMs in advanced cognitive functions such as natural language understanding, semantic analysis, and knowledge reasoning through the QA task. The QA task with LLMs has shown broad application prospects, including intelligent customer service (Xu et al., 2024), financial analysis (Panwar et al., 2023), medical inquiry (Lucas et al., 2024), and other scenarios.

Early works use Knowledge Graph (KG)-enhanced LLMs as the core technique for QA tasks. They construct reasoning steps with high accuracy and interpretability by walking on the KG with LLMs (Fang et al., 2024; Mavromatis and Karypis, 2024; Dong et al., 2024; Sanmartin, 2024; Li et al., 2024). However, the KG has a complex structure, which makes it difficult for the LLMs to identify the knowledge and logical relationships in the KG (Pan et al., 2023). To better acquire the key information, subsequent works use the chain of thought (CoT) to identify key details, delineate the scope, and offer guidance for graph reasoning (Wang et al., 2024; Jin et al., 2024; Wang et al., 2023a). However, semantic inconsistency exists because the CoT and the KG may describe knowledge differently. It will cause errors when matching and locating knowledge. Furthermore, the CoT is usually used to locate relevant entities in the KG, failing to use its mechanism to explore the commonsense knowledge and reasoning capability of the LLM itself.

Another type of work is based on CoTs. They offer examples with intermediate steps (Wei et al., 2022) or prompt the LLM to think step by step (Kojima et al., 2022), enabling the LLM to show the reasoning process. This not only improves the interpretability of the answers but also offers a traceable

*Corresponding author

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 259–273
November 4-9, 2025

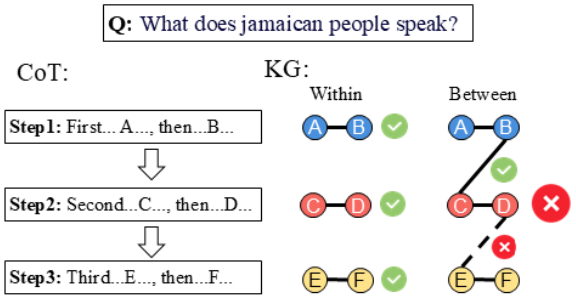


Figure 1: A broken reasoning chain has correct steps but wrong logical relationships between steps.

foundation for the model's decision-making process. However, the internal working mechanism of the LLM relies on statistical learning and probability prediction (Ye et al., 2023), so the LLM sometimes generates plausible text, known as "hallucination" (Ji et al., 2023). To address hallucination, subsequent works introduce the KG. These methods typically use semantic correlation technology to retrieve relevant information from the KG for each step in the CoT, then verify and correct the steps and guide the subsequent reasoning (Luo et al., 2024b; Wang et al., 2023b; Zhao et al., 2024). The logical relationships between steps in CoTs sometimes require the topological structure between multiple triplets in KG to verify. The correctness of the long path CoT requires verification of complex paths in KG. Both of the above situations cannot be solved by a single triplet and require the matching of KG and CoT topology structures. Figure 1 presents a broken reasoning chain: each step in CoT is correct since each corresponds to a triplet A-B/C-D/E-F in the KG, while the whole chain is incorrect since the logical relationships between steps do not exist. This logical structure inconsistency will cause broken reasoning chains and wrong reasoning directions, affecting the coherence and accuracy of the reasoning.

To address these shortcomings, we propose the **DCMKC** (Dual Consistency Matching for KG and CoT) method. This method converts the CoT into a graph based on the reasoning steps and introduces super nodes into the KG. This converts them into graphs of the same granularity for accurate matching. In each iteration, DCMKC matches the KG reasoning chains with CoTs based on semantic similarity and judges the structural consistency between them. Then it modifies CoTs using the matched chains. After iterations, the CoTs and KG reasoning chains reach high semantic and structural consistency, which is theoretically and experimentally demonstrated by kernel and spectral methods. The two kinds of chains are then used to generate the final answers.

In summary, our contributions are as follows:

- To achieve the semantic and structural consistency between CoT and KG, we innovatively propose to convert them into granularity-aligned graphs. This transformation turns multi-hop reasoning and KG matching into iterative matching and modification of two graphs, leading to the DCMKC method.
- We present a matching method based on graph structures. We convert the CoT into a graph and introduce super nodes into the KG. By computing the weighted maximum matching and checking the structural consistency, the two correspond structurally and semantically.
- Our method performs the best on datasets, especially on multi-answer questions.

2 Related Works

2.1 LLM Reasoning Centered on the KG

Early work uses KG-enhanced LLMs as the core technique. It walks on the KG with LLMs and constructs reasoning steps with high accuracy and interpretability (Yang et al., 2024b; Yasunaga et al., 2021). Some works follow previous graph algorithms to construct reasoning paths. TRACE (Fang et al., 2024) uses an auto-regressive reasoning path constructor to build paths from the KG. GNN-RAG (Mavromatis and Karypis, 2024) first retrieves candidate answers to the query on the KG and then uses Graph Neural Networks (GNNs) to extract the reasoning paths. Other works use the LLM to plan and make decisions on the walking process. EffiQA (Dong et al., 2024) uses an LLM to decompose the problem and generate exploration instructions. KG-RAG (Sanmartin, 2024) employs the "Chain of Exploration" (CoE) to search the KG following the exploration plan from the LLM. GraphReader (Li et al., 2024) uses an LLM to explore the KG, continuously updating a notebook to record relevant information. However, the KG contains multi-layer entities and relationships, forming a complex structure. The structure makes it difficult for the LLMs to identify and use the knowledge and logical relationships in KGs (Pan et al., 2023).

To better obtain the key information, subsequent work uses the CoT to locate information and offer

guidance for graph reasoning. RoK (Wang et al., 2024) uses CoTs to delineate the range of answers and then selects the KG reasoning paths according to the range. Graph-CoT (Jin et al., 2024) uses CoT to make LLMs traverse the KG step by step to find the key information. KEQING (Wang et al., 2023a) first breaks down the query into sub-queries, then aligns them with the pre-collected CoTs to retrieve the candidate entities. However, since the CoT and the KG come from different sources, there may be differences in the description of knowledge. This semantic inconsistency can lead to errors in knowledge matching and locating on the KG.

2.2 LLM Reasoning Centered on the CoT

Early works apply the CoT independently in LLM reasoning. They provide examples containing intermediate steps (Wei et al., 2022) or prompt the LLM to think step by step (Kojima et al., 2022) to make it show the reasoning process. However, the internal working mechanism of the LLM is based on statistical learning and probability prediction (Ye et al., 2023), so the LLM may generate plausible texts, known as the "hallucination" (Ji et al., 2023). The hallucination weakens the reliability of reasoning and may mislead subsequent reasoning.

To address hallucination, subsequent works use KG-enhanced CoT technology. This method typically uses semantic correlation technology to retrieve relevant information from the KG for each step in the CoT, then verifies and corrects the steps and guides the subsequent reasoning. GCR (Luo et al., 2024b) employs the KG to guide the LLM in CoT reasoning and constrains the LLM's generation with the KG. KD-CoT (Wang et al., 2023b) verifies and modifies the generated CoT by interacting with external knowledge. SSC-CoT (Zhao et al., 2024) allows the LLM to more accurately identify key intermediate steps by analyzing similarities between different reasoning paths and retrieving the KG. Validating and fixing the current content of the CoT can make sure that each step is correct, but it can't make sure that the logical relationships between steps are right. As a result, the logical relationships between steps in the generated CoT may not align with the KG's topology. This logical structure inconsistency will cause broken reasoning chains and wrong reasoning direction, affecting the coherence and accuracy of the reasoning.

3 Preliminary

The **bipartite graph** is a graph whose node set can be divided into two disjoint subsets. Each edge in the graph connects nodes from two different node sets. The **weighted bipartite graph** assigns a weight to each edge within the graph. The weight can represent the relationship between nodes from different node sets. Here are some concepts about the weighted bipartite graph. The examples are shown in the lower left corner of Figure 2.

Matching. Select a set of edges in the bipartite graph such that no two edges have shared endpoints. This set is referred to as a matching of the bipartite graph. In a matching, two disjoint sets of nodes represent two different types of entities, and edges represent the connection between them.

Weighted Maximum Matching. The weighted maximum matching is the matching that has the most edges and the largest weight sum. It represents the combination of nodes with the greatest overall matching degree between two node sets.

Perfect Matching. In a bipartite graph, if there is a matching where every node in one set is connected to a node in the other, then this matching is called the perfect matching. A perfect matching ensures that each node in the smaller set is covered.

Alternating Path. A path that starts with an unmatched edge and continues with an alternating sequence of unmatched edges and matched edges.

Augmenting Path. We refer to an alternating path as an augmenting path when both ends are non-matching edges. Since there is one more non-matching edge in an augmenting path, the current matching can be expanded by toggling the edges.

4 Method

4.1 Iteration Framework

For a question, we convert the CoTs related to it into a graph and introduce super nodes into the KG to achieve the granularity alignment between the graphs. The core of our method is to iteratively match and correct the two graphs at semantic and structural levels and finally get the accurate CoTs and KG reasoning chains. These chains are then fed into the LLM to get the final answer. The overall framework is shown in Figure 2.

Specifically, we first directly generate a CoT "answer" C_1 with the given question q by the LLM, and take this "answer" as the start for the iteration. The accuracy of "answer" C_1 is challenging to reach a high level. However, we only need to

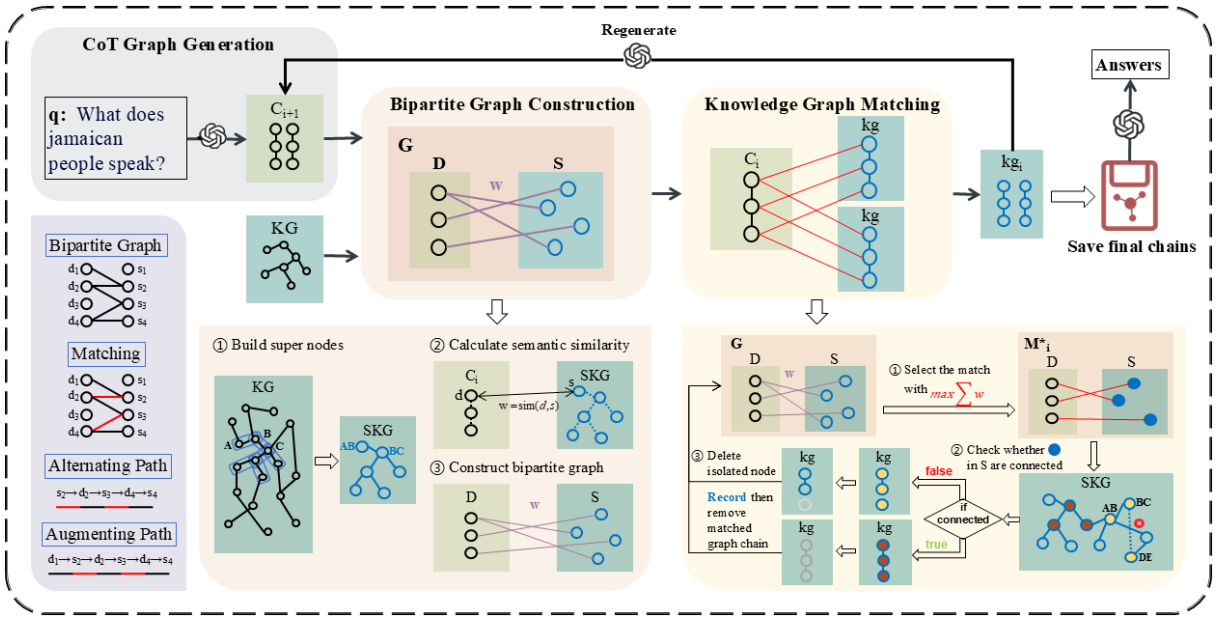


Figure 2: The overview of DCMKC with three key steps. (a) **CoT Graph Generation** makes the subgraph of the KG "correct" the CoT graph. (b) **Bipartite Graph Construction** establishes the semantic connection between the CoT graph and the KG. (c) **KG Matching** makes the CoT "identify" the KG subgraph. The purple part is an example illustrating the concepts in Section 3.

focus on the correlation captured by it when matching and extracting reasoning chains kg_1 (Gao et al., 2023; Jiang et al., 2023). After that, it goes into the iterative process. Each iteration round contains three key steps: **CoT Graph Generation**, **Bipartite Graph Construction**, and **KG Matching**.

CoT Graph Generation. To make the subgraph of the KG "correct" the CoT graph, we regenerate the CoTs based on the KG reasoning chains kg_{i-1} obtained in the previous step and the given question q, and transform them into a new CoT graph CG_i. That is, $(kg_{i-1}, q) \rightarrow CG_i$. The number and content of the CoTs will change according to kg_{i-1} to build a more accurate CoT graph.

Bipartite Graph Construction. To establish the semantic connection between the modified CoT graph CG_i and the KG, we connect the nodes in the two graphs according to semantic similarity to construct a bipartite graph G_i. That is, $(CG_i, KG) \rightarrow G_i$. In G_i, the edge weights represent the semantic similarity between endpoints, used for subsequent KG matching.

KG Matching. To make the CoT "identify" the KG subgraph, we perform multiple matches on the constructed bipartite graph G_i. In each matching, we find the weighted maximum matching M_i^* of G_i according to the semantic similarity and check the structural consistency of the CoT nodes and the KG nodes to extract a reasonable KG reasoning chain. By matching multiple times, we record as many KG reasoning chains as possible and obtain the new set kg_i of them. That is, $G_i \rightarrow M_i^* \rightarrow kg_i$. This new set of KG reasoning chains kg_i will provide the basis for the next iteration.

The iterative process continues until the matched KG reasoning chains no longer change greatly or the number of iterations achieves the preset limit. The process is shown in Algorithm 1. We obtain the CoTs C' with high accuracy and the KG reasoning chains KG' with strong correlation, which can support the subsequent answer generation.

4.2 CoT Graph Generation

To update the CoT graph with the more accurate KG reasoning chains in step $i-1$, in step i, we generate CoTs based on the KG reasoning chains kg_{i-1} and the question q. The prompt template is shown in Appendix A. Since there may be multiple KG reasoning chains, we generate as many CoTs as possible with all of them to get a set of CoTs $C = \{c_1, c_2, c_3, ...\}$.

$$C = \text{LLM}\left(q, \text{kg}_{i-1}\right) = (c_1, \ldots, c_n) = \left((d_{11}, \ldots, d_{1m_1}), \ldots, (d_{n1}, \ldots, d_{nm_n})\right) \quad (1)$$

Where n represents the number of CoTs, and d_{ij} represents the j step of CoT c_i. After that, each

Algorithm 1 Iterative Refinement of CoTs and KG Reasoning Chains

Input: question q, Knowledge Graph KG, max iterations n
Output: final KG reasoning chains KG', final CoTs C'

current iteration $i \leftarrow 0$
graph reasoning chains $kg_0 \leftarrow \emptyset$
while $i \leq n$ **do**
  $i \leftarrow i + 1$
  if $kg_{i-1} = \emptyset$ **then** { Section 4.2.}
    $CG_i \leftarrow LLM(q)$
  else
    $CG_i \leftarrow LLM(q, kg_{i-1})$
  end if
  $G_i \leftarrow Build(CG_i, KG)$ {G_i: bipartite graph. Section 4.3.}
  $M_i^* \leftarrow Match(G_i)$ {M_i: weighted maximum matching. Section 4.4.}
  $kg_i \leftarrow Extract(M_i^*)$ {Details are in Algorithm B}
  if kg_i is similar to kg_{i-1} **then**
    final KG reasoning chains $KG' \leftarrow kg_i$
    final CoTs $C' \leftarrow CG_i$
    break
  end if
end while
return KG' and C' =0

step is converted into a node, and the nodes are connected according to the order of the chains to form the CoT graph $CG_i = \{CG_{i1} \cup \ldots \cup CG_{in}\}$. After that, we use the pre-trained Sentence-BERT model (Reimers, 2019) to generate semantic vector embeddings of nodes. This vector embedding is then used to retrieve and obtain relevant information by calculating semantic similarity.

4.3 Bipartite Graph Construction

To establish the semantic relation between the modified CoT graph of step i and the KG, we construct a bipartite graph. The two node sets of the bipartite graph are the CoT graph nodes and the KG nodes. Each edge in the graph represents the semantic similarity between two endpoints.

Since the CoT graph nodes are reasoning steps and the KG nodes are entities, they cannot be directly matched because of the different granularity. To better match the two graphs, we consider each triplet in the KG as a node, called a super node. For a triplet (A, R, B), a super node s is:

$$s = \{h : A, r : R, t : B\} \tag{2}$$

A super node s contains three parts: the head entity A, the relationship R, and the tail entity B. If the head and tail entities in one super node have an intersection with another, we connect the two nodes. As shown in Figure 3, since the tail entity of s_1 is the same as the head entity of s_2, the two super nodes can be connected in the super KG SKG.

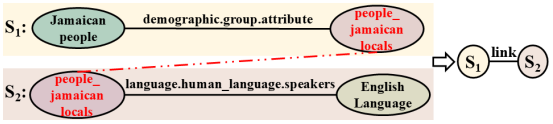


Figure 3: An example for connecting super nodes.

After the super KG SKG is constructed, for a super node s, its internal information (h, r, t) is first concatenated into a string s'. Then the strings are input into the Sentence-BERT model to obtain the vector embeddings. After obtaining the vector embeddings of the super KG SKG and the CoT graph CG_i nodes, we calculate the semantic similarity between the two types of nodes. Since the nodes in the super KG SKG are much more than those in the CoT graph CG_i, for each node in the CoT graph CG_i, we select the top k similar nodes in the super KG to connect to it. The weight of edges is the semantic similarity between nodes. A weighted bipartite graph $G_i(D \cup S, E)$ is:

$$\begin{aligned} D &= \{d_{11}, d_{12}, \ldots, d_{nm_n}\} \\ S &= \{s_1, s_2, \ldots, s_{|S|}\} \\ E &= \{e_{ij} \mid e_{ij} = \{d_i, s_j\}, d_i \in D, s_j \in S\} \end{aligned} \tag{3}$$

The left node set of the bipartite graph G_i is the node set D in the CoT graph CG_i, and the right node set is the selected super node set S in the super KG SKG. For any two nodes $d \in D$ and $s \in S$, if the similarity between them is positive, an edge e is connected between them. The weight of the edge e is their semantic similarity $sim(d, s)$. Therefore, the CoT graph CG_i and the subgraph of the super KG establish the semantic relationships between nodes, which are used for subsequent matching.

4.4 KG Matching

To prune the super KG subgraph using CoTs, we compute weighted maximum matching based on the semantic bipartite graph. After getting the weighted maximum matching based on semantic similarity, we extract reasonable KG reasoning chains according to the structural correspondence of the node sets. These chains have a high degree of structural and semantic consistency with CoTs.

Specifically, for a bipartite graph $G_i(D \cup S, E)$, the maximum matching M_i^* means that the weight sum of the edges in the matching is larger than or equal to any other matching. For $\forall M \prec G$

$$W_{M_i^*} = \sum_{e_{ij} \in M_i^*} w_{ij} \geq W_M \tag{4}$$

Since the weight of edges in the bipartite graph $G_i(D \cup S, E)$ is the semantic similarity between two types of nodes, the maximum matching M_i^* represents the combination of the super KG nodes and the CoT graph nodes with the highest semantic consistency. To obtain this matching, we refer to the KM algorithm (Kuhn, 1955; Munkres, 1957) and assign a label $wlbl$ to each node. The details of the process to obtain the maximum weighted matching M_i^* are in Appendix C. The matching M_i^* is not obtained by matching single nodes. In fact, it's obtained by matching two sets of nodes as a whole. These sets are the CoT nodes and the super KG nodes with the highest semantic similarity.

After that, we extract the KG reasoning chains kg_i based on the matching M_i^*. We first check the connectivity between nodes belonging to S in M_i^*. The KG reasoning chains must be complete, so if there is an independent node, it should be deleted to obtain the maximum weighted matching again. Since there are multiple KG reasoning chains, we should remove all nodes in the chain from S after we successfully extract a KG reasoning chain and match again until no new chain can be found. The process is shown in Appendix B.

In this way, the KG reasoning chains match the CoTs with the highest weight, indicating the highest semantic consistency. The logical relationship and topological structure of the two are also consistent, achieving a high structural similarity. The KG reasoning chains will help to supplement and modify the CoT graph in the next iteration.

4.5 Answer Generation

In the generation stage, the prompts submitted to the LLM are divided into hard prompts and soft prompts (Hu et al., 2024). The hard prompts are plain texts, and the soft prompts are textualized representations of the graph topology (He et al., 2025). By combining them, text information and graph topological information can both be preserved in the prompts to reduce the knowledge loss. The prompt templates are shown in Appendix A.

Hard prompt refers to combining the CoTs C', the KG reasoning chains KG' and the question q with a prompt template to form a plain text prompt.

$$\mathbf{p}_{\text{hard}} = \left[q; C'; KG'\right] \tag{5}$$

To keep the topological information of KG reasoning chains KG', we textualize the graph structure to get the soft prompt. The hard and soft prompts are combined and submitted to the LLM to generate the final answer. By combining them, text and graph topological information can both be preserved to improve the LLM reasoning.

5 Experiments

5.1 Experiment Setup

Datasets. Following previous works (Luo et al., 2024a; Wang et al., 2023b; Luo et al., 2024b), we conduct experiments on two datasets, WebQuestionSP (WebQSP) (Yih et al., 2016) and Complex WebQuestions (CWQ) (Talmor and Berant, 2018). Details of datasets are in Appendix D.

Baselines. We compared DCMKC with 12 baselines grouped into 4 categories: (1) LLM only, including Qwen2-7B (Yang et al., 2024a), Llama-2-7B (Touvron et al., 2023), Llama-3.1-8B (Dubey et al., 2024). (2) Retrieving+LLM, including BM25 (Robertson et al., 2009), LaBSE (Feng et al., 2022), E5-Base (Wang et al., 2022). (3) KG+LLM, including G-Retriever (He et al., 2024), GRAG (Hu et al., 2024), EffiQA (Dong et al., 2024), RoG (Luo et al., 2024a). (4) CoT and KG+LLM. including ToG (Sun et al., 2024), KD-CoT (Wang et al., 2023b). Details of baselines are in Appendix E.

Evaluation Metrics. We use Hit@1 and the F1 score as evaluation metrics. Hit@1 checks if the ground truth exists in the generated answers. The F1 score is a harmonic average of accuracy and recall, providing a metric that balances false positives and false negatives.

Implementations. We choose Llama-3.1-8B-Instruct and Llama-2-7B-Chat as the backbone LLMs for our method. The parameters of them are frozen. For the baselines that only use the LLM, we use a zero-shot prompt to ask the model to answer the questions. We select the number of matching nodes $k = 8$ for the bipartite graph construction, and the maximum number n of iterations is set to 3. All experiments are performed on an Intel(R) Core(TM) i5-9300H CPU @ 2.40 GHz. The results of different hyperparameter settings are in Appendix G.

5.2 Main Results

We compare our DCMKC method to other baselines on the datasets to evaluate the model's reasoning ability. As Table 1 shows, DCMKC performs

Category	Method	WebQSP		CWQ	
		F1 Score	Hit@1	F1 Score	Hit@1
LLM only	Qwen2-7B (Yang et al., 2024a)	0.3550	0.5080	0.2160	0.2530
	Llama-2-7B (Touvron et al., 2023)	0.3650	0.5640	0.2140	0.2840
	Llama-3.1-8B (Dubey et al., 2024)	0.3480	0.5550	0.2240	0.2810
Retrieving+LLM	BM25 (Llama2) (Robertson et al., 2009)	0.4284	0.5829	0.2324	0.2880
	LaBSE (Llama2) (Feng et al., 2022)	0.4665	0.6114	0.2532	0.3022
	E5-Base (Llama2) (Wang et al., 2022)	0.4865	0.6398	0.2641	0.3162
KG+LLM	G-Retriever (Llama2) (He et al., 2024)	0.4674	0.6808	0.3396	0.4721
	GRAG (Llama2) (Hu et al., 2024)	0.5022	0.7236	0.3649	0.5018
	SubgraphRAG (Llama3) (Li et al., 2025)	0.7057	0.8661	0.4716	0.5698
CoT and KG+LLM	ToG (ChatGPT) (Sun et al., 2024)	0.7232	0.7513	0.5696	0.5759
	KD-CoT (Llama3) (Wang et al., 2023b)	0.5250	0.6860	-	0.5570
	RoG (Llama2) (Luo et al., 2024a)	0.7080	0.8570	0.5620	0.6260
Our method	DCMKC (Llama3)	<u>0.7480</u>	<u>0.8695</u>	**0.6110**	<u>0.7358</u>
	DCMKC (Llama2)	**0.7503**	**0.8698**	<u>0.6002</u>	**0.7360**

Table 1: Model performance on two datasets comparing four categories of methods. The best results are **bolded**, and the second best results are <u>underlined</u>.

best on both datasets. The F1 score on WebQSP and CWQ is 5.6% and 8.7% above the best baseline, and Hit@1 is 1.5% and 5.8% above. The results show that DCMKC can effectively enhance the reasoning ability of the LLM.

It is found that when all the information in the KG is submitted to the LLM, the model gives suboptimal performance. For Llama2, on the WebQSP dataset, only adding a retrieving method can improve F1 by up to 33.3% and Hit@1 by up to 13.4%. It indicates that pruning is essential for graph reasoning to reduce the influence of redundant information. LLMs can perform better using techniques based on KGs or CoTs. On the CWQ dataset, ROG improves F1 of the corresponding LLM by 162.6%, and EffiQA improves Hit@1 of the corresponding LLM by 147.3%. It shows that KGs and CoTs play a crucial role in optimizing LLM's ability to understand complex questions, accurately extract, and effectively apply relevant knowledge.

We also observe that larger LLMs perform worse in graph-related tasks than smaller LLMs. For the performance of Llama2 and Llama3, Llama2 has a higher F1 and Hit@1 on the WebQSP dataset, by 4.9% and 1.6%. In the CWQ dataset, the situation is similar. This suggests that increasing the parameters does not inherently enhance the graph reasoning ability of LLMs.

5.3 Ablation Study

We conduct a series of evaluations of DCMKC to see which component plays a key role, including removing iteration, replacing KG matching with retrieving, and removing both parts. As shown in Table 2, the performance all decreases. This indicates that every component is indispensable. Among them, removing iteration drops model performance more than removing KG matching. On the WebQSP dataset, F1 of the former decreased by 6.4%, and Hit@1 decreased by 1.4% more than that of the latter. The situation is similar on CWQ. This suggests that the iteration framework plays a more central role in reasoning, allowing the model to optimize its answers continuously.

Method	WebQSP		CWQ	
	F1 Score	Hit@1	F1 Score	Hit@1
DCMKC	**0.7480**	**0.8695**	**0.6110**	**0.7358**
w/o Iteration	0.6572	0.8255	0.5748	0.7045
w/o KG matching	0.7095	0.8384	0.5842	0.7134
w/o both	0.5637	0.8034	0.5423	0.6721

Table 2: Performances of three model variables.

5.4 Analytical Experiments

Bi-Directional Interaction Analysis. Figure 4 shows the optimization of the KG reasoning chains and the CoTs in the iteration, including the relevance between the KG reasoning chains and the question, the average proportion of answers contained in KG reasoning chains, and the F1 score and Hit@1 of the answers obtained from the CoTs only. Relevance here refers to the average semantic similarity between the KG reasoning chain and the question, which is calculated based on the vector embeddings provided by Sentence-BERT. As the

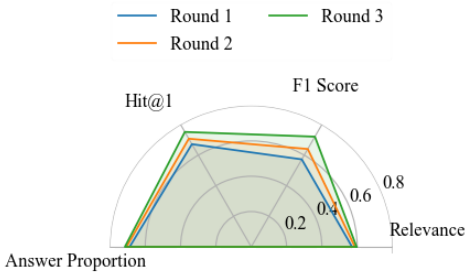


Figure 4: The optimization of the KG reasoning chains and the CoTs in the iteration.

number of iterations increases, the relevance gradually improves, indicating that the KG location is more accurate. The average proportion of answers contained increases, indicating that the quality of KG reasoning chains increases with continuous iterations. The answers obtained from the CoTs also move closer to the ground truth. The change of the CoTs reflects the effectiveness of the KG reasoning chains in modifying the CoTs, and the adjustment of the KG reasoning chains reflects the effectiveness of the CoTs in locating key information. This iterative interaction and bidirectional modification effectively enhance the quality of the two kinds of chains to improve the model's overall performance.

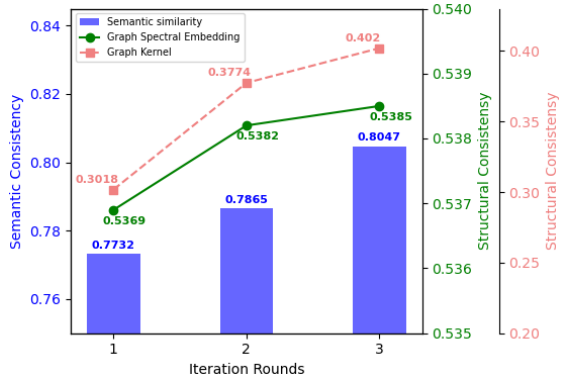
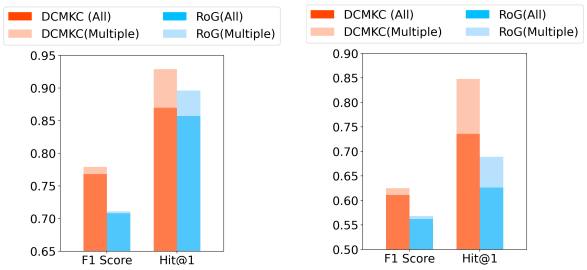


Figure 5: Semantic and structural consistency.

Semantic and Structural Consistency Analysis. Figure 5 shows the changes in the semantic and structural levels of the CoTs and the KG reasoning chains. Their semantic similarity gradually improves in the iteration, indicating that they achieve high semantic consistency, which reaches 0.8047. Our experiments revealed a consistent improvement in structural alignment between CoTs and KG reasoning chains, assessed via Graphlet Kernel (local substructures) and Graph Spectral Embedding (global topology) (see Appendix I). The normalized consistency scores increased from 0.3018 to 0.4020 (Graphlet Kernel) and 0.5369 to 0.5385 (Graph Spectral Embedding). Marginal gains in consistency diminished across iterations—25.05% and 6.51% for Graphlet Kernel (Iterations $1 \rightarrow 2$ and $2 \rightarrow 3$), and 0.24% and 0.056% for Graph Spectral Embedding—adhering to the law of diminishing marginal utility and indicating convergence toward stability. These results demonstrate that CoTs and KG reasoning chains mutually refine each other through iterative alignment.



(a) WebQSP dataset. (b) CWQ dataset.

Figure 6: Performances on multi-answer questions.

Multi-Answer Questions Analysis. We select multi-answer questions in WebQSP and CWQ datasets and conduct experiments on RoG and our method DCMKC. As shown in Figure 6, our method shows a higher improvement. It indicates that our method is more advantageous in multi-answer questions and highlights the efficacy of multiple graph matching and the iteration framework.

Time Cost Analysis. The average time cost per question of some baselines and our method is shown in Table 3. It shows that our time cost is comparable to other CoT and KG+LLM methods. We also calculate the time complexity of some baselines and our method and the results are in Appendix H.

Category	Method	Avg. Time
KG+LLM	G-Retriever	3.7
	GRAG	3.1
CoT and KG+LLM	KD-CoT	6.2
	ToG	11.5
Our method	DCMKC	7.1

Table 3: Average time cost (Second).

Parameter Quantity Analysis. Considering that the parameter quantity of base LLMs is relatively small, we conduct experiments on Llama-3.1-70B. As shown in Table 4, our method can still make improvements.

Method	WebQSP	CWQ
LLM Only	0.7310	0.4610
DCMKC(Our method)	0.8826	0.7875

Table 4: Performance on Llama-3.1-70B (Hit@1).

5.5 Case study

We choose some cases to show how our method fixes initial errors in CoTs and locates the KG more accurately. The details are shown in Appendix F.

6 Conclusion

In this paper, we introduce a model based on semantic and structural consistency matching called DCMKC. The method converts the CoTs into a graph and introduces super nodes into the KG to make the two correspond structurally and semantically, which is theoretically and experimentally demonstrated by kernel and spectral methods. In the reasoning process, the method makes the CoTs and the KG reasoning chains work together through the framework of iterative interaction and bidirectional modification. Experiments show that DCMKC outperforms the baselines on multiple datasets, especially on multi-answer questions, with up to 5.1% improvement.

7 Limitations

Although DCMKC achieves strong performance in the QA task over all baselines, there are still some limitations to our method. (1) The backbone LLM we used has frozen parameters. Future methods might need to consider fine-tuning the LLM to obtain better results. (2) The prompt templates in this study still rely on manual design, influenced by previous research that has been shown to be effective. However, it would be interesting to explore the development of automated methods for constructing prompt templates. In addition, our methods may be used for harmful data, causing issues such as privacy disclosure.

Acknowledgments

This paper is partially supported by National Natural Science Foundation of China (NSFC Grant No. 62576353 and No. 62376281) and National Natural Science Foundation of China Distinguished Young Scholar Project (Grant No. 62125604).

References

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*, pages 1247–1250.

Zixuan Dong, Baoyun Peng, Yufei Wang, Jia Fu, Xiaodong Wang, Yongxue Shan, and Xin Zhou. 2024. Effiqa: Efficient question-answering with strategic multi-model collaboration on knowledge graphs. *arXiv preprint arXiv:2406.01238*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv 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 preprint arXiv:2406.11460*.

Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. 2022. Language-agnostic bert sentence embedding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 878–891.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023. Precise zero-shot dense retrieval without relevance labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1762–1777.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2025. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. *Advances in Neural Information Processing Systems*, 37:132876–132907.

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. *arXiv preprint arXiv:2402.07630*.

Yuntong Hu, Zhihan Lei, Zheng Zhang, Bo Pan, Chen Ling, and Liang Zhao. 2024. Grag: Graph retrieval-augmented generation. *arXiv preprint arXiv:2405.16506*.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. In *Proceedings of the 2023*

Conference on Empirical Methods in Natural Language Processing, pages 7969–7992.

Bowen Jin, Chulin Xie, Jiawei Zhang, Kashob Kumar Roy, Yu Zhang, Zheng Li, Ruirui Li, Xianfeng Tang, Suhang Wang, Yu Meng, et al. 2024. Graph chain-of-thought: Augmenting large language models by reasoning on graphs. *arXiv preprint arXiv:2404.07103*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Harold W Kuhn. 1955. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97.

Mufei Li, Siqi Miao, and Pan Li. 2025. Simple is effective: The roles of graphs and large language models in knowledge-graph-based retrieval-augmented generation. *Preprint*, arXiv:2410.20724.

Shilong Li, Yancheng He, Hangyu Guo, Xingyuan Bu, Ge Bai, Jie Liu, Jiaheng Liu, Xingwei Qu, Yangguang Li, Wanli Ouyang, et al. 2024. Graphreader: Building graph-based agent to enhance long-context abilities of large language models. *arXiv preprint arXiv:2406.14550*.

Mary M Lucas, Justin Yang, Jon K Pomeroy, and Christopher C Yang. 2024. Reasoning with large language models for medical question answering. *Journal of the American Medical Informatics Association*, 31(9):1964–1975.

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

Linhao Luo, Zicheng Zhao, Chen Gong, Gholamreza Haffari, and Shirui Pan. 2024b. Graph-constrained reasoning: Faithful reasoning on knowledge graphs with large language models. *arXiv preprint arXiv:2410.13080*.

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

James Munkres. 1957. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38.

Jeff Z Pan, Simon Razniewski, Jan-Christoph Kalo, Sneha Singhania, Jiaoyan Chen, Stefan Dietze, Hajira Jabeen, Janna Omeliyanenko, Wen Zhang, Matteo Lissandrini, et al. 2023. Large language models and knowledge graphs: Opportunities and challenges. *arXiv preprint arXiv:2308.06374*.

Shivam Panwar, Anukriti Bansal, and Farhana Zareen. 2023. Comparative analysis of large language models for question answering from financial documents. In *International Conference on Communication and Intelligent Systems*, pages 297–308. Springer.

N Reimers. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Diego Sanmartin. 2024. Kg-rag: Bridging the gap between knowledge and creativity. *arXiv preprint arXiv:2405.12035*.

Pasi Shailendra, Rudra Chandra Ghosh, Rajdeep Kumar, and Nitin Sharma. 2024. Survey of large language models for answering questions across various fields. In *2024 10th International Conference on Advanced Computing and Communication Systems (ICACCS)*, volume 1, pages 520–527. IEEE.

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

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.

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*.

Chaojie Wang, Yishi Xu, Zhong Peng, Chenxi Zhang, Bo Chen, Xinrun Wang, Lei Feng, and Bo An. 2023a. keqing: knowledge-based question answering is a nature chain-of-thought mentor of llm. *arXiv preprint arXiv:2401.00426*.

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

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Yuqi Wang, Boran Jiang, Yi Luo, Dawei He, Peng Cheng, and Liangcai Gao. 2024. Reasoning on efficient knowledge paths: Knowledge graph guides large language model for domain question answering. *arXiv preprint arXiv:2404.10384*.

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.

Zhentao Xu, Mark Jerome Cruz, Matthew Guevara, Tie Wang, Manasi Deshpande, Xiaofeng Wang, and Zheng Li. 2024. Retrieval-augmented generation with knowledge graphs for customer service question answering. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2905–2909.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. 2024a. Qwen2 technical report. *Preprint*, arXiv:2407.10671.

Rui Yang, Haoran Liu, Edison Marrese-Taylor, Qingcheng Zeng, Yuhe Ke, Wanxin Li, Lechao Cheng, Qingyu Chen, James Caverlee, Yutaka Matsuo, et al. 2024b. Kg-rank: Enhancing large language models for medical qa with knowledge graphs and ranking techniques. *CoRR*.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. Qa-gnn: Reasoning with language models and knowledge graphs for question answering. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.

Hongbin Ye, Tong Liu, Aijia Zhang, Wei Hua, and Weiqiang Jia. 2023. Cognitive mirage: A review of hallucinations in large language models. *arXiv preprint arXiv:2309.06794*.

Wen-tau Yih, Matthew Richardson, Christopher 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.

Zilong Zhao, Yao Rong, Dongyang Guo, Emek Gözlüklü, Emir Gülboy, and Enkelejda Kasneci. 2024. Stepwise self-consistent mathematical reasoning with large language models. *arXiv preprint arXiv:2402.17786*.

A Prompt Templates

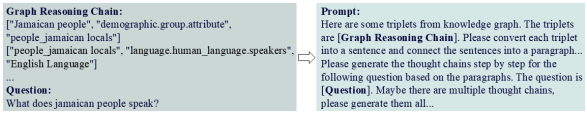


Figure 7: Prompt template.

For the prompt to regenerate the CoTs, the triplets in the KG reasoning chains kg_{i-1} are first converted into sentences and then combined with the question q according to the preset prompt template, as shown in Figure 7. The prompt is then submitted to the LLM to generate new CoTs.

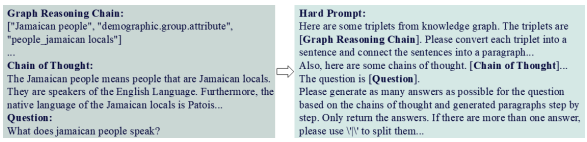


Figure 8: Hard prompt template.

For the hard prompt, the triplets in the KG reasoning chains KG' are first converted into sentences and then combined with the question q and the CoTs C' according to the preset prompt template, as shown in Figure 8.

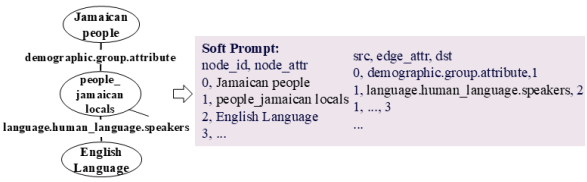


Figure 9: Soft prompt template.

For the soft prompt, we textualize the graph structure, as shown in Figure 9.

B Algorithm of KG matching

For KG matching, the whole process is shown in Algorithm 2. It shows how our method DCMKC obtains the weighted maximum matching and extracts KG reasoning chains.

C The Maximum Weighted Matching

To obtain the maximum weighted matching M_i^*, we refer to the KM algorithm and assign a label $wlbl$ to each node, transferring the weight of the

Algorithm 2 Obtain the Weighted Maximum Matching and Extract KG Reasoning Chains

Input: bipartite graph $G_i(D \cup S, E)$, knowledge graph KG
Output: KG reasoning chains kg_i

KG reasoning chains $kg_i \leftarrow \emptyset$
while $len(S) \geq len(D)$ **do**
  $M_i^* \leftarrow Match(G)$ {M_i: weighted maximum matching.}
  $chain \leftarrow Reconnect(M_i^*, KG)$ {$chain$: matched reasoning chain.}
  if $chain$ is complete **then**
    $kg_i \leftarrow kg_i + chain$
    $G \leftarrow G - chain$
  end if
  if an isolated node s in $chain$ **then**
    $G \leftarrow G - s$
  end if
end while
return kg_i =0

edges to the nodes. For $\forall d_i \in D, \forall s_j \in S$, the label should satisfy

$$wlbl\left[d_i\right] + wlbl\left[s_j\right] \geq w_{ij} \tag{6}$$

The label is used to transform the problem of finding a weighted maximum matching into the problem of finding a perfect matching without considering the edge weights. First, we introduce the concept of the equal subgraph. An edge is equal if the label sum of its two endpoints equals the weight of it. All equal edges with their endpoints form an equal subgraph. The label sum of nodes in an equal subgraph is equal to the weight sum of the edges.

Consider the following theorem.

Theorem 1. *If the maximum matching of some equal subgraph in a bipartite graph is a complete matching of the bipartite graph, then this complete matching is a maximum weighted matching of the bipartite graph.*

Proof. If there exists a perfect matching of the bipartite graph that does not belong to the equal subgraph, considering the definition of the label, the weight sum of the edges in this matching is less than the sum of all the labels. However, the weight sum of the edges in a perfect matching belonging to the equal subgraph is equal to the sum of the labels of the nodes, so this perfect matching is the maximum weighted matching of the bipartite graph. □

So the maximum weighted matching $M*_i$ is the maximum matching including all nodes in D (Perfect matching) of the equal subgraph. To more easily adjust the labels, we set the label of the node d_i to the largest weight among edges associated with it, and set the label of node s_j to 0. For $\forall d_i \in D$ and $\forall s_j \in S$

$$\begin{aligned} wlbl\left[d_i\right] &= \max_{s \in S}\left(w_{ij}\right) \\ wlbl\left[s_j\right] &= 0 \end{aligned} \tag{7}$$

After the initial labels are set, we find an augmenting path for a node d_i in the equal subgraph. If an augmenting path is found, the matching can be expanded by toggling the matching edges and the non-matching edges. If the augmenting path is not found, we need to adjust the label to expand the equal subgraph. We reduce the label of the nodes belonging to D in the found alternating path by a value h, and add a value h to the node belonging to S. Because the modified labels should still satisfy Equation 6, we go through all the edges e_{ij} in the previously selected augmenting path tree and update h to the smallest $wlbl\left[s_i\right] + wlbl\left[d_j\right] - w_{ij}$. There will be four results: (1) An edge with both ends in the alternating path has no change in the label sum. It still belongs to the equal subgraph. (2) An edge with both ends not in the alternating path has no change in the label sum. It still doesn't belong to the equal subgraph. (3) For an edge, if the node belonging to D is not in the alternating path, and the node belonging to S is in, the label sum will increase. It still doesn't belong to the equal subgraph. (4) For an edge, if the node belonging to S is not in the alternating path, and the node belonging to D is in, the label sum will decrease. It didn't belong to the equal subgraph before, but now it may belong to it.

After expanding the equal subgraph, the augmenting path is found again with this node. The algorithm terminates when every node belonging to D finds the augmenting path and toggles matching. The matching now is the maximum matching, including all nodes in D of the equal subgraph.

D Datasets

Dataset	#Graphs	Avg. #Nodes
WebQSP	4737	1371
CWQ	34689	1306

Table 5: Statistics of datasets.

Table 5 and Table 6 show the statistics of the datasets. WebQuestionSP (WebQSP) (Yih et al., 2016) is a large multi-hop KGQA dataset contain-

Dataset	#Answer=1	#Answer>1
WebQSP	51.2%	48.8%
CWQ	70.6%	29.4%

Table 6: Statistics of answer numbers.

ing 4737 questions, of which 48.8% are multi-answer questions. Complex WebQuestions (CWQ) (Talmor and Berant, 2018) is a large dataset of complex multi-hop questions with 34,689 questions, of which 29.4% are multi-answer questions. The two datasets test the model's ability to understand and answer questions with multiple facts and reasoning steps. The KG for both datasets is Freebase (Bollacker et al., 2008).

E Baselines

The baselines can be divided into four categories: (1) LLM only, (2) Retrieving+LLM, (3) KG+LLM, (4) CoT and KG+LLM.

(1) LLM-only methods only use LLMs for reasoning without other enhancement methods.

Qwen2-7B (Yang et al., 2024a) is one of a series of LLMs developed by the Alibaba Cloud Tongyi Qianwen team, with a parameter size of 7 billion.

Llama-2-7B (Touvron et al., 2023) is one of the Llama 2 series of LLMs developed by Meta AI, with a parameter size of 7 billion.

Llama-3.1-8B (Dubey et al., 2024) is one of the Llama 3 series of LLMs developed by Meta AI, with a parameter size of 8 billion.

(2) Retriever-enhanced LLM methods use retrieval to enhance LLM reasoning.

BM25 (Robertson et al., 2009) is a statistical model that scores documents based on term frequency, inverse document frequency, and document length, using probabilistic principles to estimate the relevance of documents to queries.

LaBSE (Feng et al., 2022) is a BERT-based model that uses a dual encoder framework to learn sentence embeddings across languages.

E5-Base (Wang et al., 2022) adopts a contrastive pre-training strategy using dual encoder architecture to optimize the similarity between related pairs while using in-batch negative samples to distinguish the similarity between unrelated pairs.

(3) KG-enhanced LLM methods use KGs to enhance LLM reasoning.

G-Retriever (He et al., 2024) retrieves the relevant nodes and edges, then constructs the relevant subgraph using the bonus Steiner tree method.

GRAG (Hu et al., 2024) retrieves text subgraphs and performs soft pruning to identify relevant subgraph structures effectively, and proposes a new cue strategy.

SubgraphRAG (Li et al., 2025) generates accurate and explainable answers by efficiently retrieving relevant subgraphs from KGs and leveraging LLMs for reasoning.

(4) CoT and KG-enhanced LLM methods use CoTs and KGs to enhance LLM reasoning.

ToG (Sun et al., 2024) allows LLM to dynamically explore multiple reasoning paths in KGs to form CoTs and make decisions based on them.

KD-CoT (Wang et al., 2023b) retrieves relevant knowledge from KGs to modify CoTs and generate reliable reasoning plans with the LLM.

RoG (Luo et al., 2024a) proposes a planning-search-reasoning framework, which retrieves reasoning paths from KGs to guide LLMs in reasoning.

F Case Study

For the correction of factual errors, as shown in Table 7, there is a factual error ('California') in the CoT (Round 1), and the error is corrected by KG in Round 2 later.

For the correction of logical errors, as we can see in Table 8, Step 1 and Step 2 of CoT (Round 1) are both correct, but there is no logical relationship between them, leading to errors in reasoning. Then we get KG (Round 1) through correlation and learn an important relationship, "government.us_president.vice_president" from it. After that, we regenerate a new CoT based on KG (Round 1) with more accurate logical relationships between steps, which is CoT (Round 2). According to the logical relationship, we get the accurate KG (Round 2) and finally get the correct answer.

At the same time, the KG in the initial round has a wide scope, and with the help of CoT, the KG location is more accurate.

G Hyperparameter Analysis

For the number of matching nodes k, we conduct multiple experiments with different values of it in one iteration. As shown in Figure 10, when $k = 8$, Hit@1 is the highest. Therefore, the number of matching nodes k is set to 8.

For the maximum number n of iterations, we conduct experiments on multiple values of it to

Q	**where is jamarcus russell from?**
A	**Mobile**
CoT (Round 1)	JaMarcus Russell was born in **California**.
KG (Round 1)	[JaMarcus Russell, people.person.place_of_birth, **Mobile**] [JaMarcus Russell, people.person.nationality, United States of America] [JaMarcus Russell, people.person.ethnicity, African American]
CoT (Round 2)	JaMarcus Russell is from **Mobile**.
KG (Round 2)	[JaMarcus Russell, people.person.place_of_birth, **Mobile**]

Table 7: Correction of the initial factual errors.

Q	**who was vice president after kennedy died?**
A	**Lyndon B. Johnson**
CoT (Round 1)	John F. Kennedy was **US President**. The **vice president** plays the role of assistant to the president.
KG (Round 1)	['US President, person.role.in_time, John F. Kennedy', 'Vice president, type.property.schema, US President'] ['Dwight D. Eisenhower, base.kwebbase.kwtopic.has_sentences, He was succeeded by President Kennedy', 'Dwight D. Eisenhower, **government.us_president.vice_president**, Richard Nixon']
CoT (Round 2)	John F. Kennedy was US President. **A president has a vice president.**
KG (Round 2)	['US President, person.role.in_time, John F. Kennedy', 'John F. Kennedy, government.us_president.vice_president, **Lyndon B. Johnson**']
CoT (Round 3)	John F. Kennedy was US President. **Lyndon B. Johnson** was vice president after Kennedy died.
KG (Round 3)	['US President, person.role.in_time, John F. Kennedy', 'John F. Kennedy, government.us_president.vice_president, **Lyndon B. Johnson**']

Table 8: Correction of the initial factual errors.

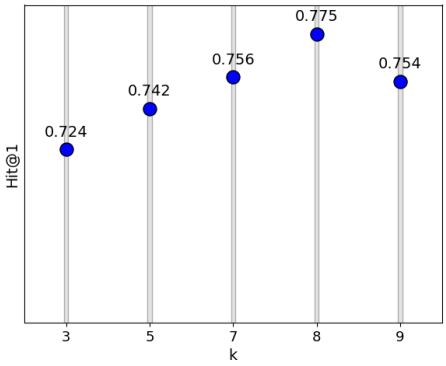


Figure 10: Performances on different values of k.

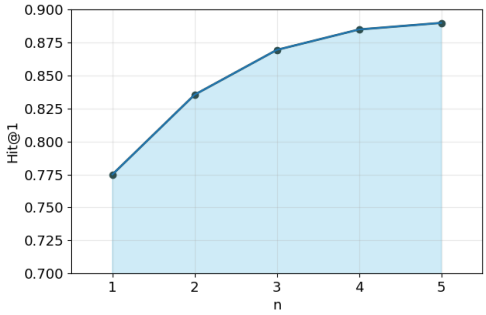


Figure 11: Performances on different values of n.

see how the performance of the model improves. As shown in Figure 11, the performance of the model continues to improve as n increases. However, when $n > 3$, the increase gradually becomes flat. Considering the performance of the model and the time and cost of the experiment, we set the maximum number n of iterations to 3.

H Time Complexity

As Table 9 shows, most methods use a complex framework. It shows that our time complexity is comparable to the baselines.

I Structural Consistency Analysis Methods

I.1 Graphlet Kernel

The Graphlet Kernel is a similarity metric based on subgraph pattern statistics. Its core idea is to measure structural similarity by comparing the frequency distribution of connected subgraphs (called graphlets) of size k (typically 3-5 nodes). The key steps are as follows:

- Graph Decomposition: Enumerate all connected k-node subgraphs (graphlets) $\mathcal{G}_k$ from the input graph G.

Category	Method	Time Complexity	Explanation
KG+LLM	G-Retriever	$O((V+E)\cdot \log k + T^2)$	V: number of KG nodes E: number of KG edges k: number of candidate subgraphs (≤ 5) T: LLM input length
	GRAG	$O(V \cdot d^k + T^2)$	V: number of KG nodes d: number of subgraph nodes k: number of search hops (≤ 2) T: LLM input length
	RoG	$O(K \cdot L + d^l + T^2)$	K: number of reasoning paths (≤ 3) L: reasoning path length (≤ 4) d: number of expanded edges (≤ 100) l: relationship path length (≤ 4) T: LLM input length
CoT and KG+LLM	KD-CoT	$O(k \cdot (N + L^2 + T^2))$	k: number of CoT steps (≤ 4) N: number of candidate paragraphs ($= 100$) L: candidate paragraph length T: LLM input length
	ToG	$O((2N \cdot D + D + 1) \cdot T^2)$	N: beam search width ($= 3$) D: maximum search depth (≤ 3) T: LLM input length
Our method	DCMKC	$O(k \cdot (C + t \cdot d^3 + T^2))$	k: number of iterations (≤ 3) C: number of CoT nodes (≤ 4) t: number of matching d: number of candidate nodes (≤ 8) T: LLM input length

Table 9: Time complexity of baselines and our method.

- Frequency Counting: Count the occurrences of each graphlet $g_i \in \mathcal{G}_k$ in G, forming a frequency vector $f_G = (n_{g_1}, n_{g_2}, \ldots, n_{g_{|\mathcal{G}_k|}})$.
- Normalization and Kernel Calculation: Normalize the frequency vector as $h_G = \frac{f_G}{\|f_G\|_1}$, and compute the similarity between graphs G and G' via the kernel function:

$$K_{\text{graphlet}}(G, G') = h_G^T h_{G'}$$

This method captures global structural similarity by analyzing local subgraph pattern distributions.

I.2 Graph Spectral Embedding

Graph Spectral Embedding maps nodes into a low-dimensional space using spectral decomposition of the graph Laplacian matrix, leveraging spectral properties to characterize graph structures. The key steps are:

- Laplacian Matrix Construction: For a graph G with adjacency matrix $A(G)$ and degree matrix $D(G)$, the Laplacian matrix is defined as $L(G) = D(G) - A(G)$. If two graphs G and G' differ in scale, isolated vertices are added to ensure identical vertex counts, and zero-padding is applied to align the dimensions of $L(G)$ and $L(G')$.
- Eigen Decomposition: Perform eigen decomposition on $L(G)$ and select the top t (≥ 1) eigenvectors $U = [u_1, u_2, \ldots, u_t] \in R^{n \times t}$ corresponding to the smallest non-zero eigenvalues, where n is the number of nodes.
- Procrustes Alignment: Align the embedding matrices U_1 and U_2 of two graphs G and G' via Procrustes analysis, find an orthogonal matrix W to minimize:

$$\min_{W^T W = I} \|U_1 - U_2 W\|_F^2$$

The minimized residual $\|U_1 - U_2 W\|_F$ quantifies the overall topological difference between graphs G and G': a larger residual indicates lower structural similarity between the two graphs. Based on this, a normalized similarity metric is defined as follows:

$$S = \frac{1}{1 + \|U_1 - U_2 W\|_F}$$

to reflect the structural consistency between the two graphs.