Received February 8, 2022, accepted February 20, 2022, date of publication February 28, 2022, date of current version March 8, 2022.

*Digital Object Identifier 10.1109/ACCESS.2022.3155520*

# An Accurate Matching Query Method of Natural Language Knowledge Graph Based on Hierarchical Graph Topological Sequence

**QIFENG ZOU[1] AND CHAOZE LU[2]**
[1]School of Humanities, Ningbo University of Finance and Economics, Zhejiang, Ningbo 315175, China
[2]School of Electronic and Information Engineering, Ningbo University of Technology, Zhejiang, Ningbo 315211, China

Corresponding author: Chaoze Lu (luchaoze@126.com)

This work was supported in part by the Research Foundation for Advanced Talents, and in part by the Grants-in-Aid for Scientific Research.

**ABSTRACT** In recent years, although the application of knowledge graph in natural language processing has made some progress, there are still some key problems to be solved, especially the matching query problem in natural language knowledge graph. Since the basic data model of knowledge graph is graph, the matching query problem in natural language knowledge graph is usually transformed into graph matching query problem. However, at present, the traditional graph matching technology applied in knowledge graph consumes too much time and has low query efficiency, which cannot meet the needs of users for large-scale natural language knowledge graph query. Based on the full analysis of the defects of the traditional graph matching technology applied in the knowledge graph, according to the characteristics of the natural language knowledge graph, in order to improve the query efficiency, we propose an accurate matching query method of graph hierarchical topological sequence based on the graph model of knowledge graph. Through experiment analysis, compared with the traditional graph model matching algorithm applied in knowledge graph query, this method can quickly filter the unqualified knowledge graph candidate sets, effectively reduce the number of knowledge graph candidate sets, and make it have more advantages in matching efficiency and time performance. In addition, compared with two algorithms of *GIndex* and *FG-Index*, this method has better performance in index construction time, average size of candidate set and average running time of online update.

**INDEX TERMS** Knowledge graph query, knowledge matching, graph matching, graph hierarchical topological sequence, natural language processing.

## I. INTRODUCTION

The concept of knowledge graph was put forward in the next generation intelligent search engine project released by Google in 2012 [1]. Its core technology is to extract the entity and its attribute feature information and the relationship information between entities from web pages, so as to form a knowledge network and create a new intelligent search mode with the support of knowledge network. The success of knowledge graph in intelligent search has also prompted many companies and research institutions to construct their own knowledge graph, such as Wikidata [2], Freebase [3], DBpedia [4], Yago [5], CN-DBpedia [6], zhishi.me [7], etc. These knowledge graphs are usually called generic knowledge graph. Due to the success of generic knowledge graph, many fields have begun to build their own domain knowledge graph. For instances, Tan *et al.* [8] constructed a knowledge graph of urban traffic domain for mining urban traffic problems. Dong *et al.* [9] constructed a knowledge graph about legal domain in order to efficiently mine case knowledge. Rotmensch *et al.* [10] constructed a knowledge graph in the medical domain based on electronic medical records to help medical diagnosis. Abu Salih *et al.* [11] gave the concepts of domain knowledge graph and generic knowledge graph and the differences between them, and also discussed the importance of domain knowledge from different perspectives [12]–[14], etc. From the current development, the domain knowledge graph is

The associate editor coordinating the review of this manuscript and approving it for publication was Justin Zhang.

more widely used and studied. However, whether general knowledge graph or domain knowledge graph, most of them are constructed from documents, materials and web pages described by natural language. Natural language is the main way of original information carrier. The information mining and using of natural language has been transformed into the knowledge mining and using of knowledge graph, which makes the research of natural language knowledge graph become a hot spot at present [15], [16]. To mine and use knowledge effectively, knowledge matching query is the most basic operation. In other words, the matching query on the knowledge graph is the most basic operation of applying the knowledge graph [17]. As we all know, the data model of knowledge graph is based on graph structure, which uses nodes to represent real world object entities and edges to represent the basic relationships between these entities. This structured description can better reflect the complex relationships in the real world. Obviously, the processing of knowledge graph information is essentially the processing of graph structure information. In graph matching, in order to improve the matching efficiency, the current main methods achieve this goal by establishing index technology based on graph features. According to the different features of the graph, it can be divided into three feature types: path, frequent subtree and frequent subgraph. Among them, GraphGrep [18] is a representative matching algorithm for constructing index based on path feature, and TreePi [19] and C-Tree [20] are representative matching algorithms for constructing index based on frequent subtree feature, Gindex [21] and FG-index [22] are representative matching algorithms for building indexes based on frequent subgraph features. Although these algorithms improve the matching efficiency, they ignore the huge time cost in the process of feature extraction and index construction. In addition, the knowledge graph is a graph that is updated frequently. Each update needs to reconstruct the index, and the online update time is long. In a word, the efficiency of existing algorithms in large-scale natural language knowledge graph matching query is too low. A more efficient algorithm is needed to meet the requirements of natural language knowledge graph query. Focusing on this problem, we fully analyzed the shortcomings of existing algorithms, developed an accurate query and matching method of natural language knowledge graph based on hierarchical graph topological sequence, and achieved the following innovative achievements:

- According to the basic framework of natural language knowledge graph query process, this paper formally defines the basic problem of knowledge matching of knowledge graph, and analyzes that this problem is essentially a subgraph isomorphic matching problem, so as to establish a theoretical basis for accurate matching.
- Inspired by the generation method of traditional graph topological sequence, a hierarchical graph topological sequence generation method of knowledge graph is proposed. The graph topological sequence generated by this method not only reflects the parallel information of nodes, but also shows the information of edges, which solves the defect that the traditional graph topological sequence loses the parallel information of nodes and edge information, it makes the graph topological sequence have richer semantic information.
- Based on the generation of hierarchical graph topological sequence, a subgraph isomorphic matching algorithm based on knowledge graph hierarchical topological sequence is proposed. Compared with *GIndex* and *FG-Index*, the algorithm has better performance in index construction time, average size of candidate set and online update running time.

The rest of this paper is organized as follows: Section II reviews the related work. Section III shows the basic definition of accurate matching query about natural language knowledge graph. Section IV discusses hierarchical topological sequence construction of knowledge graph. Section V designs knowledge graph matching method based on hierarchical graph topological sequence. In Section VI, experiment analysis is carried out. Section VII gives the conclusion and the future directions of research.

## II. RELATED WORKS

As a representation model of complex knowledge structure, knowledge graph has been widely used in different fields in recent years, such as knowledge graph model of social network relationship, knowledge graph model of protein interaction network relationship, knowledge graph model of intelligent transportation network, etc. Due to the construction of a large number of knowledge graphs in these fields, the demand for knowledge application query is also growing. Therefore, the subgraph matching technology of knowledge graph has become a research hotspot. In fact, the subgraph matching problem has been proved to be an NP complete problem by Garey *et al.* in 1979 [23], which also brings many difficulties and challenges about the subgraph matching problem in the application of knowledge graph query. For the research of subgraph matching, some scholars have also proposed some effective algorithms and methods in recent years, mainly divided into non-index matching method and index matching method.

For the non-index matching method, the most classical algorithm is Ullmann algorithm [24], which is a search method based on state space. In order to reduce the search space, the algorithm designs three pruning rules: node degree constraint, one-to-one mapping relationship and the consistency of node adjacency relationship. Through these rules, the search space and matching times are continuously reduced. The algorithm is simple and effective, and is widely used in small-scale subgraph matching problems. However, without considering the matching order of nodes, the matching efficiency of the algorithm is low, and its worst matching time complexity is $O(n!n^2)$ [25]. Cordella *et al.* [25] improved the problem that Ullmann algorithm does not consider the matching order for nodes, and proposed VF2 algorithm. The algorithm not only considers the matching order for nodes, but also improves the pruning strategy, which greatly reduces the search times. The worst time complexity of matching is $O(n!n)$, and the matching efficiency is significantly higher than Ullmann algorithm. Ullmann and VF2 algorithms only use the structure and semantic constraints of nodes to reduce the scope of search and matching, and only adapt to the small-scale subgraph matching problem. In order to improve this situation, many researchers improved the algorithm by considering the more complex path or subgraph structure characteristics around the nodes and making more effective pruning rules for the search process, so as to further reduce the search complexity, reduce the search times, and be more suitable for large-scale subgraph matching problems. For examples, the Spath algorithm proposed by Shang *et al.* [26] improves the ability and efficiency of screening candidate nodes by introducing the nearest distance centered on nodes as a constraint. The GraphQL algorithm proposed by Zhao *et al.* [27] quickly filters out a large number of nodes that do not meet the conditions by considering the spanning tree with pattern graph and alternative nodes as root nodes and depth first. The GADDI algorithm proposed by Zou *et al.* [28] filters out a large number of candidate nodes by defining the proximity discrimination substructure distance as a constraint. Compare with the previous two algorithms, Spath, GraphQL and GADDI algorithms strengthen the constraints of nodes from different perspectives, thus reducing the number of search processes. In practical application, they show better matching performance with the increase of graph scale [29].

For the index matching method, this method selects the effective features of the graph to establish the index table, and then extracts the features by searching the subgraph during the matching, so as to quickly reduce the search space in the index table. The accurate matching is completed according to the structure and semantic index information of the search subgraph, which is more suitable for large-scale subgraph matching problems. According to whether the graph needs feature mining, the technology is divided into the index technology of non-feature mining and the index technology of feature mining. The index technology of non-feature mining mainly uses the node, path or the structure around the node as the index information. There is no need to mine the features of the graph, which reduces the cost of feature mining. For instances, Shasha *et al.* [18] proposed an index construction GraphGrep algorithm based on path, which is mainly used for graph matching problems with node labels. It enumerates all paths of the graph whose length does not exceed a certain threshold, and arranges the node labels on the path according to the search order as the label path feature. He and Singh [20] proposed a C-tree algorithm for constructing index based on tree structure. The algorithm aggregates the graph into a hierarchical tree structure through multi-level clustering. Each node represents an aggregation result, and each node also contains important characteristics of subsequent nodes, which can efficiently filter alternative results, etc. The advantage of this kind of technology is that it uses simple structures such as simple path and tree to encode each node in the graph, which can quickly to establish the index. When the graph changes, it can efficiently adjust the index and meet the needs of real-time subgraph matching. The disadvantage is that it destroys the original structure of the graph, loses important structure information, and will cause a large number of wrong candidate graphs. The index technology based on feature mining uses the subgraph structure of the graph to replace the simple structures such as path and tree centered on the node as the index feature. It can better preserve the important structure information, but the number of subgraph structures will far exceed the number of simple structures, resulting in the exponential increase of the index scale. Therefore, it is necessary to mine the subgraph structure with strong discrimination in the index subgraph structure as the index feature to reduce the index scale. For examples, Yan *et al.* [21] proposed a *GIndex* algorithm with frequent subgraph of strong discrimination as the index feature. Experiments show that the index space of the algorithm is only 1/10 of the index space of GraphGrep algorithm, and the scale of alternative graphs is only 1/10 ∼ 1/3 of GraphGrep algorithm. The main defect of the algorithm is that mining frequent subgraph requires a large number of subgraph isomorphic calculations, and the time complexity is high. The GraphQL algorithm proposed by Zhao *et al.* [27] is based on the frequent subtree as the index feature. It shows that the frequent subtree and the frequent subgraph have the same candidate graph filtering ability, and the tree structure feature is simpler and more efficient than the frequent subgraph, because the time complexity of computing tree isomorphism is only O(n). The advantage of this kind of indexing technology is that it can quickly filter out about 99% of the graphs that do not meet the requirements on the static graph, and only 1% of the remaining graphs need to be accurately matched, which is very efficient. The disadvantage is that in the dynamic graph, because the graph will be constantly updated over time, it is necessary to remind the index features in the graph to establish a new index. The complexity of mining algorithm is high, which affects the overall matching performance, and it is difficult to meet the requirements in real time.

In a word, knowledge graph is a graph that needs to be updated frequently. Although the non-index matching method can find the matching graph accurately, the matching algorithm has high complexity and is not suitable for large-scale knowledge graph matching problems. In the index matching method, the index technology of non-feature mining improves the matching efficiency, but the time complexity is also high. The index technology of feature mining needs to constantly reconstruct the index for the changing knowledge graph, so the matching performance is poor and is not suitable for knowledge graph matching. Therefore, combined with the characteristics of natural language knowledge graph, this paper proposes a knowledge graph matching query method based on hierarchical graph topological sequence to solve the problem of knowledge matching of natural language knowledge graph.

## III. BASIC DEFINITION OF ACCURATE MATCHING QUERY OF NATURAL LANGUAGE KNOWLEDGE GRAPH

### A. NATURAL LANGUAGE KNOWLEDGE GRAPH

Documents, materials, web pages and other information are usually described in natural language. However, the diversity, scale and differentiation of these information in data terminals lead to the phenomenon of large amount, redundancy, looseness and disorder of information. How to organize the unstructured information to form useful information so that people can quickly and accurately obtain the required information is a difficult problem in the current big data information age. As a way of structured graphic information expression, knowledge graph takes knowledge as the object to describe the relationship between knowledge, which provides great possibility for obtaining useful information quickly and accurately. Because of the superior characteristics of knowledge graph, a large number of information expressed in natural language has been described by knowledge graph, and good results have been achieved in some fields. Because of the structural characteristics of knowledge graph, a large number of information expressed with natural language has been described by knowledge graph, and good results have been achieved in some fields. For instances, in the field of Web information search, Google describes web information as a knowledge graph to improve search accuracy. In the field of literature query, the HowNet library expresses the main information of literature into a knowledge graph, so as to improve the speed and accuracy of query. In the field of corpus construction, a large number of corpus document information is expressed by knowledge graph, which makes the corpus knowledge more orderly and hierarchical description, and promotes the search and use of corpus more convenient, accurate and fast. As shown in Figure 1, the knowledge graph of football player C Ronaldo's web pages information is constructed. The node mainly describes the relevant entity information extracted from the document, while the edge mainly describes the information of the relationship between entities.

*Definition 1 (Natural Language Knowledge Graph):* the documents, materials, web pages, etc. described by natural language form a directed acyclic graph through the construction rules of knowledge graph, which is called natural language knowledge graph, denoted as $N = (C, E, \Omega, l)$. Where $C$ represents the vertex set, and $E$ represents an edge set. Each edge $e \in E$ is composed of an ordered pair of two vertices $c_i \in C$ and $c_j \in C$, that is, $e =< c_i, c_j >$. $\Omega = \{\varphi_1, \varphi_2, \ldots, \varphi_n\}$ represents the set of vertex labels, vertex labels come from nouns of natural language, and vertex labels are not repeatable, i.e. $\forall\varphi_i \neq \forall\varphi_j$. $l$ represents the vertex label function, i.e. $l{:}V \to \Omega$, which is used to assign labels to each vertex.

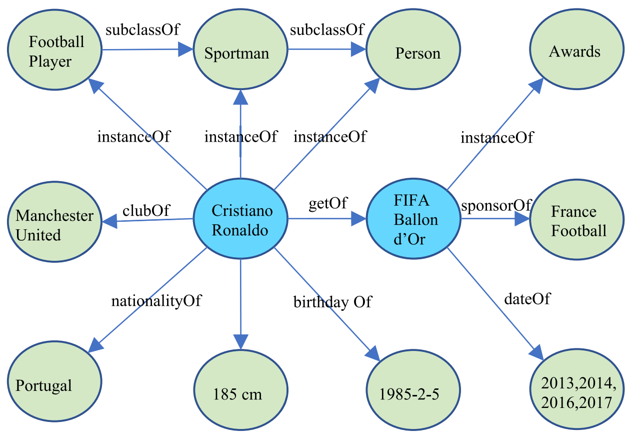


**FIGURE 1. Knowledge graph of web pages described by natural language.**

From definition 1, it can be seen that the main difference between natural language knowledge graph and generic knowledge graph is node label. The node label of natural language knowledge graph comes from nouns in natural language, and the label is non repeatable, while the label of generic knowledge graph can be a user-defined objective world entity, and it is allowed to be repeated. In addition, the natural language knowledge graph has domain characteristics, and its representation of the objective world knowledge network is more detailed.

### B. GENERAL FRAMEWORK OF KNOWLEDGE MATCHING QUERY

As a carrier of knowledge information, natural language knowledge graph has the greatest advantage that it can retrieve information according to the graph structured knowledge, so as to obtain the required knowledge. In the field of traditional knowledge graph information query, it is usually based keyword query, that is, to find a qualified node in a knowledge graph. However, this query method cannot give full play to the advantages of the knowledge graph. It only obtains the knowledge information of a single node and does not obtain the information associated with multiple nodes, resulting in too many query results and redundant information, which is difficult to meet the more accurate query requests for users. In order to overcome this shortcoming, people first extract the key knowledge information from the query request information and express it into a small knowledge graph. Then take this small knowledge graph as the input information of query, and search in the documents, materials and web pages represented by the knowledge graph, so as to improve the accuracy of query information. The internal reason for this good accuracy is that this small knowledge graph not only includes the knowledge of entity nodes in the query request, but also includes the association knowledge between entity nodes, so that it can obtain more accurate knowledge information when querying in the knowledge graph, so as to eliminate

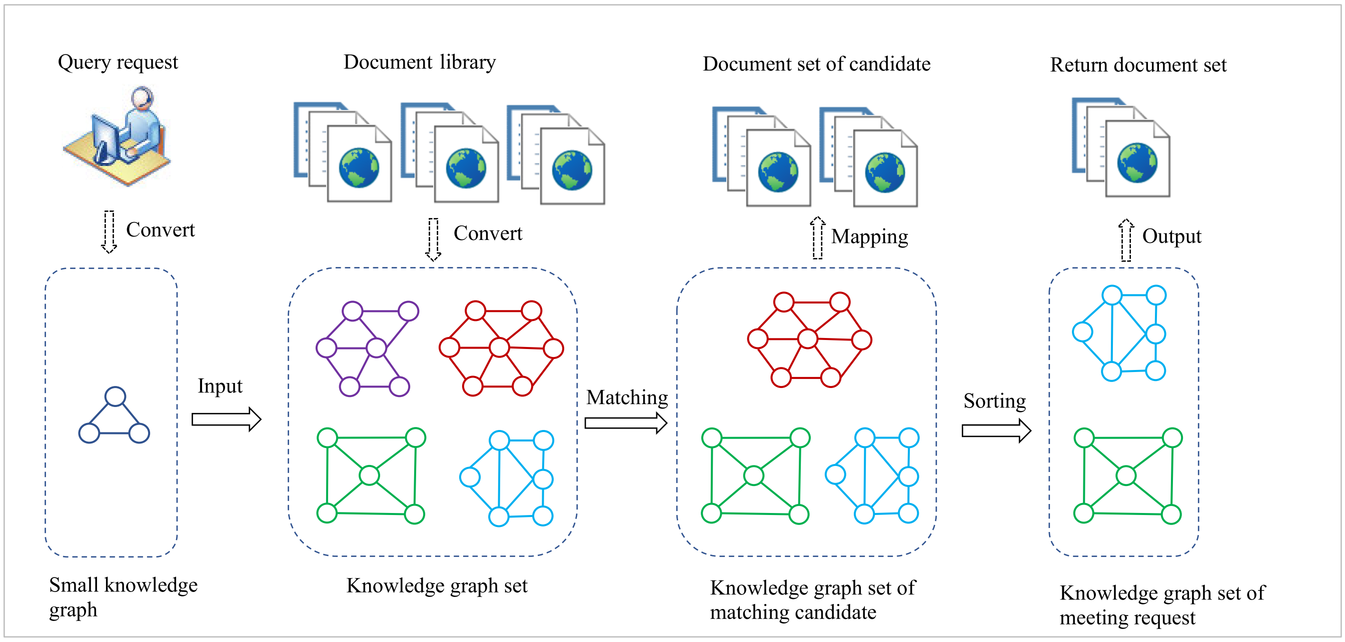


**FIGURE 2.** General process of knowledge matching query in natural language knowledge graph.

more redundant information. In essence, it is actually a graph data query problem. In other words, documents, materials and web pages are represented as knowledge graphs, and these knowledge graphs form a graph database. Then the query request is described as a small knowledge graph as input, and the matching search is carried out in the graph database. All the matching knowledge graph results in the graph database will be returned.

As shown in Figure 2, the user inputs the content of a query request. Firstly, these contents are converted into small knowledge graphs through the knowledge graph construction rules. Secondly, each document in the document library is also converted into one knowledge graph to form a knowledge graph set. Then, the small knowledge graph is matched on each knowledge graph in the knowledge graph set. If the matching is successful, the corresponding knowledge graph is added to the candidate knowledge graph set. Finally, the candidate knowledge graph set is sorted, mapped to the corresponding document set, and finally the document set that meets the request is returned. In this process, the key is to match the small knowledge graph on each knowledge graph in the knowledge graph set, which is also an important process that determines the query accuracy. In essence, it is the matching problem of a small graph in a large graph, that is, the isomorphic matching problem of subgraph. Subgraph isomorphic matching problem is an NP complete problem. Most of the solutions to this problem are to design subgraph query algorithms, but most of these algorithms construct indexes through subgraph features, so as to improve the performance of subgraph query. However, the maintenance cost of these indexes is high, and a more efficient algorithm is needed to meet the basic requirements of users' online query. This paper also discusses this problem in depth.

### C. DEFINITION OF KNOWLEDGE MATCHING QUERY PROBLEM

In order to more clearly describe the problems studied and uniform terminology in this paper, it is necessary to formally define some concepts and basic assumptions.

*Definition 2 (Data Knowledge Graph):* the natural language knowledge graph is also called data knowledge graph, denoted as $G = (C, E, \Omega, l)$. Where $C$ represents the node set, and $E$ represents the edge set. Each edge $e \in E$ is composed of an ordered pair of two nodes $c_i \in C$ and $c_j \in C$, that is, $e =< c_i, c_j >$. $\Omega$ represents the set of node labels. $l$ represents the node label function, i.e. $l$: $C \rightarrow \Omega$, which is used to assign labels to each node.

*Definition 3 (Query Knowledge Graph):* a piece of text entered in the query forms a directed acyclic graph through the construction rules of knowledge graph, which is called query knowledge graph, denoted as $P = (C_p, E_p, \Omega_P, l_p)$. Where $C_p$ represents the node set, and $E_p$ represents the edge set. Each edge $e \in E_p$ is composed of an ordered pair of two nodes $c_i \in C_p$ and $c_j \in C_p$, that is, $e =< c_i, c_j >$. $\Omega_P$ represents the set of node labels. $l$ represents the node label function, i.e. $l$:$C_p \rightarrow \Omega_P$, which is used to assign labels to each node.

Generally, the scale of query knowledge graph $P$ is much smaller than that of data knowledge graph $G$. In other words, the number of nodes of $P$ should be less than the number of nodes of $G$, and the number of edges of $P$ should be less than the number of edges of $G$, that is, size($C_p$) $\ll$ size($C$), size($E_p$) $\ll$ size($E$).

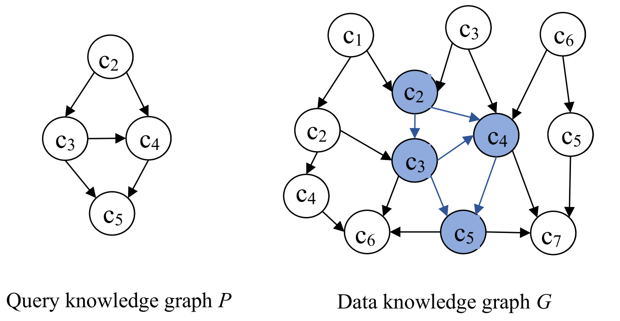


**FIGURE 3. A case of query knowledge graph *P* accurately matches data knowledge graph *G*.**

*Definition 4 (Data Knowledge Graph Set):* the set constructed from a set of data knowledge graph is called data knowledge graph set, denoted as $D_G = \{G_1, G_2, \ldots, G_n\}$, where $G_i$ represents any data knowledge graph.

*Definition 5 (Accurate Knowledge Matching Query):* suppose there is a data knowledge graph $G = (C, E, \Omega, l)$ and a query knowledge graph $P = (C_p, E_p, \Omega_P, l_p)$. If there is a sub data knowledge graph $G_{sub} = (C_{sub}, E_{sub}, \Omega_{sub}, l_{sub})$ in the data knowledge graph $G = (C, E, \Omega, l)$, that is, $G_{sub} \subseteq G$, so that there is a bijection function $f$ between $G_{sub}$ and $P$: (1) $\forall c \in C_p, l_p(c) = l_{sub}(f(c))$; (2) $\forall (c_i, c_j) \in E_p, < f(c_i), f(c_j) > \in E_{sub}$, and $l_p(c_i, c_j) = l_{sub}(f(c_i), f(c_j))$, that is, there is an isomorphic subgraph $G_{sub}$ of $P$ in $G$, which is called query knowledge graph $P$ accurately matches data knowledge graph $G$.

As shown in Figure 3, suppose there is a query knowledge graph $P$ and a data knowledge graph $G$, and there is a subgraph $G_{sub}$ (marked in blue) in $G$, so that $P$ and $G_{sub}$ meet the conditions of bijective function, that is, they have the same nodes $c_2$, $c_3$, $c_4$ and $c_5$, and the same edges $(c_2, c_3)$, $(c_2, c_4)$, $(c_3, c_4)$, $(c_3, c_5)$, $(c_4, c_5)$. However, as mentioned above, isomorphic subgraph matching is an NP complete problem. In the existing methods, graph features are used as indexes to solve this problem and improve the matching performance. Unfortunately, each update costs too much for index maintenance, so a more efficient subgraph isomorphic matching algorithm is urgently needed. Therefore, we adopt a matching method based on hierarchical graph topological sequence to solve this problem.

## IV. HIERARCHICAL TOPOLOGICAL SEQUENCE CONSTRUCTION OF KNOWLEDGE GRAPH

### A. TRADITIONAL GRAPH TOPOLOGICAL SEQUENCE

As mentioned above, we assume that both data knowledge graph and query knowledge graph are directed acyclic graph (DAG). Generally, the main idea of the traditional DAG topological sequence generation algorithm is as follows: in DAG, select a node with an indegree 0 each time as an output of the topological sequence, delete the node and all its outdegree edges, and then repeat the above operations until all nodes are output, and the output sequence of all nodes is the topological sequence of DAG. As shown in Figure 4, the topological sequence generation process of a DAG: select the first node $c_1$ with an indegree 0 as the output, and delete all its outdegree edges. At this time, the output sequence is: $c_1$. Then select the second node $c_3$ with an indegree 0, and delete all its outdegree edges. At this time, the output sequence is: $c_1c_3$. Repeat the above selective and deletion operation, and the final output sequence is: $c_1c_3c_4c_2c_6c_5c_7c_8$.

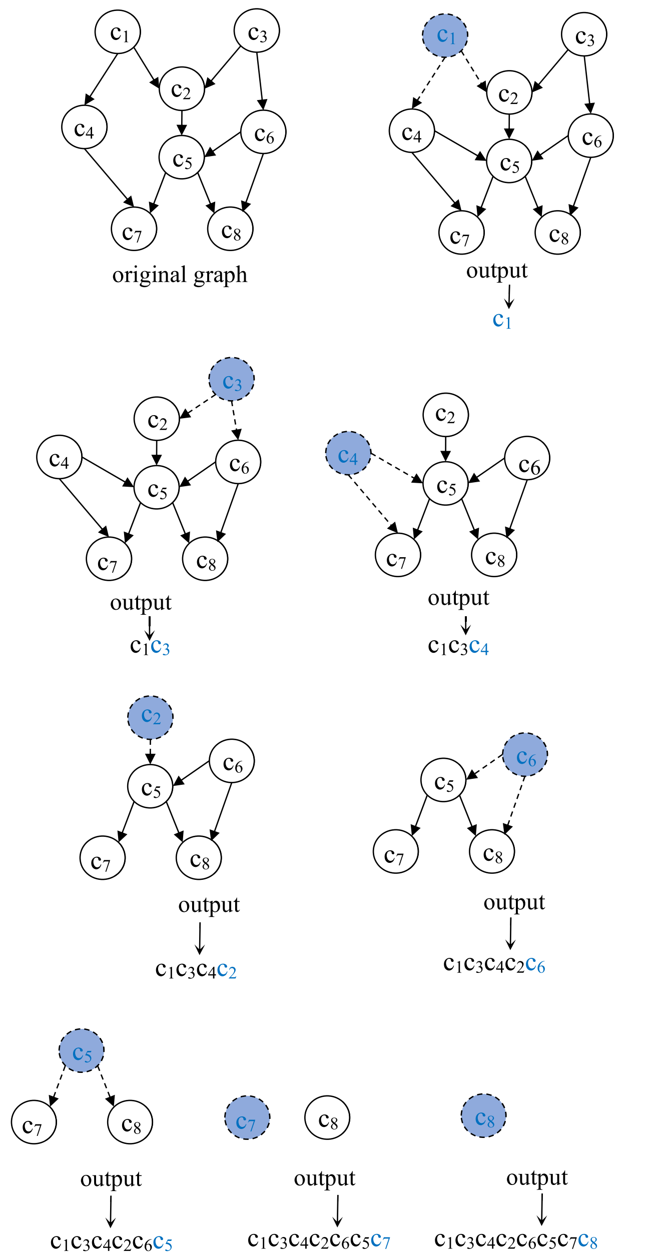


**FIGURE 4. Generation process of topological sequence of directed acyclic graph.**

However, such a topological sequence has two disadvantages: (1) the topological sequence loses the description of node parallel information. For example, both nodes $c_1$ and $c_3$ are nodes with an indegree 0. From the perspective of one-time output, they are parallel relations, which should be output at the same time, and such parallel relations should be reflected in the sequence. (2) The topological sequence loses the description of edge information. In the topological sequence, only the node information of the graph is represented, and the edge information is ambiguous, which not only makes the sequence indistinguishable, but also can not reflect the connection relationship of the nodes. Therefore, inspired by reference[30], we developed a hierarchical graph topological sequence to solve the above two defects.

## B. HIERARCHICAL GRAPH TOPOLOGICAL SEQUENCE

*Definition 6 (Partial Ordering Relation of Nodes):* In directed acyclic graph $G = (C, E, \Omega, l)$, for node set $C$, define partial ordering relation $c_i < c_j$ on each edge $(c_i, c_j)$ in $E$, where "$<$" is the strict partial ordering on node set $C$, which satisfies anti-reflexivity, asymmetry and transitivity, meaning that $c_i$ is ahead of $c_j$.

*Definition 7 (Hierarchical Topological Sequence of DAG):* In a directed acyclic graph $G = (C, E, \Omega, l)$, if there is $C = C_1 \bigcup C_2 \bigcup \ldots. \bigcup C_n$, and $C_i \bigcap C_j = \emptyset$, $i \neq j$, any $C_i = \{c_1, c_2, \ldots, c_m\}$, $m < n$, $C_i$ represents the nodes set with indegree 0 that after deleting all nodes from the node set $C_w$ and its outdegree edges, $w < i$, $C_w$ is the precursor nodes set of $C_i$, so the sequence $S$: $C_1/C_2/\ldots./C_n$ is called the hierarchical topological sequence of DAG, where "/" is the interlayer separation identifier.

Due to the traditional DAG topological sequence has defects and cannot meet the efficient graph matching performance, we propose a hierarchical directed acyclic graph topological sequence generation algorithm to make the sequence more distinguishable, so as to provide as much information as possible for graph topological sequence matching. Its main idea is as follows: select all nodes with an indegree 0 outputs each time, mark the parallel relationship about these nodes, and mark the information of outdegree nodes of each node about these nodes, so as to reflect the edge information. The process of one-time selection is called one layer of the sequence. The layer information is marked with special symbols, and the interlayer information is also marked with special symbols. Repeat the above process, until all nodes are output.

In the hierarchical graph topological sequence representation, the meanings of the symbols used for marking are as follows: "{}" represents all the information of the output nodes of a layer, "," represents the parallel relationship between nodes in a layer,"[]" represents the nodes of all outdegree edges of a node, and there is no order between nodes,"/" represents the separation of information between layers. For example, as shown in Figure 5, in the first layer, $c_1$ and $c_3$ are nodes with indegree 0, then they are placed in "{}" and separated by "," to represent the parallel relationship, that is, $\{c_1, c_3\}$. Then, the outdegree nodes $c_2$ and $c_4$ of $c_1$ are placed in "[]" and attached to the back of node $c_1$, that is, $\{c_1[c_2c_4], c_3\}$, and the outdegree nodes $c_2$ and $c_6$ of $c_3$ are placed in "[]" and attached to the back of node $c_3$, that is, $\{c_1[c_2c_4], c_3[c_2c_6]\}$, thus forming the first layer of the

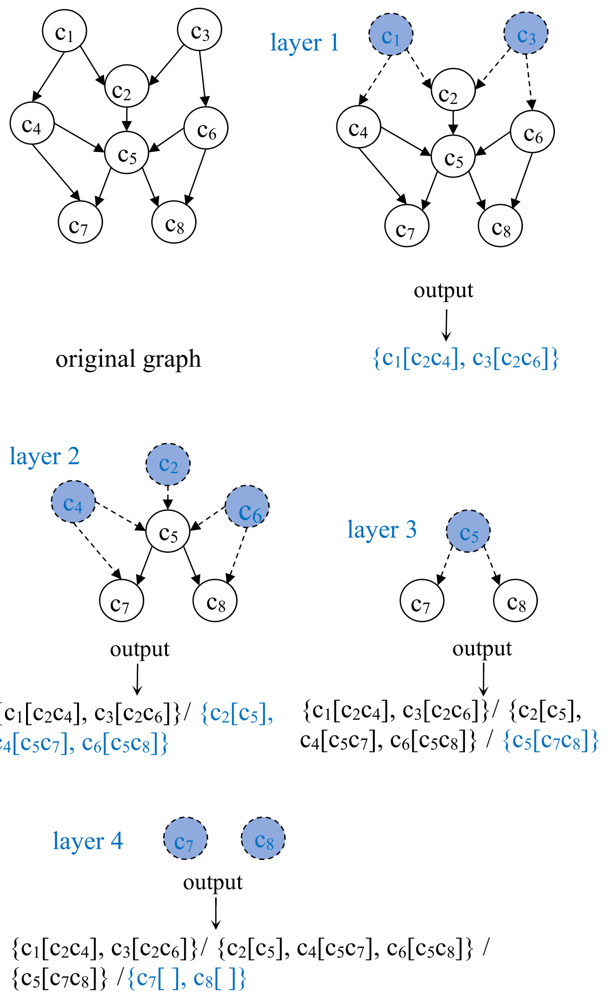

original graph
layer 1
output
{c1[c2c4], c3[c2c6]}
layer 2
output
{c1[c2c4], c3[c2c6]}/ {c2[c5], c4[c5c7], c6[c5c8]}
layer 3
output
{c1[c2c4], c3[c2c6]}/ {c2[c5], c4[c5c7], c6[c5c8]} / {c5[c7c8]}
layer 4
output
{c1[c2c4], c3[c2c6]}/ {c2[c5], c4[c5c7], c6[c5c8]} / {c5[c7c8]} /{c7[ ], c8[ ]}

**FIGURE 5. Generation process of hierarchical topological sequence of directed acyclic graph.**

topological sequence. Similarly, the topological sequence of the second layer is generated, but the layers are separated by "/", i.e. $\{c_1[c_2c_4], c_3[c_2c_6]\}/\{c_2[c_5], c_4[c_5c_7], c_6[c_5c_8]\}$. Other layers are generated similarly. Generation algorithm of hierarchical graph topological sequence is shown as follows.

*Proposition 1:* In a directed acyclic graph, when the number of nodes in the graph is greater than or equal to 2, the number of nodes in each layer from the hierarchical graph topological sequence must be less than the total number of nodes in the graph.

*Proof:* In a directed acyclic graph, when the number of nodes is equal to 2, there must be an edge $(c_i, c_j)$. According to algorithm 1, the two nodes from this edge must be distributed among two layers, i.e., there is one node in each layer, so the number of nodes in each layer is less than the total number of nodes in the graph. When the number of nodes is greater than 2, according to the characteristics of directed acyclic graph, as long as there is an edge between any two nodes, all nodes in a layer cannot be output, i.e., the number of nodes

**Algorithm 1** Generation Algorithm of Hierarchical Graph Topological Sequence, *GAH*()

```
Input: Directed acyclic graph $G = (C, E, \Omega, l)$
Output: Hierarchical graph topological sequence $S$:
        $C_1/C_2/\ldots./C_n$
1: $S \leftarrow \emptyset$; $C_j \leftarrow \emptyset$; $E_k \leftarrow \emptyset$;
2: while $C \neq \emptyset$
3:    for $\forall c_i \in C$ // Find nodes in the same layer
4:        if indegree($c_i$) = 0
5:          $N$ ←get_outdegree_nodes($c_i$);
6:          $C_j \leftarrow C_j + c_i + N$; // $C_j$ is the set of nodes in the
                 // same layer and its outdegree nodes
7:          $c_i.flag = 1$;
8:        end if
9:     end for
10:     $S \leftarrow S + C_j$;
11:    for $\forall c_i \in C$ //Delete all nodes and outdegree
                 //edges of the previous layer
12:    if indegree($c_i$) = 0 and $c_i.flag = 1$
13:     $C$ ←$C$- $c_i$;
14:    $E_k$ ← get_outdegree_edges($c_i$);
                 // $E_k$ is the outdegree edge set of $c_i$
15:    $E$ ←$E$- $E_k$;
16:    end if
17: end while
18: return $S$
```

in each layer is less than the total number of nodes in the graph. □

*Proposition 2:* Any two nodes with partial ordering relationship must not be on the same layer of hierarchical graph topological sequence.

Proof: Since any two nodes $c_i$ and $c_j$ with partial ordering relationship belong to two nodes of one edge $(c_i, c_j)$, according to algorithm 1, each layer always outputs a node with indegree 0. When the $c_i$ indegree is 0, the indegree of $c_j$ must not be 0, and $c_j$ must not be the output node of this layer, i.e., $c_i$ and $c_j$ are not on the same layer. □

## V. KNOWLEDGE GRAPH MATCHING METHOD BASED ON HIERARCHICAL GRAPH TOPOLOGICAL SEQUENCE

### A. MATCHING ALGORITHM OF HIERARCHICAL GRAPH TOPOLOGICAL SEQUENCE

*Definition 8 (Node matching):* Suppose $c_i$ and $c_j$ are nodes in the data knowledge graph $G = (C, E, \Omega, l)$ and the query knowledge graph $P = (C_p, E_p, \Omega_P, l_p)$ respectively. That is, $c_i \in C$ and $c_j \in C_p$, and the node sets of its outdegree edges are $O_{c_i}$ and $O_{c_j}$ respectively, if $l(c_i) = l_p(c_j)$, $O_{c_j} \subseteq O_{c_i}$, then node $c_j$ matches node $c_i$.

*Definition 9 (Knowledge Graph Sequence Matching):* Suppose that the data knowledge graph is $G = (C, E, \Omega, l)$ and the query knowledge graph is $P = (C_p, E_p, \Omega_P, l_p)$, and their corresponding hierarchical graph topological sequence is $S_G$ and $S_P$, if each node $c_j \in C_p$ in the $S_P$ is hierarchically matching node $c_i \in C$ in $S_G$, the topological sequence $S_P$ matches $S_G$.

Through the above *GAH* algorithm, the corresponding hierarchical graph topological sequences of data knowledge graph $G = (C, E, \Omega, l)$ and query knowledge graph $P = (C_p, E_p, \Omega_P, l_p)$ can be generated $S_G$ and $S_P$, then $S_G$ and $S_P$ realize graph topological sequence matching to achieve graph matching. The main idea of graph topological sequence matching is to match by layer, as follows: get all nodes of one layer from $S_P$, and each node respectively searches the corresponding matching node in $S_G$. If any node $c_j \in S_P$ searches the corresponding matching node $c_i \in S_G$ in $S_G$, it meets $c_i = c_j$, $O_{c_j} \subseteq O_{c_i}$, then the matching of one layer is completed and the matching of the next layer is carried out. Otherwise, one of the nodes does not meet $c_i = c_j$, $O_{c_j} \subseteq O_{c_i}$, the matching ends and a mismatch is returned. In this process, in order to improve the matching efficiency, we mark the positions of matching nodes in the $S_G$ of the upper layer. Then, the layer in front of the mark position is used as the starting position of the matching of the next layer, so that it is unnecessary to match from the front end when searching each layer, so as to reduce the matching length and matching time. Matching algorithm of hierarchical graph topological sequence is shown as follows.

**Algorithm 2** Matching Algorithm of Hierarchical Graph Topological Sequence, *MAH*()

```
Input: Sequence of data knowledge graph
     $S_G$: $C_1/C_2/\ldots./C_m$ and Sequence of query
     knowledge graph $S_P$: $C_1/C_2/\ldots./C_n$
Output: match_succe or match_fail.
1: $O_{c_i} \leftarrow \emptyset$;
2: $O_{c_j} \leftarrow \emptyset$;
3: for $\forall C_i \in S_P$ //Extract the node information of each layer
                 //of $S_P$
4:     for $\forall c_i \in C_i$
5:        $O_{c_i}$ ←get_outdegree_nodes($c_i$);
6:        for $\forall C_j \in S_G$ //Match by node information of each
                 //layer in $S_G$
7:          for $\forall c_j \in C_j$
8:            if $c_j \in C_j$ && $c_i = c_j$
9:              $O_{c_j}$ ←get_outdegree_nodes($c_j$);
10:             if $O_{c_j} \nsubseteq O_{c_i}$
11:              return match_fail;
12:             else if $O_{c_j} \subseteq O_{c_i}$
13:                  $S_P \leftarrow S_P$ - $c_i$-$O_{c_i}$;
14:                  if $S_P = \emptyset$
15:                    return match_succe;
16:                  end if
17:               end if
18:             end if
19:           end if
20:         end for
21:       end for
21:    end for
21: end for
```

For example, Figure 6 shows a data knowledge graph $G$ and a query knowledge graph $P$, which generate graph topological sequences $S_G$ and $S_P$ respectively through algorithm 1. Firstly, select the first layer node {$c_3[c_2c_6]$, $c_4[c_5c_7]$} in the $S_P$ to match in the $S_G$, and match the two nodes in the first and second layers of the $S_G$, which are shown by an underline in the Figure 6. Since the position of $c_3[c_2c_6]$ matched in $S_G$ is

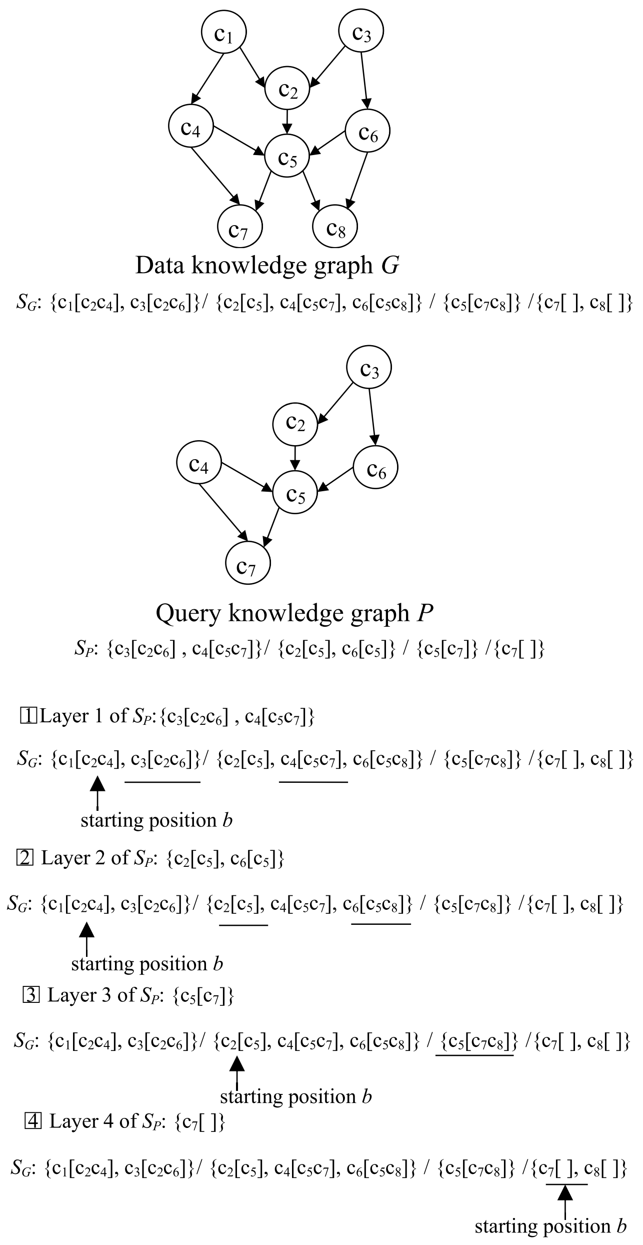

c1
c3
c2
c4
c6
c5
c7
c8
Data knowledge graph G
S_G: {c1[c2c4], c3[c2c6]}/ {c2[c5], c4[c5c7], c6[c5c8]} / {c5[c7c8]} /{c7[ ], c8[ ]}
c3
c2
c4
c6
c5
c7
Query knowledge graph P
S_P: {c3[c2c6] , c4[c5c7]}/ {c2[c5], c6[c5]} / {c5[c7]} /{c7[ ]}
1 Layer 1 of S_P: {c3[c2c6] , c4[c5c7]}
S_G: {c1[c2c4], c3[c2c6]}/ {c2[c5], c4[c5c7], c6[c5c8]} / {c5[c7c8]} /{c7[ ], c8[ ]}
starting position b
2 Layer 2 of S_P: {c2[c5], c6[c5]}
S_G: {c1[c2c4], c3[c2c6]}/ {c2[c5], c4[c5c7], c6[c5c8]} / {c5[c7c8]} /{c7[ ], c8[ ]}
starting position b
3 Layer 3 of S_P: {c5[c7]}
S_G: {c1[c2c4], c3[c2c6]}/ {c2[c5], c4[c5c7], c6[c5c8]} / {c5[c7c8]} /{c7[ ], c8[ ]}
starting position b
4 Layer 4 of S_P: {c7[ ]}
S_G: {c1[c2c4], c3[c2c6]}/ {c2[c5], c4[c5c7], c6[c5c8]} / {c5[c7c8]} /{c7[ ], c8[ ]}
starting position b

**FIGURE 6.** **Matching process of hierarchical topological sequence of knowledge graph.**

the first layer, set the first layer of $S_G$ as the matching starting position $b$ of the next layer of $S_P$. The second layer nodes $\{c_2[c_5], c_6[c_5]\}$ of the $S_P$ are matched to these two nodes in the second layer of the $S_G$, so the second layer of the $S_G$ is set as the matching starting position $b$ of the next layer of the $S_P$. It should be noted that the node of $S_P$ is $c_6[c_5]$, and its matching node in $S_G$ is $c_6[c_5c_8]$, which meets the conditions of "$l(c_i) = l_p(c_j)$, $O_{c_j} \subseteq O_{c_i}$" from definition 7, so it is regarded as node matching. When the layer 3 node $\{c_5[c_7]\}$ of the $S_P$ is matched in the $S_G$, but the starting position $b$ is in the layer 2 of the $S_G$, which reduces the matching time of the first layer. Similarly, the matching of other layers can be completed.

*Proposition 3:* Suppose there are data knowledge graph $G = (C, E, \Omega, l)$ and query knowledge graph $P = (C_p, E_p, \Omega_P, l_p)$, and their corresponding hierarchical graph topological sequence is $S_G$ and $S_P$, if the data knowledge graph $G$ contains the query knowledge graph $P$, the hierarchical topological sequence $S_P$ must be matched in $S_G$.

Proof: Since the data knowledge graph $G$ contains the query knowledge graph $P$, it indicates that there is $P_m$ in $G$, so that $P_m = P$, and the hierarchical graph topological sequence $S_{P_m}$ of $P_m$ is equivalent to the hierarchical graph topological sequence $S_P$ of $P$. So $S_{P_m}$ is contained in the hierarchical graph topological sequence of $G$. Obviously, the hierarchical topological sequence $S_P$ must be matched in $S_G$. □

### B. THE PROCESS OF SUBGRAPH MATCHING IN KNOWLEDGE GRAPH QUERY

Through the above designed hierarchical graph topological sequence generation and hierarchical graph topological sequence matching methods, the process of subgraph matching in knowledge graph query can be summarized, as shown in Figure 7. Specifically, it includes four parts: input, hierarchical graph topological sequence generation of knowledge graph, hierarchical graph topological sequence matching of knowledge graph and output. Firstly, obtain the query knowledge graph $P$ to be searched in the input, and select a data knowledge graph $G_i$ in the data knowledge graph database $D_G$, and use the *GAH* algorithm to generate the hierarchical graph topological sequence $S_P$ and $S_{G_i}$ respectively. $S_P$ and $S_{G_i}$ are used *MAH* algorithm to calculate the topological sequence matching of hierarchical graph. If it matches, the matching knowledge graph $G_i$ is output. Otherwise, if there is no match, judge whether each knowledge graph in the data knowledge graph database $D_G$ has been matched. If not, i,e. $D'_G = D_G$-$G_i \neq \emptyset$, then return to the input and reselect a data knowledge graph in $D_G$ for a new round of matching. Otherwise, it has been matched, i.e. $D'_G = D_G$-$G_\text{i} = \emptyset$, then no matching result is returned and the query is ended. The algorithm of knowledge graph query subgraph matching process is shown as follow.

## VI. EXPERIMENT ANALYSIS

### A. EXPERIMENT ENVIRONMENT

In order to verify the effectiveness of the proposed method, we conducted experiments on a Dell OptiPlex 7070 PC, which has Intel(R) Core(TM) i7-9700 CPU, 16GB RAM, 4TB HD, and runs 64 bit Windows 10 operating system. Under this basic hardware and software platform, we choose Visual Studio 2010 integrated development environment to realize the relevant algorithms, and the programming language is C++.

### B. EXPERIMENT DATA GENERATION

In recent years, graph matching technology has been widely used in many graph data application calculations and analysis

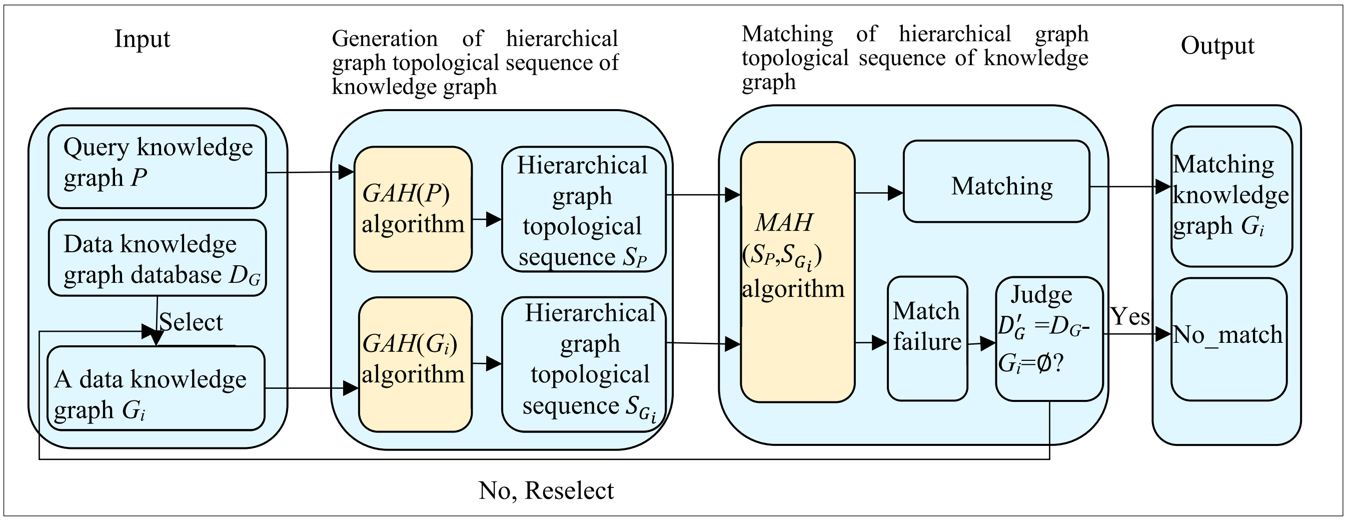

Input
Generation of hierarchical graph topological sequence of knowledge graph
Matching of hierarchical graph topological sequence of knowledge graph
Output
Query knowledge graph $P$
Data knowledge graph database $D_G$
Select
A data knowledge graph $G_i$
$GAH(P)$ algorithm
Hierarchical graph topological sequence $S_P$
$GAH(G_i)$ algorithm
Hierarchical graph topological sequence $S_{G_i}$
$MAH$ $(S_P, S_{G_i})$ algorithm
Matching
Match failure
Judge $D'_G = D_G - G_i = \emptyset$?
Yes
Matching knowledge graph $G_i$
No_match
No, Reselect

**FIGURE 7.** The subgraph matching process of knowledge graph query.

**Algorithm 3** The Algorithm of Knowledge Graph Query Subgraph Matching Process, *SMP*()

```
Input: Query knowledge graph $P = (C_p, E_p, \Omega_P, l_p)$ and
       Data knowledge graph $D_G = \{G_1, G_2, \ldots, G_n\}$.
Output: Matching knowledge graph $G_i$ or No_match.
1: $S_P \leftarrow \emptyset$;
2: $S_{G_i} \leftarrow \emptyset$;
3: $G_i \leftarrow$ select_a_knowledge-graph($D_G$);
4: while $D'_G = D_G$-$G_i \neq \emptyset$
5:     $S_P \leftarrow GAH(P)$;
6:     $S_{G_i} \leftarrow GAH(G_i)$;
7:     $M \leftarrow MAH(S_P, S_{G_i})$;
8:     if $M$ = match_succe
9:         return $G_i$;
10:    else if $M$ = match_fail
11:        if $D'_G = D_G$-$G_i = \emptyset$
12:            return no_match;
13:        else if $D'_G = D_G$-$G_i \neq \emptyset$
14:            continue;
15:        end if
16:        end if
17:    end if
18:    end if
19: end while
```

fields, and has attracted more and more attention from academia and industry. Many institutions have also published relevant evaluation data sets, which provides a basic public platform for the research of graph matching algorithm. Among them, the evaluation data sets include two kinds: data graph dataset and pattern graph dataset. For the data graph dataset, it is mainly divided into two categories: one is the real data graph dataset, and the other is the synthetic data graph dataset. Representative real data graph data sets mainly include AIDS Antiviral dataset [31], [32], NASA dataset [33] [34], Yeast dataset [35], Human dataset [36], etc. For example, AIDS antivirus dataset is the atomic structure dataset of chemical substances, NASA dataset is the XML dataset, Yeast dataset and Human dataset are the protein relationship dataset. These datasets are not representative of knowledge graph, and there are limitations in structure and scale. Therefore, we choose synthetic datasets for experiments. GraphGen [37] is a widely used graph data generation tool. It can automatically generate graph data to meet different needs by controlling different input parameters, including the number of generated graphs, the average number of nodes and edges of each graph, the number of labels of different nodes and edges, the average density of each graph, etc. The density of graph $G = (V, E)$ is defined as $d = (|E|)/(|V|(|V| - 1)/2)$, that is, the ratio of the number of edges in graph $G$ to the number of edges in a complete graph with the same number of nodes. The density of the generated graph data follows a normal distribution, with $d$ as the mean and 0.01 as the variance. For example, the dataset Synthetic.10k. E30. D3. L50 generated by Mohammadi and Wit [37] using GraphGen tool indicates that the dataset contains 10000 graphs, the average number of edges contained in each graph is 30, the average density of each graph is 0.3, and the number of labels of different nodes and edges is 50. In order to verify the correctness and effectiveness of the proposed method, we use GraphGen to generate a knowledge graph database, and then analyze the performance of various algorithms.

## C. VERIFY THE CORRECTNESS AND EFFECTIVENESS OF THE PROPOSED ALGORITHM

### 1) GENERATION TIME OF HIERARCHICAL GRAPH TOPOLOGICAL SEQUENCE OF KNOWLEDGE GRAPH

In this experiment, we mainly verify whether the hierarchical graph topological sequence of knowledge graph can be generated normally and what factors are related to the generation time of graph topological sequence.

Firstly, whether the hierarchical graph topological sequence of knowledge graph can be generated normally. We generated the graph dataset $G_d$ of synthetic.1K. E30.

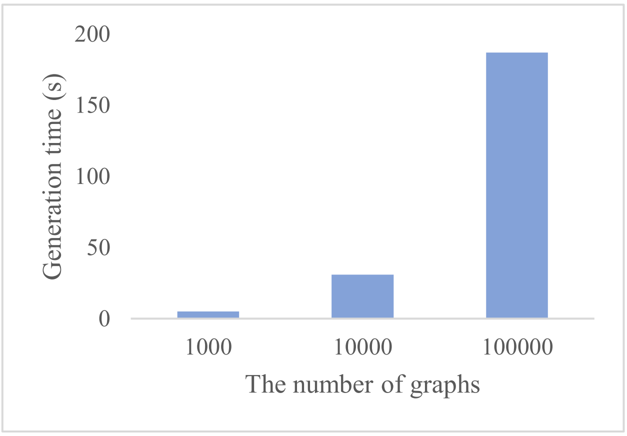


**FIGURE 8. Generation time of hierarchical graph topological sequence with different number of graphs.**

D3. L50 through GraphGen. In the above experiment environment, 1000 graphs in the graph dataset $G_d$ are selected and the hierarchical graph topological sequence is generated by *GAH* () algorithm 1. In order to reflect the randomness, 100 graphs are randomly selected from 10 groups to verify the correctness of hierarchical topological sequence output. Through experiment analysis, the hierarchical graph topological sequence can be correctly output, which proves the correctness of hierarchical graph topological sequence generation algorithm *GAH*()

Secondly, in order to verify which factors are related to the generation time of hierarchical topological sequence of knowledge graph, we set up four groups of experiments respectively. The experiment parameters of the first group are (1K. E30. D3. L50), (10K. E30. D3. L50), (100K. E30. D3. L50). That is, when the number of edges, density and labels remain unchanged, the generation time of hierarchical graph topological sequence with the number of graphs of 1000, 10000 and 100000 is shown in Figure 8. The experiment parameters of the second group are (10K. E20. D3. L50), (10K. E30. D3. L50), (10K. E40. D3. L50). That is, when the number, density and label number of graphs remain unchanged, the generation time of hierarchical graph topological sequence with 20, 30 and 40 edges is shown in Figure 9. The experiment parameters of the third group are (10K. E30. D2. L50), (10K. E30. D3. L50), (10K. E30. D4. L50). That is, when the number of graphs, edges and labels remain unchanged, the generation time of hierarchical graph topological sequence with average density of 0.2, 0.3 and 0.4 is shown in Figure 10. The experiment parameters of the fourth group are (10K. E30. D3. L30), (10K. E30. D3. L40), (10K. E30. D3. L50). That is, when the number, density and number of edges of the graph remain unchanged. The generation time of hierarchical graph topological sequence with 30, 40 and 50 labels is shown in Figure 11.

From the above experiment results, it can be seen from Figure 8 that the generation time of hierarchical graph topological sequence of knowledge graph is directly proportional to the number of graphs. Figure 9 shows that the

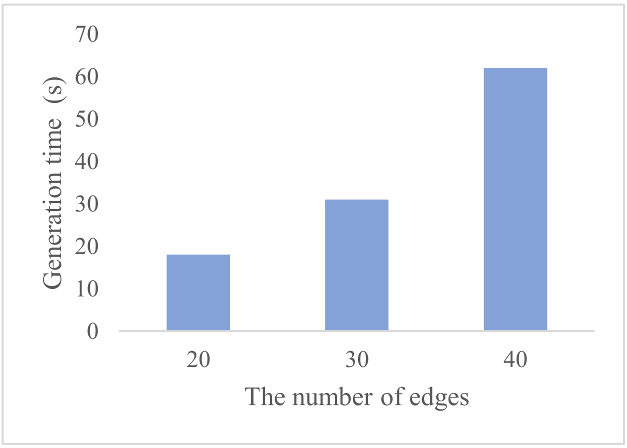


**FIGURE 9. Generation time of hierarchical graph topological sequence with different number of edges.**

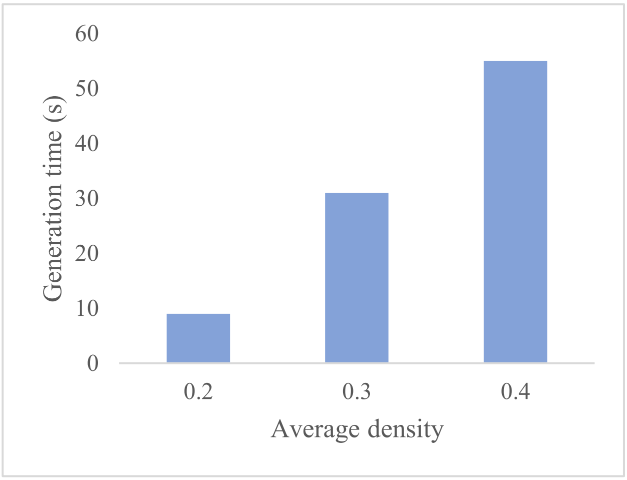


**FIGURE 10. Generation time of hierarchical graph topological sequence with different average density.**

generation time of hierarchical graph topological sequence increases with the increase of the number of edges, but it is not completely proportional. Figure 10 shows that the generation time of hierarchical graph topological sequence increases with the increase of graph average density, but it is not completely proportional. Figure 11 shows that the generation time of hierarchical graph topological sequence has nothing to do with the number of labels. Overall, in a knowledge graph set, the generation time of hierarchical topological sequence is positively proportional to the number, density and edge number of graphs.

### 2) MATCHING TIME OF HIERARCHICAL GRAPH TOPOLOGICAL SEQUENCE OF KNOWLEDGE GRAPH

In order to verify the influencing factors of the matching efficiency of hierarchical graph topological sequence, we set up two groups of experiments: (1) When the data knowledge graph set is fixed and the query knowledge graph is growing, we carry out the matching experiment analysis. (2) When the query knowledge graph is fixed and the data knowledge graph

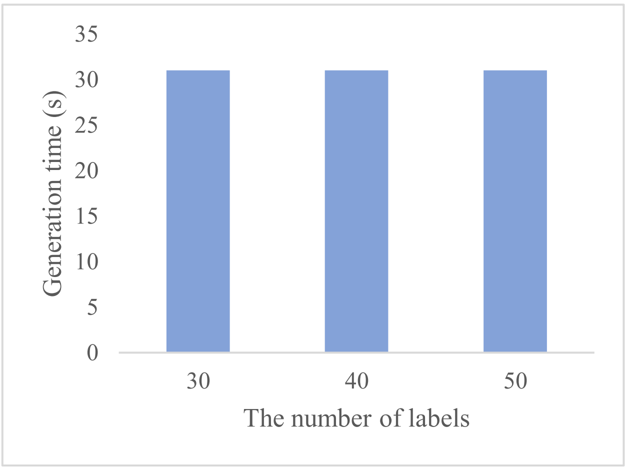


FIGURE 11. Generation time of hierarchical graph topological sequence with different number of labels.

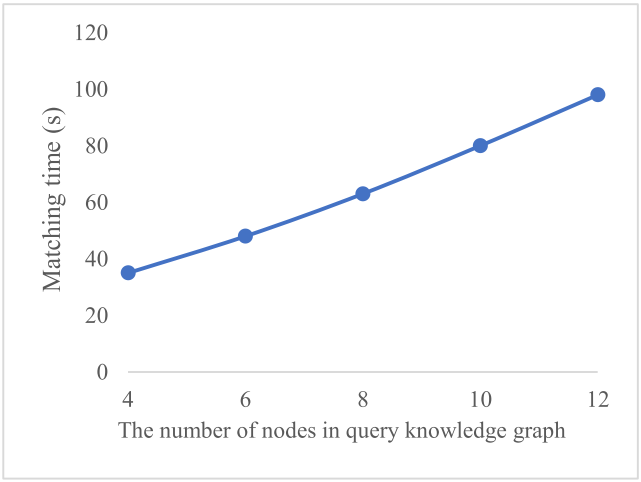


FIGURE 12. Matching time performance with increasing number of nodes in query knowledge graph.

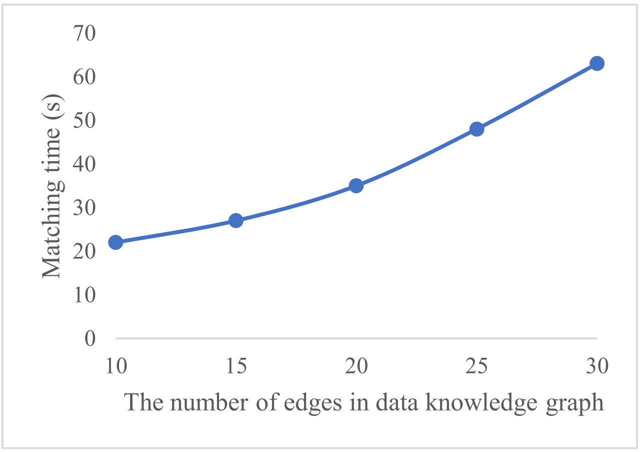


FIGURE 13. Matching time performance with increasing number of edges of data knowledge graph.

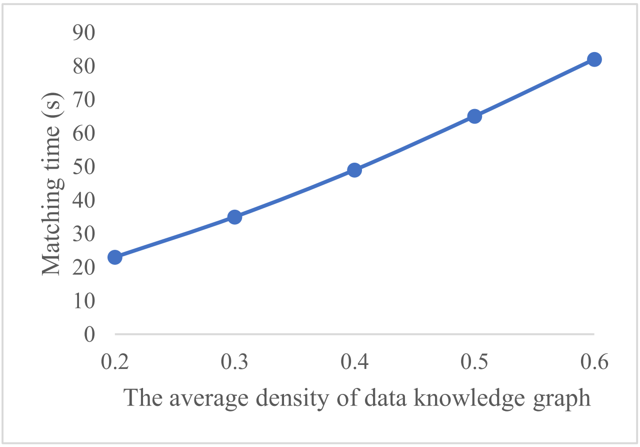


FIGURE 14. Matching time performance with increasing average density of data knowledge graph.

set is growing, the matching experiment analysis is carried out.

When the data knowledge graph set is fixed and the query knowledge graph is growing, the data knowledge graph set is generated with experiment parameters (10K. E30. D3. L50), and then five query knowledge graphs are generated with $n = (4, 6, 8, 10, 12)$ nodes, which are denoted as $P = \{P_4, P_6, P_8, P_{10}, P_{12}\}$. Both data knowledge graph and query knowledge graph use *GAH* algorithm to generate hierarchical graph topological sequences. Then query the knowledge graph $P_n$ use the *MAH* algorithm to do the topological sequence matching experiment on the data knowledge graph set. The time performance effect of matching is shown in Figure 12. With the increase of the number of query knowledge graph nodes, the matching time will continue to increase. It is proved that the higher the complexity, the slower the matching.

When the query knowledge graph is fixed and the data knowledge graph is growing, five data knowledge graph sets with increasing sides are generated respectively with the experiment parameters (10K. E10. D3. L50), (10K. E15. D3. L50), (10K. E20. D3. L50), (10K. E25. D3. L50) and (10K. E30. D3. L50), and $P_8$ with 8 nodes is used as the fixed query knowledge graph. Both data knowledge graph and query knowledge graph use *GAH* algorithm to generate hierarchical graph topological sequences respectively. Then, the topological sequence matching experiment is carried out on five data knowledge graph sets by using *MAH* algorithm with query knowledge graph $P_8$. The time performance effect of matching is shown in Figure 13. With the increase of the number of edges of the data knowledge graph, the matching time of the graph topological sequence will continue to increase, which also reflects the characteristics of the graph topological sequence we designed. Because there are all the information of outdegree nodes of a node in the sequence, it takes some time to process the matching of these node information.

When the query knowledge graph is fixed and the data knowledge graph is growing, then five data knowledge graph sets with increasing density are generated respectively with

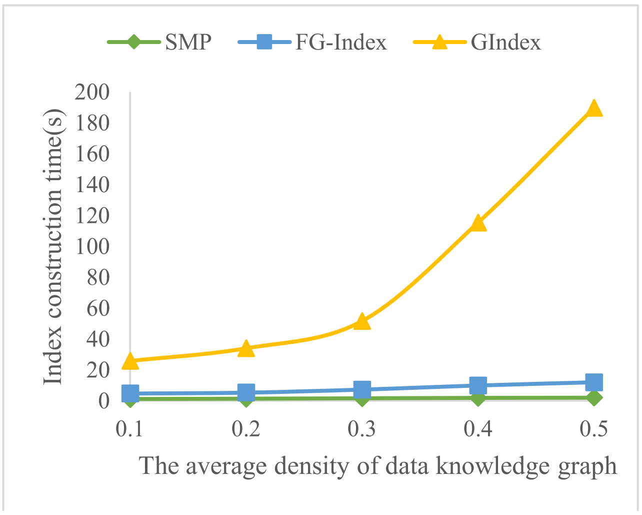


**FIGURE 15. Comparison of index construction time for three algorithms.**

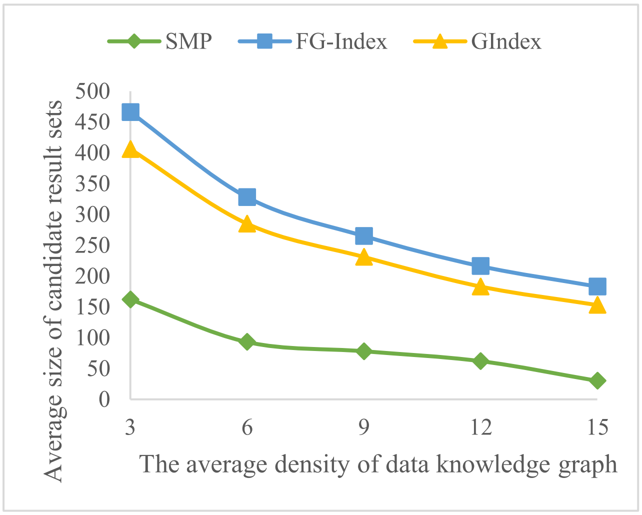


**FIGURE 16. Comparison of the average size of candidate result set for the three algorithms.**

the experiment parameters (10K. E20. D2. L50), (10K. E20. D3. L50), (10K. E20. D4. L50), (10K. E20. D5. L50), (10K. E20. D6. L50), and P8 with 8 nodes as the fixed query knowledge graph. and $P_8$ with 8 nodes is used as the fixed query knowledge graph. Both data knowledge graph and query knowledge graph use *GAH* algorithm to generate hierarchical graph topological sequences respectively. Then, the topological sequence matching experiment is carried out on five data knowledge graph sets by using *MAH* algorithm with query knowledge graph $P_8$. The time performance effect of matching is shown in Figure 14. With the increasing average density of data knowledge graph, the matching time of graph topological sequence will continue to increase, which also reflects the characteristics of graph topological sequence we designed. Because the number of nodes in the sequence will increase and the related outdegree nodes will also increase, it will take some time to process the matching of these node information.

### *D. PERFORMANCE ANALYSIS OF PROPOSED ALGORITHM AND OTHER ALGORITHMS*

In order to verify the performance of our proposed *SMP* algorithm, *GIndex* and *FG-Index* are selected to compare and analyze the performance in three aspects: index construction time, average size of candidate result set and average running time of online update.

#### 1) COMPARISON OF INDEX CONSTRUCTION TIME

The index construction time was analyzed on three knowledge graph sets with different densities (10K. E30. D2. L50), (10K. E30. D3. L50), (10K. E30. D4. L50). As shown in Figure 15, the *SMP* algorithm takes less time to construct the index, and the index construction time is almost stable at a low time value at different densities. *GIndex* algorithm takes the longest time, because it needs to mine and select frequent subgraphs with strong identification and filtering ability as index items, which consumes a lot of time in mining and selection, so index construction takes the most time. The construction time of *FG-Index* algorithm is mainly spent on mining all features and establishing a two-layer index structure. However, compared with *GIndex* algorithm, it is still less time-costing, but it is still too time-costing compared with *SMP* algorithm. Overall, *SMP* algorithm is the best, *FG-Index* algorithm is the second, and *GIndex* algorithm is the worst.

#### 2) COMPARISON OF AVERAGE SIZE OF CANDIDATE RESULT SETS

The average size of candidate result sets of *SMP*, *GIndex* and *FG-Index* algorithms are analyzed on the knowledge graph set (10K. E30. D3. L50). As shown in Figure 16, the average size of candidate result set of *SMP* algorithm is the smallest, the average size of candidate result set of *GIndex* algorithm is the second, and the average size of candidate result set of *FG-Index* algorithm is the largest. Therefore, it can be concluded that the filtering ability of *SMP* algorithm is stronger than *GIndex* algorithm and *FG-Index* algorithm, which is conducive to improving the time efficiency of candidate result set.

#### 3) COMPARISON OF AVERAGE RUNNING TIME OF ONLINE UPDATE

The average running time of online update of *SMP*, *GIndex* and *FG-Index* algorithms is analyzed on the knowledge graph set (10K. E30. D3. L50). As shown in Figure 17, the average running time of online update of *GIndex* algorithm is the largest, and the average running time of online update of *FG-Index* algorithm is the second, because they all need to reconstruct the index of the whole database when the update occurs. The average running time of online update of *SMP* algorithm is the smallest, because it only needs to construct a new index for the updated part of the data. It belongs to

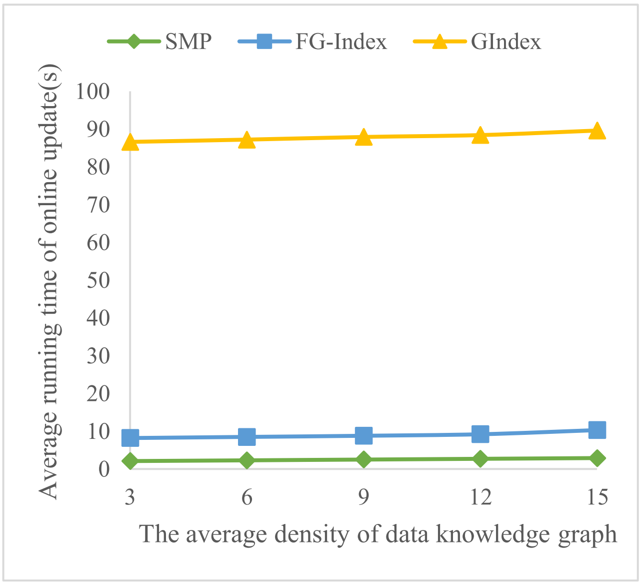


**FIGURE 17. Comparison of average running time of online update for three algorithms.**

incremental index structure, which has superior performance and is more suitable for online update query.

In summary, through the above experiments, we get the following conclusions: our proposed algorithm can normally generate the hierarchical graph topological sequence of knowledge graph. The generation time of hierarchical graph topological sequence is directly proportional to the number of graphs, density and edges, but not to the number of labels. The matching efficiency of hierarchical graph topological sequence is affected by the scale of query knowledge graph and data knowledge graph. The *SMP* algorithm we designed is better than the traditional *GIndex* algorithm and *FG-Index* algorithm in three aspects: index construction time, candidate set size and average running time of online update related to matching performance time.

## VII. CONCLUSION

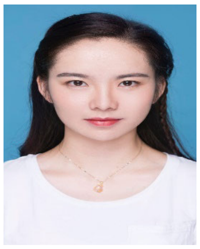
Focusing on the matching query problem of natural language knowledge graph, this paper presents a matching query method of hierarchical topological sequence of knowledge graph from the perspective of graph model, and designs the corresponding *SMP* algorithm. This method not only provides a new idea for knowledge graph matching query, but also solves the problem that the index construction time of knowledge graph matching query is too long. In addition, because the algorithm only needs to update the index of the new knowledge part when updating the knowledge graph online, it solves the problem that all the indexes need to be reconstructed when updating the knowledge graph online, so as to improve the efficiency of matching query. The comparison of *SMP* algorithm with traditional *GIndex* algorithm and *FG-Index* algorithm fully confirms these facts.

Although our algorithm improves the matching time efficiency of natural language knowledge graph, we still have to rely on parallel processing technology to further improve the matching efficiency when dealing with large-scale knowledge graph sets. Therefore, our future work will be carried out from the following aspects: studying the dynamic scheduling algorithm of hierarchical knowledge graph topological sequence in parallel processing, and studying the efficient parallel processing technology of graph matching based on hierarchical partition strategy, and studying the efficient knowledge graph matching algorithm based on the characteristics of processing nodes in parallel system, and studying the efficient knowledge graph matching algorithm for quickly unloading invalid knowledge data, and so on.

## REFERENCES

[1] Z. Chen, Y. Wang, B. Zhao, J. Cheng, X. Zhao, and Z. Duan, "Knowledge graph completion: A review," *IEEE Access*, vol. 8, pp. 192435–192456, 2020.
[2] D. Vrandečić and M. Krötzsch, "Wikidata: A free collaborative knowledgebase," *Commun. ACM*, vol. 57, no. 10, pp. 78–85, 2014.
[3] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: A collaboratively created graph database for structuring human knowledge," in *Proc. Int. Conf. Manage. Data*, 2008, pp. 1247–1250.
[4] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer, and C. Bizer, "DBpedia—A large-scale, multilingual knowledge base extracted from Wikipedia," *Semantic Web*, vol. 6, no. 2, pp. 167–195, 2015.
[5] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: A core of semantic knowledge," in *Proc. 16th Int. Conf. WWW*, 2007, pp. 697–706.
[6] B. Xu, Y. Xu, J. Liang, C. Xie, B. Liang, W. Cui, and Y. Xiao, "CN-DBpedia: A never-ending Chinese knowledge extraction system," in *Proc. Int. Conf. Ind., Eng. Other Appl. Appl. Intell. Syst.*, 2017, pp. 428–438.
[7] X. Niu, X. Sun, H. Wang, S. Rong, G. Qi, and Y. Yu, "Zhishi.me—Weaving Chinese linking open data," in *Proc. Int. Conf. Semantic Web*, 2011, pp. 205–220.
[8] J. Tan, Q. Qiu, W. Guo, and T. Li, "Research on the construction of a knowledge graph and knowledge reasoning model in the field of urban traffic," *Sustainability*, vol. 13, no. 6, pp. 1–20, 2021.
[9] B. Dong, H. Yu, and H. Li, "A knowledge graph construction approach for legal domain," *Tehnički Vjesnik*, vol. 28, no. 2, pp. 357–362, 2021.
[10] M. Rotmensch, Y. Halpern, A. Tlimat, S. Horng, and D. Sontag, "Learning a health knowledge graph from electronic medical records," *Sci. Rep.*, vol. 7, no. 1, pp. 1–11, Dec. 2017.
[11] B. Abu-Salih, "Domain-specific knowledge graphs: A survey," *J. Netw. Comput. Appl.*, vol. 185, no. 5, pp. 1–21, 2021.
[12] B. Abu-Salih, M. Al-Tawil, I. Aljarah, H. Faris, P. Wongthongtham, K. Y. Chan, and A. Beheshti, "Relational learning analysis of social politics using knowledge graph embedding," *Data Mining Knowl. Discovery*, vol. 35, pp. 1497–1573, May 2021.
[13] P. Wongthongtham and B. A. Salih, "Ontology-based approach for identifying the credibility domain in social big data," *J. Org. Comput. Electron. Commerce*, vol. 28, no. 4, pp. 354–377, Oct. 2018.
[14] B. Abu-Salih, P. Wongthongtham, and C. Y. Kit, "Twitter mining for ontology-based domain discovery incorporating machine learning," *J. Knowl. Manage.*, vol. 22, no. 5, pp. 949–981, Jun. 2018.
[15] A. Ait-Mlouk and L. Jiang, "KBot: A knowledge graph based ChatBot for natural language understanding over linked data," *IEEE Access*, vol. 8, pp. 149220–149230, 2020.
[16] L. He, B. Dong, and P. Jiang, "A heuristic grafting strategy for manufacturing knowledge graph extending and completion based on nature language processing: KnowTree," *IEEE Access*, vol. 9, pp. 90847–90862, 2021.
[17] Q. Xu, X. Wang, J. Li, Q. Zhang, and L. Chai, "Distributed subgraph matching on big knowledge graphs using pregel," *IEEE Access*, vol. 7, pp. 116453–116464, 2019.
[18] D. Shasha, J. T. L. Wang, and R. Giugno, "Algorithmics and applications of tree and graph searching," in *Proc. 21st ACM SIGMOD-SIGACT-SIGART Symp. Princ. Database Syst.*, 2002, pp. 39–52.

[19] S. Zhang, H. Meng, and J. Yang, "TreePi: A novel graph indexing method," in *Proc. IEEE Int. Conf. Data Eng.*, Apr. 2007, pp. 966–975.
[20] H. He and A. K. Singh, "Closure-tree: An index structure for graph queries," in *Proc. IEEE Int. Conf. Data Eng. (ICDE)*, Apr. 2006, pp. 38–48.
[21] X. Yan, P. S. Yu, and J. Han, "Graph indexing: A frequent structure-based approach," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2004, pp. 335–346.
[22] J. Cheng, Y. Ke, W. Ng, and A. Lu, "FG-index: Towards verification-free query processing on graph databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2007, pp. 857–872.
[23] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: Freeman, 1979.
[24] J. R. Ullmann, "An algorithm for subgraph isomorphism," *J. ACM*, vol. 23, no. 1, pp. 31–42, Jan. 1976.
[25] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento, "A (sub)graph isomorphism algorithm for matching large graphs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 10, pp. 1367–1372, Oct. 2004.
[26] H. Shang, Y. Zhang, X. Lin, and J. X. Yu, "Taming verification hardness: An efficient algorithm for testing subgraph isomorphism," in *Proc. 33rd Int. Conf. Very Large Data Bases*, 2008, pp. 364–379.
[27] P. Zhao, J. X. Yu, and P. S. Yu, "Graph indexing: Tree + delta >= graph," in *Proc. 33rd Int. Conf. Very Large Data Bases*, 2007, pp. 938–949.
[28] L. Zou, L. Chen, J. X. Yu, and Y. Lu, "A novel spectral coding in a large graph database," in *Proc. 11th Int. Conf. Extending Database Technol. Adv. Database Technol.*, 2008, pp. 181–192.
[29] J. Lee, W.-S. Han, R. Kasperovics, and J.-H. Lee, "An in-depth comparison of subgraph isomorphism algorithms in graph databases," in *Proc. 33rd Int. Conf. Very Large Data Bases*, 2012, pp. 133–144.
[30] L. Xi, L. Jianhua, and Z. Baili, "Topological sequence based DAG subgraph containment query," *J. Comput. Res. Develop.*, vol. 48, no. 2, pp. 343–349, 2011.
[31] Postech Database Lab. *NASA, Yeast, and Human Datasets*. Accessed: Oct. 18, 2021. [Online]. Available: http://dtp.nci.nih.gov/docs/aids/aids_data.html
[32] *NCI and AIDS Developmental Therapeutics Program*. Accessed: Oct. 18, 2021. [Online]. Available: http://dtp.nci.nih.gov/docs/aids/aids_data.html
[33] Postech Database Lab. *NASA, Yeast, and Human Datasets*. Accessed: Oct. 18, 2021. [Online]. Available: http://dblab.postech.ac.kr/iGraph/dataset/iGraph20_Data.zip
[34] UW CSE-UW Database Group. *XML Data Repository*. Accessed: Oct. 18, 2021. [Online]. Available: http://www.cs.washington.edu/research/xmldatasets
[35] W.-S. Han, J. Lee, M.-D. Pham, and J. X. Yu, "iGraph: A framework for comparisons of disk-based graph indexing techniques," in *Proc. Int. Conf. Very Large Data Bases*, 2010, pp. 449–459.
[36] *Gene Ontology Consortium*. Accessed: Oct. 18, 2021. [Online]. Available: http://www.geneontology.org/
[37] A. Mohammadi and E. Wit. *Synthetic Graph Data Generator*. Accessed: Oct. 18, 2021. [Online]. Available: http://www.cse.ust.hk/graphgen



**QIFENG ZOU** received the Ph.D. degree in linguistics from the Department of Humanities, University of Macau, in 2019. She is an Assistant Professor with the School of Humanities, Ningbo University of Finance and Economics. Her research interests include computational linguistics, comparative linguistics, corpus, knowledge graph, knowledge mining, natural language processing, and language translation processing method.

**CHAOZE LU** received the Ph.D. degree in computer science and technology from the Department of Computer Science and Technology, Tongji University, in 2021. He is an Assistant Professor with the School of Electronic and Information Engineering, Ningbo University of Technology. His research interests include graph data processing, data mining, knowledge modeling, knowledge graph, machine learning, software parallel computing, software evolution, software architecture, software modeling, software verification, and service computing.