Research Article

Dominance-Partitioned Subgraph Matching on Large RDF Graph

Bo Ning , Yunhao Sun , Deji Zhao , Weikang Xing , and Guanyu Li

School of Information Science and Technology, Dalian Maritime University, Dalian 116026, China

Correspondence should be addressed to Bo Ning; ningbo@dlmu.edu.cn

Received 2 November 2020; Revised 16 November 2020; Accepted 6 December 2020; Published 23 December 2020

Academic Editor: Wei Chen

Copyright © 2020 Bo Ning et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Subgraph matching on a large graph has become a popular research topic in the field of graph analysis, which has a wide range of applications including question answering and community detection. However, traditional edge-cutting strategy destroys the structure of indivisible knowledge in a large RDF graph. On the premise of load-balancing on subgraph division, a dominance-partitioned strategy is proposed to divide a large RDF graph without compromising the knowledge structure. Firstly, a dominance-connected pattern graph is extracted from a pattern graph to construct a dominance-partitioned pattern hypergraph, which divides a pattern graph as multiple fish-shaped pattern subgraphs. Secondly, a dominance-driven spectrum clustering strategy is used to gather the pattern subgraphs into multiple clusters. Thirdly, the dominance-partitioned subgraph matching algorithm is designed to conduct all isomorphic subgraphs on a cluster-partitioned RDF graph. Finally, experimental evaluation verifies that our strategy has higher time-efficiency of complex queries, and it has a better scalability on multiple machines and different data scales.

1. Introduction

The problem of subgraph matching is one fundamental issue in graph search, which is an NP-complete problem [1]. Specifically, given a query graph q and a large data graph G , the problem of subgraph matching is to extract all isomorphic subgraphs of q on G . However, one aspect is that the complex structure of a query graph depresses the query accuracy and performance on large data graphs, as the eruptive growth of data scale the in real world. Another aspect is that the data of social network tend to be organized as a rich-semantic structure. In this paper, we devote to research the subgraph matching problem on the large rich-semantic RDF graph.

Despite the complexity of knowledge structure and the polynomial-time problem of subgraph matching, recent existing researches have made significant advances in improving the performance of subgraph matching on large knowledge graph in a distributed environment.

One aspect is to encapsulate RDF data into triple-based relational tables [2, 3], which ensures the completeness of triple-based indivisible knowledge. Since the relational methods ignore the inherent graph-like structures of RDF

data, an expensive cost is incurred to consume the excessive join operations over relational tables. Another aspect is to manage RDF data into native graph formats, which typically employs adjacency lists to index RDF data [4–6]. Since the minimum edge-cutting strategies on large graphs depress the structure of indivisible knowledge, the enormous intermediate results are rigorous to balance the loading of partitioned RDF subgraphs.

To ensure the completeness of indivisible knowledge on graph-based formats, most researchers devoted to decompose the pattern graph into special-shaped subgraphs. StarMR [7, 8] decomposed query graphs to a set of star-shaped subgraphs, and then, two optimization techniques were employed to filter invalid input data and reduce the data of stars. CFLMatch [9] postponed the aggregate operations on a tree-shaped index constructed from the core-forest-leaf query partitioned model.

1.1. Contributions. In this paper, we devoted to decompose the pattern graph into fish-shaped subgraphs, considering the attributed and topological structures of the RDF graph. Then, the fish-shaped subgraphs are clustered and used to

guide the partition of large RDF graph. Finally, a subgraph matching algorithm is designed to conduct all isomorphic subgraphs on partitioned RDF subgraphs. Our contributions are illustrated as follows:

We proposed a dominant connected pattern graph to extract the dominating relationships of pattern graph, including node denotative relationship and node connotative relationship. The node denotative and connotative relationships discover the dominant and semidominant nodes in pattern graph. Then, fish-shaped pattern subgraphs are obtained through the dominant node-centered expansion.

We design a dominance-partitioned pattern hypergraph to model the fish-shaped pattern subgraphs. Each hypernode refers to a fish-shaped pattern subgraph, and each hyperedge denotes the common subgraph between fish-shaped pattern subgraphs.

We employ a dominance-driven spectrum clustering strategy to gather the fish-shaped pattern subgraphs to multiple clusters. A dominance-partitioned weighted matrix is first constructed from dominance-partitioned pattern hypergraph. Then, the spectrum clustering strategy is employed to gather the hypernodes into multiple clustering based on the weighted matrix.

We design a state transition model to describe the transition states of changed candidates, which consists of three states and six transition rules. Based on the state transition model, we analyze the influence of changed candidates to adjacent region and design our incremental maintenance strategy.

We propose a dominance-partitioned subgraph matching algorithm to conduct all isomorphic subgraphs on a cluster-partitioned RDF graph.

The rest of this paper is organized as follows: Section 2 introduces the preliminaries about problem definitions and related works. A framework of a dominance-partitioned RDF graph is provided in Section 3, including a dominant-connected pattern graph, dominance-partitioned pattern hypergraph, and dominance-driven spectrum clustering strategy. Section 4 presents a dominance-partitioned subgraph matching algorithm. Experimental results are reported in Section 5. A conclusion is given in Section 6.

2. Preliminaries

In this section, the definitions of RDF graph and subgraph matching are first given. Then, the related researches are introduced.

2.1. Problem Definitions. Resource description framework (RDF) [10] is a standard semantic model designed by a W3C group^{footnote} {<https://www.w3.org/community/kg-construct/>}, which is represented by a set of triples $\langle S, P, \text{and } O \rangle$. Each triple $\langle s, p, \text{and } o \rangle$ consists of three components: a subject, a predicate, and an object. Further, a triple $\langle s, p, \text{and } o \rangle$ is formed as $I \times I \times IL$, where I denotes an

IRI (Internationalized Resource Identifier) and L represents a literal.

Definition 1 (RDF graph). An RDF graph is a directed labeled graph, formed as $G(V, E, L, \text{and } \phi)$. Here, V is a set of vertices, $E \subseteq V \times V$ represents a set of directed edges, L denotes a set of labels about vertex and edge, and $V \cup E \longrightarrow L$ indicates a labeling function that assigns vertex and edge with the instantiated labels.

The labels of a RDF graph are classified as instance-label, relation-label, attribute-label, and type-label according to the resource and interresource relationship of RDF data. An RDF triple $\langle s, p, o \rangle$ is considered, o is named as **type-label** if and only if both s and o are IRIs, and p is a typed predicate, e.g., `rdf:type`, `rdf:subclass of`. The s and o are called as **instance-label**, p is named as **relation-label** if and only if both s and o are IRIs and p is not a typed predicate, and p is called as **attribute-label** if and only if o is a literal.

Considering an RDF graph in Figure 1, each vertex is labeled by an instance-label or a type-label or an attribute-label or a literal and each edge is labeled by a relation-label. The set of instance-labeled vertex is collected as $\{Person_A, Person_B, Publication_A, Course_B, Course_C, Department_A, University_A\}$. The set of type-labeled vertex is illustrated as $\{Publication, GraduateCourse, Course, Full-Professor, ResearchAssistant, Department, University\}$. The set of literal-labeled vertex is described as $\{123@163.com\}$. All the labels of edges are mapped to relation-labels.

Definition 2 (pattern graph). A pattern graph is a directed labeled graph, formed as $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{L}, \text{and } \psi)$, where $\mathcal{V}$ is a set of vertices, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ represents a set of directed edges, $\mathcal{L}$ denotes a set of labels about vertex and edge, and $\psi: \mathcal{V} \cup \mathcal{E} \longrightarrow var$ indicates a labeling function that assigns vertex and edge with the conceptual labels.

Considering a pattern graph in Figure 2, each vertex is mapped by type-label or attribute-label and each edge is mapped by relation-label. The difference of RDF and pattern graph is that the pattern graph does not contain the instance-labels. Further, the pattern graph is a conceptual network and each query graph is a subgraph of the pattern graph.

2.1.1. Subgraph Matching. The problem of subgraph matching is to search all possible subgraphs of data graph G that are isomorphic to query graph q . The subgraph matching is formally defined as a problem of subgraph isomorphism, described in Definition 3.

Definition 3 (subgraph isomorphism). Given a data graph $G(V, E, L, \text{and } \phi)$ and query graph $q(\mathcal{V}', \mathcal{E}', \mathcal{L}', \text{and } \psi')$, q is subgraph isomorphic to G if and only if there exists a bijective mapping M from $\mathcal{V}'$ to V such that $\forall u \in \mathcal{V}', \exists M(u) \in V: \mathcal{L}'[u] \subseteq L[M(u)]$ and $\forall u, u' \in \mathcal{V}', \exists (u, u') \in \mathcal{E}': (M(u), M(u')) \in E$ and $\mathcal{L}'[u, u'] = L[(M(u), M(u'))]$.

A query graph q is subgraph isomorphic to a data graph G if there exists a **subgraph isomorphic mapping** (subgraph mapping for short) of q on G . Simply, considering data and

FIGURE 1: RDF graph.

FIGURE 2: Pattern graph.

query graph in Figures 2 and 3 respectively, q is subgraph isomorphic to G since there exists a subgraph isomorphic mapping $M_1 \{ \langle \text{Person_A}, \text{FullProfessor} \rangle, \langle \text{Person_B}, \text{ResearchAssistant} \rangle, \langle \text{Course_B}, \text{GraduateCourse} \rangle \}$.

The k -partition problem on the RDF Graph is shown in Definition 4.

Definition 4 (K -partition problem on RDF graph (RG-KP)). Given an RDF graph $G(V, E, L, \text{ and } \varphi)$, the K -partition problem on the RDF graph refers to divide q into k subgraphs, satisfying $G = \{G'_1, \dots, G'_k\}$, such that overlapped cost $|G_{kp}| = \sum_{1 \leq i, j \leq k} (G'_i \wedge G'_j)$ is minimum and subgraph cost satisfies the condition $|G'_1| \approx \dots \approx |G'_k|$.

In this paper, our research of the RG-KP problem focuses on the dominance-partitioned strategy to divide the topological and tree-shaped structures of pattern graph. Then, a dominance-driven spectrum clustering is used to gather the dominance-partitioned pattern subgraphs into multiple clusters. Finally, dominance-partitioned subgraph

matching algorithm is designed to conduct all isomorphic subgraphs on a cluster-partitioned RDF graph.

In this paper, we focus on the directed labeled graphs. Both q and G are directed labeled graphs, and the directed or undirected edges cannot affect the execution scheduling of subgraph matching. Thus, the dominant connected pattern subgraph and dominance-partitioned pattern hypergraph are defined as an undirected graph without the graph-labels. The detailed notations and meanings are described in Table 1.

2.2. Related Works. In this section, we mainly review the related works on triple-based relational and graph-based traversal strategies in distributed environment.

2.2.1. Triple-Based Relational Strategy. Most RDF systems store and index RDF data as a set of triple tables in a relational database. SW-store [11] vertically partitioned RDF

FIGURE 3: Query graph.

TABLE 1: Notations and meanings.

Notations	Meanings
$G(V, E, L, \phi)$	An RDF graph
$\mathcal{Q}(\mathcal{V}, \mathcal{E}, \mathcal{L}, \psi)$	A pattern graph
$q(\mathcal{V}', \mathcal{E}', \mathcal{L}', \psi')$	A query graph, which is a subgraph of $\mathcal{Q}$
$\mathcal{Q}_{dc}(\mathcal{V}_{dc}, \mathcal{E}_{dc})$	A dominant-connected pattern subgraph
$\mathcal{Q}_d(\mathcal{V}_d, \mathcal{E}_d)$	A dominance-partitioned pattern hyper-graph
$\mathcal{V}_d$	A set of dominant nodes
$dom(u)$	A set of nodes dominated by u
$u < u'$	u dominates u'
$u \leq u'$	u semidominates u'
$\mathcal{P}(u)$	A circular-pattern subgraph of dominating node u
$\mathcal{T}(u)$	A tree-shaped pattern subgraph of dominating node u
$\mathcal{PT}(u)$	A dominance-partitioned pattern subgraph of node u , $\mathcal{PT}(u) = \mathcal{P}(u) \cup \mathcal{T}(u)$
C_i	A cluster of dominating nodes
$\mathcal{PT}_i$	A dominance-clustered pattern subgraph of C_i

triples into multiple attribute tables. RDF-3X [2, 12] and hexastore [3] implemented index-based query schemes through directly storing multiple arrangements of triple redundantly in B^+ -tree. Peng et al. [13] designed an RDF graph storage scheme to optimize graph division and balance the query loading, where the query processing was classified as two stages: scanning and joining. During the scanning phase, the query engine decomposed the SPARQL query into a set of triple patterns. In the joining phase, the scanned intermediate results are first bound into a left-joining tree, and then, the query results are conducted through the left-joining tree.

Distributed systems H-RDF-3X [14] and SHARD [15] horizontally divided RDF data into multiple computing nodes and used Hadoop as the communication layer for cross-node queries. H-RDF-3X divided the RDF graph into the specified number of partitioned data subgraphs through the minimum edge cut method METIS [16]. Then, the strategies of 1-hop or 2-hop replication were employed to extend the partitioned boundary of data subgraphs which ensured that small-diameter queries can obtain complete answers within single partitioned data subgraphs. The query processing of two systems used the reduce-side strategy that RDF triples are scanned in the mapping phase and the intermediate results are combined to the final results iteratively in the reducing phase. However, the iterative mapping and reducing operations of RDF triples can conduct expensive time-consumption on the complex topological structure of query graphs.

To reduce the complexity of topological query graph, a query decomposition model, called as TwinTwig [17], was designed to an efficient subgraph enumeration algorithm on distributed undirected graph. S2RDF [18] converted SPARQL

queries into RDD operations on the spark-distributed computing framework. Even though the offline indexes were built to speed up the online subgraph matching processing, the expensive time-consumption of index construction needs to be paid to match the large-scale data graph. TriAD [4] combined join-ahead pruning via the form of RDF graph summarization with a locality-based horizontal partitioning of RDF triples into a grid-like distributed index structure.

2.2.2. Graph-Based Traversal Strategy. The graph-based traversal strategies were employed to store RDF data in native graph format, which focused on the construction of data indexes and pruning rules of redundant intermediate results.

The constructed indexes of large data graphs were used to shrink the search space of candidate intermediate results. BitMat [19] proposed a compressed bit-matrix structure to store the huge RDF graphs, and a variable-binding-matching algorithm was directly designed to produce the final results without indexing the intermediate results. TripleBit [20] presented a fast and compact system for storing and accessing RDF data, which designed two auxiliary index structures to minimize the cost of index selection during query evaluation. A signature technology was proposed by gStore [21], which stored RDF data in disk-based adjacency lists and transformed an RDF graph into a data signature graph by encoding each entity and class vertex. Then, VS^* -tree was proposed over the data signature graph with light maintenance overhead. To enhance gStore, a redesigned gStore [22] was given a new query plan generation module that generated different query plans according to the structures of query graphs. Furthermore, it redesigned the

vertex encoding strategy to achieve more pruning power and a new multijoint algorithm to speed up the subgraph matching process.

The researches of pruning rules were employed to cut the redundant intermediate results in the processing of subgraph matching, such as Trinity.RDF [23] and WuKong [5]. Trinity.RDF was a distributed memory-based graph engine for web-scale RDF data. Instead of managing the RDF data in triple stores or as bitmap matrices, the engine stored RDF data in its native graph format to support the graph-based operations on RDF graphs, e.g., random walks, reachability, and community discovery. However, Trinity.RDF only used one-hop pruning rules to avoid the redundant path-based intermediate results, and a master machine was needed to aggregate all positive intermediate results. The researches [4, 5] found that the single-machine aggregate operations can easily conduct the bottleneck of a big query graph because the huge intermediate results can cause memory overflow on the single-master machine. Further, experiment [5] shows that the aggregate operations consumed more than 90% of the total matching time. Based on the experimental verification, WuKong adopted a full history pruning strategy to reduce the redundant intermediate results previously. However, a cost model of aggregate operation was designed to guide a matching order on the relational database. The expensive time-consumption of full-join Cartesian products should restrict the time-efficiency of conducted final results, but it only uses the cost model based on the predicate connection in the relational method to guide the query execution.

Most existing studies are devoted to ensure the completeness of indivisible knowledge. Xu et al. [24] studied the problem of multiobjective spatial keyword query with semantics and designed the LIR-tree index to integrate the spatial and semantic information of all objects in a balanced way. Wang et al. [25] created a new social network with more complete knowledge and proposed a k -Dcore framework to retrieve effective communities in the directed social network. Chen et al. [26] proposed a pivot-based hierarchical indexing structure S^2R -tree to integrate spatial and semantic information in a seamless way, which carefully designed a space mechanism to transform the high-dimensional semantic vectors to a low-dimensional space so that more effective pruning effect can be achieved. Cheng et al. [27] studied to automatically repair the graph with some repairing rules that designed a decomposition and join strategy to solve the polynomial time complexity of finding isomorphic subgraphs of graph data on graph-repairing rules. DistR [28] was an efficient-distributed strategy to solve the problem of reachability query over large uncertain graphs that found all of the maximal subgraphs of an original graph on the step of distributed graph reduction and transform the problem into a relational join process on the step of distributed consolidation. Deep NBCN [29] discovered the homogeneous and multibranch architecture to model the complex internal relationship between amino acid sequence and protein secondary structure sequence.

In this paper, we devote to decompose the pattern graph into partial subgraphs for reducing the time-consumption of aggregate operations. Our research motivation is induced by

the previous researches on a special structure discovery of the pattern graph. The first existing study was StarMR [7], which decomposed query graphs to a set of attribute stars to filter redundant input star-shaped data. The second existing study demonstrated [30, 31] that a topological structure was discovered by the analysis of anchored and followed relationships to reduce the discontinuous intermediate results. The third empirical study [32] is our previous work for subgraph matching on static knowledge graph that constructed a flow-based subgraph index to reduce redundant RDF data.

Benefiting from the previous researches, a dominance-partitioned pattern subgraph is designed to encapsulate the topological and attribute structures of pattern graph. Then, the large RDF graph can be partitioned in the aid of our dominance-partitioned pattern subgraphs, and the framework of the partitioned RDF graph is introduced detailedly in Section 3.

3. Framework of Dominance-Partitioned RDF Graph

In this section, a dominance-partitioned subgraph matching framework is proposed to conduct the subgraph mappings of query graphs on large data graphs. Firstly, a dominant-connected pattern graph is acquired from a pattern graph. Secondly, the large data graph is partitioned by a method of pattern-driven spectrum clustering. Finally, the subgraph mappings of query graphs on partitioned data graphs are conducted iteratively.

A pseudocode of DP-SM is described in Algorithm 1. A dominant-connected pattern subgraph (DCPG for short) is acquired from a pattern graph $\mathcal{Q}$, which is formed as $\mathcal{Q}_{dc}$. Firstly, a model of flow graph is employed to extract the dominated vertices and dominating relationships of the pattern graph. Then, the $\mathcal{Q}_{dc}$ is constructed from dominated vertices and dominating relationships of pattern graph (Line 1 and Section 3.1). Secondly, a method of spectrum clustering is used to divide the larger data graph based on the hypergraph of $\mathcal{Q}_{dc}$ (Line 2 and Section 3.2). Finally, the subgraph mappings of query graph q on partitioned data graph G_k are conducted iteratively (Line 3 and Section 4).

3.1. Dominant Connected Pattern Graph. The dominant-connected pattern subgraph $\mathcal{Q}_{dc}$ refers to a subgraph of pattern graph $\mathcal{Q}$, satisfying $\mathcal{Q}_{dc} \subseteq \mathcal{Q}$, which is expanded from the theory of Dominator Tree [33]. Considering a pattern graph $\mathcal{Q}(\mathcal{V}, \mathcal{E})$ and vertices $u, u' \in \mathcal{V}$, if there exists an artificially designed root $u_r \in \mathcal{V}$, such that u is the one necessary vertex on the path from u_r to u' , then u is a dominant node of u' , formed as $u < u'$. Similar to the dominating relationship among query vertices, if there exists an edge e of $\mathcal{E}$, satisfying e is a necessary edge on the path from u_r to u' , then $e < u'$. On the basis of dominating relations in $\mathcal{Q}$, the definition of dominant connected pattern graph is described in Definition 5.

Definition 5 (dominant connected pattern subgraph). Given a pattern graph $\mathcal{Q}(\mathcal{V}, \mathcal{E})$ and a root node $u_r \in \mathcal{V}$, a pattern subgraph is a dominant-connected pattern subgraph, formed as $\mathcal{Q}_{dc}(\mathcal{V}_{dc}, \mathcal{E}_{dc})$, if and only if it satisfies the conditions: (1) for any a node $u' \in (\mathcal{V} - \mathcal{V}_{dc})$, it always finds a node $u \in \mathcal{V}_{dc}$ such that $u < u'$ on the paths $|u_r, u' \rangle$. (2) For any an edge $e' \in (\mathcal{E} - \mathcal{E}_{dc})$ and the end-node u' of e , satisfying $e < u'$ on the paths $|u_r, u' \rangle$ where $|u_r, u' \rangle$ is the paths from root node u_r to query vertex u' . The dominating relationships $u < u'$ and $e < u'$ on $|u_r, u' \rangle$ refer that u and e are the necessary vertex and edge on the path from u_r to u' , respectively. We define the collector of dominant nodes in $\mathcal{V}_{dc}$ as $\mathcal{V}_d$, and then, it satisfies the condition $\mathcal{V}_d \subseteq \mathcal{V}_{dc} \subseteq \mathcal{V}$, and the set of nodes dominated by u is denoted as a dominant set of u , formed as $dom(u)$.

Considering a pattern graph $\mathcal{Q}$ and an artificial root node u_1 in Figure 4(a), the $\mathcal{Q}_{dc}$ of $\mathcal{Q}$ is shown in Figure 4(b). The dominating relationship is used to acquire the topological and attribute structures of the pattern graph; thus, the directionality of pattern graph cannot be considered in the calculation of $\mathcal{Q}_{dc}$. Considering the $\mathcal{Q}_{dc}$ in Figure 4(b), if any a node of $\mathcal{Q}_{dc}$ is deleted, such that $\mathcal{Q}_{dc}$ is invalid, then the $\mathcal{Q}_{dc}$ is minimum. Regarding a deleted node u_3 , satisfying edges (u_6, u_3) and (u_2, u_3) are also deleted from $\mathcal{Q}_{dc}$, and then $\mathcal{Q}_{dc}$ will be invalid, because there cannot find an edge dominating u_3 in $\mathcal{V} - \mathcal{Q}_{dc}$.

In this paper, we explore the characteristics of $\mathcal{Q}_{dc}$ to analyze the node denotative and connotative relationships in a pattern graph $\mathcal{Q}$. The node denotative relationship is discovered by the dominating relationships of vertices in pattern graph, as described in Theorem 1.

Theorem 1 (node denotative relationship). *Given a DCPG $\mathcal{Q}_{dc}(\mathcal{V}_{dc}, \mathcal{E}_{dc})$, if exists a dominant node $u \in \mathcal{V}_{dc}$, satisfying $dom(u)$ is nonempty, and then, the nodes of $dom(u)$ are constructed as a tree rooted by u .*

Proof. (for Theorem 1). A dominant set $dom(u)$ is considered, satisfying u is the one necessary vertex on the path $|u_r, u' \rangle$, thus $dom(u)$ and u can be combined into a flow graph originating from u . Different edges $e, e' \in \mathcal{E} - \mathcal{E}_{dc}$ is considered, if the common end-node u' is contained by them, then e and e' are not the dominant edges of u' . The nondominating relationship between vertex and edge contradicts the condition (2) in Definition 5.

The node connotative relationship is discovered by the vertex semidominant relation in a pattern graph. A DCPG $\mathcal{Q}_{dc}(\mathcal{V}_{dc}, \mathcal{E}_{dc})$ and a dominated vertex $u \in \mathcal{V}_{dc}$ are considered, if there exists a dominant vertex u on the path $|u_r, u' \rangle$, satisfying $u < u'$, such $d(u_r, u')$ is a minimum distance of $D(u_r, dom(u_r, u'))$, then u is a semidominant node of u' , formed as $u \leq u'$. Here, $dom(u_r, u')$ is a set of vertices dominating u' on the paths $|u_r, u' \rangle$, $d(u_r, u')$ denotes the distance from u_r to u' , and it is collected into the set $D(u_r, dom(u_r, u'))$, satisfying $u' \in dom(u_r, u')$, and $d(u_r, u') \in D(u_r, dom(u_r, u'))$. The node connotative relationship is defined in Theorem 2. $\square$

Theorem 2 (node connotative relationship). *Given a minimal DCPG $\mathcal{Q}_{dc}(\mathcal{V}_{dc}, \mathcal{E}_{dc})$, if exists a dominant node $u \in \mathcal{V}_{dc}$ semidominated by $u' \in \mathcal{V}_{dc}$, then the paths from u' to u are combined as a node or single-circular graph or multicircular graph.*

Proof (for Theorem 2). A minimal DCPG refers to the DCPG generated the least dominant nodes of a pattern graph. DCPG $\mathcal{Q}_{dc}(\mathcal{V}_{dc}, \mathcal{E}_{dc})$ is considered, if exists a dominant node $u \in \mathcal{V}_{dc}$ semidominated by $u' \in \mathcal{V}_{dc}$, satisfying there is a path from u' to u , and then, there must exist a smaller DCPG that does not contain u , thus $\mathcal{Q}_{dc}$ is not a minimal DCFG. Therefore, if u' and u are different dominant nodes, there must exist multiple paths from u' to u , which should be combined as a single-circular or multicircular graph, otherwise, they are the common dominant node. $\square$

3.2. Dominance-Partitioned Pattern Hypergraph. In this section, the method of dominance-partitioned pattern hypergraph (DPPG for short) is first given to construct the dominant nodes and dominating relationship. Then, a method of dominance-driven spectrum clustering is employed to divide a pattern graph as k subgraphs. The dominance-partitioned pattern hypergraph is defined in Definition 6.

Definition 6 (dominance-partitioned pattern hypergraph). Given a minimal DCPG $\mathcal{Q}_{dc}(\mathcal{V}_{dc}, \mathcal{E}_{dc})$, a dominance-partition pattern hypergraph is a hypergraph of DCPG, formed as $\mathcal{Q}_d(\mathcal{V}_d, \mathcal{E}_d)$ where $\mathcal{V}_d$ is a set of dominant nodes in $\mathcal{V}_{dc}$ and $\mathcal{E}_d \subseteq \mathcal{V}_d \times \mathcal{V}_d$ denotes a set of edges.

Both hypernodes and hyperedges of DPPG indicate the geometries of a pattern graph $\mathcal{Q}$, which are conducted on the basic of node denotative and connotative relationships in $\mathcal{Q}$.

The geometry of a node is a fish-shaped subgraph of $\mathcal{Q}$. A dominating set $dom(u)$ is considered, we define the denotative and connotative relationships of u as a tree-pattern subgraph $\mathcal{T}(u)$ and circular-pattern subgraph $\mathcal{P}(u)$, respectively. The combination of a tree-pattern subgraph and circular-pattern subgraph of dominating node u is denoted as (u) . Thus, (u) is a fish-shaped graph with a semidominant node of (u) as the fish head and leaves of (u) as fish tail. The geometry of edge indicates the circular-pattern common subgraph, which is composed of multiple paths between any dominant nodes in $\mathcal{V}_{dc}$.

Considering the pattern graph $\mathcal{Q}$ and dominant-connected pattern subgraph $\mathcal{Q}_{dc}$ in Figure 4, the dominance-partitioned pattern subgraphs are illustrated in Figure 5, where the rounds filled with diagonal lines denote the dominant nodes, and the rounds filled with vertical line indicate the semidominant nodes. Each dominance-partitioned pattern subgraph is composed of the multiple paths from a semidominant node to the dominant one and the dominated tree-shape structure of dominant node. Regarding a dominance-partitioned pattern subgraph (u) , it is composed of the multiple paths $\{|u_1, u_2, u_3 \rangle, |u_1, u_6, u_3 \rangle, \text{ and } |u_1, u_7, u_6, u_3 \rangle\}$ from u_1 to u_3 and the dominated tree-shaped structure $\{u_3, u_4, \text{ and } u_5\}$.

FIGURE 4: (a) Pattern graph. (b) Dominant connected pattern subgraph.

FIGURE 5: Dominance-partitioned pattern subgraphs. (a) (u_{15}). (b) (u_3). (c) (u_9).

Thus, the k -partition problem of a pattern graph can be converted to divide k subgraphs of hypergraph $\mathcal{Q}_{dc}$, which is denoted in Definition 7.

Definition 7 (dominance-driven k -partition problem). Given a dominance-partitioned pattern hypergraph $\mathcal{Q}_d(\mathcal{V}_d, \mathcal{E}_d)$ and $\mathcal{V}_d = \{u_1, \dots, u_n\}$, the dominance-driven k -partition problem refers to divide $\mathcal{V}_d$ into k clusters, satisfying $C = \{C_1, \dots, C_k\}$, such that overlapped cost $|C_{kp}| = \sum_{1 \leq i, j \leq k} (C_i \wedge C_j)$ is minimum and subgraph cost satisfies the condition $|C_1| \approx \dots \approx |C_k|$

where $|C_i|$ denotes the pattern-clustered subgraph cost of $\mathcal{PT}(C_i)$ and $\sum_{1 \leq i, j \leq k} (C_i \wedge C_j)$ indicates the overlapped cost of pattern-clustered subgraph $\mathcal{PT}(C_i), \dots, \mathcal{PT}(C_j)$.

In this paper, we abbreviate (C_i) as $\mathcal{PT}_i$. Then, the overlapped cost of a pattern-clustered subgraph is redefined as $|C_{kp}| = \sum_{1 \leq i, j \leq k} (\mathcal{PT}_i \wedge \mathcal{PT}_j)$, and subgraph cost is

represented as $\mathcal{PT}_1 \approx \dots \approx |\mathcal{PT}_k|$. Here, $\mathcal{PT}_i$ corresponds to the i -th of cluster C_i . Actually, the overlapped cost $|\mathcal{PT}_i \wedge \mathcal{PT}_j|$ is equivalent to $|\mathcal{P}_i \wedge \mathcal{P}_j|$, proven in Lemma 1.

Lemma 1 (node connotative relationship). Given pattern-clustered subgraphs $\mathcal{PT}_i$ and $\mathcal{PT}_j$, the overlapped cost $\mathcal{PT}_i \wedge \mathcal{PT}_j$ is equivalent to $\mathcal{P}_i \wedge \mathcal{P}_j$.

Proof. (for Lemma 1). Considering the node denotative relationship in Theorem 1, regarding dominant nodes $u \in \mathcal{PT}_i$ and $u' \in \mathcal{PT}_j$, if there exists $(u) \wedge \mathcal{T}(u') \neq \emptyset$, then $u < u'$ or $u' < u$. Regarding the dominating relationship $u' < u$, there must exist a semidominant node $u'' < u$, such that $u'' < u'$ or $u'' = u'$. Consider the node connotative relationship in Theorem 2, u' must be contained into $\mathcal{P}_i$. Thus, $(u) \wedge \mathcal{T}(u') \neq \emptyset$. $\square$

3.2.1. Dominance-Pattern Weighted Matrix. The dominance-pattern weighted matrix models the similarity of pattern-partitioned subgraphs through its bidirectional adjacency matrix W of size $n \times n$, where $n = |\mathcal{V}_d|$ and $\mathcal{V}_d$ denotes a set of dominant nodes. Further, $W[i, i]$ represents a subgraph cost of $\mathcal{PT}(u_i)$ and $W[i, j]$ represents an overlapped cost between $\mathcal{PT}(u_i)$ and $\mathcal{PT}(u_j)$, $W[i, j] \neq 0$ if and only if $\mathcal{PT}(u_i) \wedge \mathcal{PT}(u_j) \neq \emptyset$, otherwise $W[i, j] = 0$.

The subgraph cost of pattern-partitioned subgraph is evaluated by the quantity of data triples mapped to pattern triples. A pattern-partitioned subgraph (u) is considered, and the subgraph cost of (u) is the quantity of data triples mapped to all pattern triples in (u) , formed as $|u|$. Pattern-partitioned subgraphs $\mathcal{PT}(u_i)$ and $\mathcal{PT}(u_j)$ are considered, and the overlapped cost is the common subgraph cost of pattern subgraphs, formed as $|\mathcal{PT}(u_i) \wedge \mathcal{PT}(u_j)|$. The subgraph and overlapped costs are described in the following formulas:

$$W[i, i] = \sum_{p \in \mathcal{PT}(u_i)} \sum_{t \in G} |\langle p, t \rangle|, \quad (1)$$

$$W[i, j] = \sum_{p \in (u_i) \wedge \mathcal{PT}(u_j)} \sum_{t \in G} |\langle p, t \rangle|. \quad (2)$$

3.2.2. K-Partition on Dominance-Pattern Weighted Graph. The k -partition is used to cut the graph DCPG into k subgraphs that are not connected to each other. We define the sets of k dominant nodes in $\mathcal{V}_d$ as $C_1, \dots, C_k$, satisfying $C_i \cap C_j = \emptyset$ and $C_1 \dots C_k = \mathcal{V}_d$, $1 \leq i, j \leq k$.

For the dominant node sets C_i and C_j of any pattern subgraphs $\mathcal{PT}_i$ and $\mathcal{PT}_j$, we define the weight of the cut-graph between C_i and C_j as the following formula:

$$\mathcal{W}(C_i, C_j) = \sum_{l \in C_i, m \in C_j} W[l, m]. \quad (3)$$

Thus, for the dominant node sets $C_1, \dots, C_k$ of k pattern subgraphs $\mathcal{PT}_1, \dots, \mathcal{PT}_k$, the weight of the cut-graph on k pattern graphs is defined as the following formula:

$$\text{cut}(C_1, \dots, C_k) = \frac{1}{2} \sum_{i=1}^k \mathcal{W}(C_i, \bar{C}_i), \quad (4)$$

where $\bar{C}_i$ is the complementary of C_i , satisfying $\bar{C}_i = \mathcal{V}_d - C_i$.

3.3. Dominance-Partitioned Algorithm on Large RDF Graph. In this section, the k -partition algorithms are designed to divide the large data graph as multiple-distributed subgraphs. We first give the construction of DPPG in Algorithm 2, and then, the dominance-driven k -partition is designed in Algorithm 3.

The first core work of DPPG construction is to select a root node of a pattern graph. Intuitively, we tend to choose the node with the smallest local matching results and the largest degree as the root node. A query vertex conducting the smallest matching results means the minimal network

transmission cost, and the one with the greatest degree means the maximal probability to prune the negative node pairs. The evaluated formula of root node is described in the following equation:

$$u_r \leftarrow \arg \min_{u \in \mathcal{V}} \frac{|M(u)|}{d(u)}, \quad (5)$$

where $M(u)$ denotes the entities typed and attributed by u , satisfying $M(u) \in V$. W3C group provides a set of vocabularies (as part of the RDF standard) to encode rich semantic information on RDF graphs. For example, type predicates (rdfs:type) provide the function of grouping vertices of RDF graphs into different categories. Different from general label graphs, the vertices of RDF graphs identify entities/text information, and the semantic characteristics of entities result in the same type of vertices usually having similar predicate combinations, which is convenient for statistics. Thus, the entities of G can be easily mapped to pattern nodes of $\mathcal{Q}$ through typed and attributed predicates on entities themselves.

The construction of DPPG is shown in Algorithm 2. The input is a pattern graph $\mathcal{Q}(\mathcal{V}, \mathcal{E})$, and the output is a dominance-partitioned pattern hypergraph $\mathcal{Q}_d(\mathcal{V}_d, \mathcal{E}_d)$. An artificial root node is first selected by formula (5). The root node u_r is encoded by an initial sequential number $\text{DFS}[u_r] = 0$ and added to the dominant set $\mathcal{V}_d$ (Line 1). Then, the four modules are sequentially executed in the construction of DPPG.

The first module is to encode a sequential unique number to each vertex of G by the order of depth-first searching (Lines 2–6). Each node u of G is encoded iteratively through searching the successors of visited node u_r (Line 2). If u is unvisited, u is encoded and the successors of u are deployed into the encoder (Lines 3–4). If u is visited, u is taken as a dominant node and added to the dominant set $\mathcal{V}_d$ (Lines 5–6).

The second module is to identify the semidominant nodes of G in a descending order of depth-first searching (Lines 7–12). Each node u' of G is traced iteratively through searching the precursors of encoded node u (Line 8). If $\text{DFS}[u] < \text{DFS}[u']$, the edge (u, u') is collected into a circular-pattern subgraph $\mathcal{P}(u)$ and the precursor of u' are deployed into the tracer (Lines 9–10). If $\text{DFS}[u] > \text{DFS}[u']$ and $\text{sdom}[u] > \text{sdom}[u']$, the semidominant node of u is replaced by u' and the edge (u, u') is collected into circular-pattern subgraph $\mathcal{P}(u)$ (Lines 11–12).

The third module is to minimize the dominant set $\mathcal{V}_d$ (Lines 13–16). Considering a dominant node $u \in \mathcal{V}_d$, if there exists a node $u' \in \mathcal{V}_d - u$, satisfying $\text{DFS}[u] < \text{DFS}[u'] < \text{DFS}[\text{sdom}[u]]$, then u' is removed from $\mathcal{V}_d$.

The fourth module is to acquire the DPPG and dominance-partitioned pattern subgraphs (Lines 17–21). The dominated node of one node u is added to tree-pattern subgraph (u) , and the union of (u) and (u) is updated to (u) (Lines 17–18). Considering dominance-partitioned pattern subgraphs (u) and (u') if $(u) \wedge (u') \neq \emptyset$, a hyperedge (u, u') is constructed into $\mathcal{E}_d$.

3.3.1. Example for DPPG Construction. An example of DPPG construction is illustrated in Figure 6, where the rounds filled with diagonal lines denote the dominant nodes

and the rounds filled with vertical lines indicate the semi-dominant nodes.

A root node is first selected as u_1 by formula (5), which should contain with the smallest local matching results and the largest degree. Then, the four modules are sequentially executed in the construction of DPPG. The first module is for encoding a sequential unique number to each vertex of G by the order of depth-first searching. The sequential unique numbers are encoded as the subscripts of nodes, and the double visited nodes are collected to a dominant set $\mathcal{V}_d$, satisfying $\mathcal{V}_d = \{u_3, u_6, u_9, u_{15}\}$.

The second module is to identify the semidominant nodes of G in descending order of depth-first searching, and the semidominating relationships are acquired as $\{u_1 \leq u_3, u_1 \leq u_6, u_7 \leq u_9, u_1 \leq u_{15}\}$. In the acquiring process of semidominating relationships, the circular-pattern edges are inserted into circular-pattern subgraphs. Regarding a dominant node u_3 , the precursor u_6 of u_3 satisfies $\text{DFS}[u_3] < \text{DFS}[u_6]$, thus precursors of u_6 should be expanded continuously and (u_6, u_3) is inserted to $\mathcal{P}(\mathcal{V}_d u_3)$. After the semidominant node of u_3 are found as u_1 , the ascending path $|u_1, u_3\rangle$ is inserted to (u_3) . The dominance-partitioned pattern subgraphs are illustrated in Figure 6.

The third module is to minimize the dominant set. Considering a dominant node $u \in \mathcal{V}_d$, if there exists a node $u' \in \mathcal{V}_d - u$, satisfying $\text{DFS}[u] < \text{DFS}[u'] < \text{DFS}[\text{sdom}[u]]$, then u' is removed from $\mathcal{V}_d$. The fourth module is to acquire the DPPG and dominance-partitioned pattern subgraphs. The dominated tree-shaped structures are combined with circular-pattern subgraphs, which are used to conduct the compact dominance-partitioned pattern subgraphs.

The dominance-driven k -partition is described in Algorithm 3. The inputs are a dominant-connected pattern graph $\mathcal{Q}_d(\mathcal{V}_d, \mathcal{E}_d)$ and an RDF graph $G(V, E, L, \varphi)$. The output is a set of k -partitioned data subgraphs $G_k = \{G'_1, \dots, G'_k\}$. The dominance-driven k -partition algorithm consists of three modules.

The first module is to construct a similar matrix of a dominance-partitioned pattern hypergraph (Lines 1–4). The dominance-pattern weighted matrix initializes the similarity of pattern-partitioned subgraphs through its bidirectional adjacency matrix W of size (Line 1). $W[i, i]$ represents a subgraph cost of $\mathcal{P}\mathcal{T}_i$ (Line 3) and $W[i, j]$ represents an overlapped cost between $\mathcal{P}\mathcal{T}_i$ and $\mathcal{P}\mathcal{T}_j$ (Line 4).

The second module is to calculate the degree matrix through accumulating elements in each line of a similar matrix (Lines 5–6). Then, the method of Graph Laplacians is employed to calculate the feature matrix, satisfying $L = (D - W)$ and $f^T L f = (1/2) \sum_{i,j=1}^n W[i, j] (f_i - f_j)^2$. Thus, the similar matrix W can be abstracted as a feature matrix F of size $k \times |\mathcal{V}_d|$ (Line 7). Further, a k -means clustering method is used to gather the feature matrix line by line as k clusters (Line 8).

Finally, the third module is to generate k -partitioned data subgraphs through the mappings of data graphs on k clusters (Lines 9–11). Considering a pattern-clustered subgraph $\mathcal{P}(C_i)$ of cluster C_i , if there exists a data triple $t \in G$, satisfying $t \in \mathcal{P}\mathcal{T}(C_i)$, then t is a mapped data triple of $\mathcal{P}\mathcal{T}(C_i)$ (Line 11).

3.3.2. Example for Dominance-Driven K -Partition. Each dominance-partition pattern subgraph is taken as a hypernode of $\mathcal{Q}_d$ and each edge indicates the common part between dominance-partition pattern subgraphs. Considering circular-pattern subgraphs (u_3) and (u_6) in Figure 6, the edge (u_1, u_6) is the common part of (u_3) and (u_6) . Regarding a dominance-driven 3-partition strategy, the clustered pattern subgraphs are illustrated in Figure 5. An RDF graph and a pattern graph are considered in Figures 1 and 2, the divided pattern graphs and the partitioned RDF graphs are illustrated in Figure 7. The divided pattern subgraphs $\mathcal{Q}_1 (= \mathcal{P}\mathcal{T}(\text{Department}))$ and $\mathcal{Q}_2 (= \mathcal{P}\mathcal{T}(\text{GraduateCourse}))$ are shown in Figure 7(a), where $\mathcal{Q}_1$ and $\mathcal{Q}_2$ are fish-shaped pattern subgraphs semidominated by the common node *ResearchAssistant*. Regarding a dominance-driven 2-partition strategy, the divided pattern subgraphs do not need to be clustered, because the clustered pattern subgraphs are still the divided ones. Large RDF graph is partitioned into different servers based on the divided pattern subgraphs $\mathcal{Q}_1$ and $\mathcal{Q}_2$, illustrated in Figures 7(b) and 7(c).

4. Subgraph Matching on k -Partitioned RDF Graph

In this section, we introduce the subgraph matching algorithm on a k -partitioned RDF graph. Given a query graph q and a data graph G , the subgraph matching problem refers to search all isomorphic subgraphs of q on G . Here, the query graph q is defined as a subgraph of $\mathcal{Q}$, formed as $q(\mathcal{V}', \mathcal{E}')$.

4.1. Feasibility of Dominating Relationship. For corresponding to the flow characteristics of a pattern graph in this paper, we attach a query graph with a virtual root u'_r that is isotopic to the artificial root u_r of the pattern graph. The root-attached query graph is redefined as $q(u'_r, \mathcal{V}', \mathcal{E}')$, where u'_r is the virtual root node. Given a pattern graph $(\mathcal{V}, \mathcal{E})$ and an artificial root u_r , if there exists a refined query graph $q(u'_r, \mathcal{V}', \mathcal{E}')$, satisfying $u_r \in \mathcal{V}'$, then u' is a non-empty node and $u_r = u'_r$, otherwise, u' is an empty node. Considering the refined query graphs q_1 and q_2 in Figure 8(a), node u_1 of q_1 is a real root node of a pattern graph in Figure 9 and node u_r of q_2 is a virtual root node which is isotopic to node u_1 . To ensure the reachability of q_2 , the virtual edges are constructed from u_2 to the nearest nodes u_1 and u_7 on the reachable paths from u_1 in Figure 9.

The dominating relationships in a pattern graph are still suitable for the refined query graph, which is proved in Theorem 3.

Theorem 3 (feasibility of dominating relationship). *Given a pattern graph $\mathcal{Q}(\mathcal{V}, \mathcal{E})$ and query graph $q(u'_r, \mathcal{V}', \mathcal{E}')$, if there exists nodes $u, u' \in \mathcal{V} \cap \mathcal{V}'$, satisfying $u < u'$ in $\mathcal{Q}$, then dominating relationship $u < u'$ are still suitable in q .*

Proof (for Theorem 3). We use contradiction to prove the feasibility of dominating relationship. Considering a pattern graph $(\mathcal{V}, \mathcal{E})$ and query graph $q(u'_r, \mathcal{V}', \mathcal{E}')$, if there exists

FIGURE 6: DFS-based dominance-partitioned pattern subgraphs. (a) (u_{15}). (b) (u_6). (c) (u_3). (d) (u_9).

FIGURE 7: Divided pattern subgraphs and partitioned RDF graphs on servers. (a) Divided pattern subgraphs. (b) Partitioned graph on server 1. (c) Partitioned graph on server 2.

nodes $u, u' \in \mathcal{V} \cap \mathcal{V}'$, satisfying $u < u'$ in $\mathcal{Q}$, such that dominating relationship $u < u'$ cannot be suitable in q . Then, there must exist another node u'' , satisfying u'' is the reachable node on path $|u'_r, u'|$. Since u'_r is isotopic to u_r of $\mathcal{Q}$, u'' must be the reachable node on path $|u'_r, u'|$. However, the u will not dominate u' in $\mathcal{Q}$, because there exists two-node u and u'' , which are the reachable nodes on path $|u'_r, u'|$.

Regarding the refined query graphs q_1 and q_2 in Figure 8, the query graphs inherit the dominating relationships of the pattern graph, because it can always find a query root node (real or virtual node) that is isotopic to the root node of a pattern graph for any query graph.

Benefiting from the feasibility of dominating relationship in Theorem 3, the combination of a computational paradigm of subgraph matching and the concept of DCPG

FIGURE 8: Refined query graphs. (a) q_1 . (b) q_2 .

FIGURE 9: DFS-based selection of dominant nodes.

can conduct many outstanding characteristics to accelerate the subgraph matching on a RDF graph. The smallest calculated unit of subgraph matching is a data-query vertex pair (node pair for short), that describes a mapping from query vertex u to data vertex v , formed as $\langle v, u \rangle$. A solution of subgraph matching is defined as a subgraph mapping, described as $M = (\langle v_1, u_1 \rangle, \dots, \langle v_n, u_n \rangle)$, $n = |V_q|$. A DCPG-based characteristic is denoted in Lemma 2. $\square$

Lemma 2. *Given node pairs $\langle v, u \rangle$ and $\langle v', u' \rangle$ in a subgraph mapping M , if there exists a virtual node pair $\langle v_r, u_r \rangle$, satisfying $u < u'$ on $|u_r, u' \rangle$, then $v < v'$ on $|v_r, v' \rangle$.*

Proof (for Lemma 2). Regarding a subgraph mapping $M = (\langle v_1, u_1 \rangle, \dots, \langle v_n, u_n \rangle)$ of q on G , it must satisfy the edge constraint of subgraph isomorphism in Definition 2: $\forall u, u' \in V_q, \exists (u, u') \in E_q: (M(u), M(u')) \in E$. Thus, consider node pairs $\langle v, u \rangle$ and $\langle v', u' \rangle$ in M , if there exists the dominating relationship $u < u'$, $v (=M(u))$ must dominate $v' (=M(u'))$ according to the transitivity of edge constraint.

Further, an interesting characteristic can be mined from the positive and negative node pair. A node pair is called as a positive node pair if and only if it satisfies the candidate verification, shown in Definition 8. $\square$

Definition 8 (candidate verification). A node pair $\langle v, u \rangle$ is positive if and only if it satisfies the following constraints: (1)

$L_V(u) \subseteq L_V(v)$, (2) $\forall u' \in N(u), \exists v' \in N(v): L_V(u') \subseteq L_V(v')$, and (3) $\forall (u, u') \in E_q, \exists (v, v') \in E: L_E(u, u') \subseteq L_E(v, v')$

where L_V and L_E denote the labeling function of vertices and edges, respectively, $L_V(u)$ represents the vertex-label set coupled with vertex u and $L_E(u, u')$ represents the edge-label set coupled with edge (u, u') .

Lemma 3. *The node pairs dominated by $\langle v, u \rangle$ cannot conduct any one subgraph mapping containing $\langle v, u \rangle$ if $\langle v, u \rangle$ is negative.*

Proof (for Lemma 3). A negative node pair $\langle v, u \rangle$ is considered, it does not satisfy the constraints of subgraph isomorphism in Definition 2, thus $M(u) \neq v$. Further, the node pairs dominated by $\langle v, u \rangle$ cannot conduct any one subgraph mapping containing $\langle v, u \rangle$.

Therefore, we employ a circular-pattern first matching order to guide the iterative processing of subgraph matching. Considering a refined query graph q_1 containing a real root u_1 in Figure 8(a), the nodes of circular-pattern (u_{15}) are first ordered in the process of subgraph matching. Considering a refined query graph q_2 containing a virtual root u_r in Figure 8(b), the nodes of circular-patterns $\mathcal{P}(u_9)$ and $\mathcal{P}(u_3)$ are first ordered in the process of subgraph matching. Since q_2 contains two circular-patterns, it needs to select a priority for executing the multiple circular-patterns (u_9) and (u_3). The priority of multiple circular-pattern is selected by the density of

Input: a pattern graph $\mathcal{Q}$, a query graph q , and a data graph G
Output: the set M of all subgraph mappings of q on G_k
(1) -extraction from;
(2) $G_k \leftarrow k$ -Partition of $\mathcal{Q}_{dc}$ on G ;
(3) $M \leftarrow \text{subMatching}(q, G_k)$;
(4) **Return** M ;

ALGORITHM 1: Dominance-partitioned subgraph matching algorithm (DP-SM).

circular-patterns, which is formed as the proportion of edge to a vertex in a circular-pattern. Regarding the circular-patterns (u_9) and (u_3), the densities are calculated as 1 and 2/3, then (u_3) is executed in front of (u_9). Note that the calculation of density does not consider the virtual root and edges for a query graph coupled with a virtual root, similar to q_2 . $\square$

4.2. Physical Storage. We design the physical storage of patterned RDF graph to accelerate the acquisition of triple-based RDF data. Before the introduction of our physical storage, a dictionary encoding mapping table is first designed to encode the strings of RDF triples as integers.

The dictionary encoding mapping table consists of data and semantic dictionaries. The data dictionary corresponds instance-labels of RDF triple to integer unique numbers and the semantic dictionary corresponds predicates, type-labels, and attribute-labels of RDF triple to integer unique numbers.

Considering the RDF graph in Figure 1, the semantic and data dictionaries are illustrated in Table 2. Both data and semantic dictionary consist of two parts: the first part records the unique integer encodes (e.g., 1, 2, ...) and the second part records the instance-label, predicate, type-label, or attribute-label (e.g., *Person_A*, and *advisorBy*) corresponded to unique integer encodes in the first part. The *v-id* denotes a data dictionary, where each line encapsulates a unique integer and an instance-label (e.g., $\langle 1, \text{Person_A} \rangle$). The semantic dictionary is shown on the table of *p/type/attribute*, where each line encapsulates a unique integer and a predicate or a type-label or an attribute-label (e.g., $\langle 5, \text{workfor} \rangle$, $\langle 8, \text{rdf:type} \rangle$, and $\langle 16, \text{hasEmail} \rangle$).

Regarding the divided RDF graph in Figures 7(b) and 7(c), our physical storage is displayed in Figure 10, which employs the structural layout of hash mapping table. The *Key* is formed as $[v\text{-id}|p|type|dir]$, and the *Value* is assigned as *p/type*, where *v-id* is a unique integer encode, *p/type* represents a predicate or a type-label or an attribute-label, and *dir* indicates the direction of RDF triple on a graph. For example, a key $[1|0|0]$ on server 1 refers to the labels of in-edges to vertex numbered as 1, that is the predicate *advisorBy* marked as 1. A key $[1|8|1]$ indicate the type-label of the vertex numbered as 1, which is the label *FullProfessor* marked as 10. A key $[0|8|0]$ denotes the vertices coupled with an in-edge label *rdftype* marked as 8, that are *Person_A*, *Person_B*, *Course_A*, and *Course_B*, marked as 1, 2, 6, and 7, respectively.

4.3. Physical Storage. The subgraph matching of a *k*-partition RDF graph is described in Algorithm 4. The input is a query graph $q(u'_r, \mathcal{V}', \mathcal{E}')$ and the outputs are the subgraph mappings $\mathcal{M}$ of q on G_k .

The subgraph matching algorithm is originating with u'_r until all subgraph mappings are conducted. The number *i* is used to count the positive node pairs and a subgraph mapping is conducted to $\mathcal{M}$ if *i* is equivalent to $|\mathcal{V}'|$ (Lines 1-2). A circular-pattern first matching order is employed to guide the iterative processing of subgraph matching which benefits from Lemmas 2 and 3. An isotopic virtual root u'_r is attached to the original query graph and assigned to an initial number $i = -1$ (Line 4). Since u'_r is a virtual node, the successors of u'_r are expanded to explore the real node pairs (Line 5).

An iterative processing inserts the positive node pairs to $\mathcal{M}$ (Lines 7-11). Considering a selected query vertex u_i , the node pairs of u_i are previously extracted from G_k (Line 7). If the node pair $\langle v, u_i \rangle$ is positive and satisfies the partial subgraph isomorphism, $\langle v, u_i \rangle$ is extended to $\mathcal{M}$ and the successor of u_i is expanded to exploring the read node pairs. Otherwise, other node pairs are sequentially verified by partial subgraph isomorphism and candidate verification (Lines 8-10). If node pairs of a query vertex u_i are negative or do not satisfy the partial subgraph isomorphism, the precursor of u_i is backtracked and repeating the extending processing until all subgraph mappings are conducted to $\mathcal{M}$ (Line 11).

4.3.1. Example for Subgraph Matching Algorithm. In the subgraph matching algorithm, we employ a circular-pattern first matching order to guide the iterative processing of subgraph matching. A ordered query graph is shown in Figure 11, where the rounds filled with left-diagonal line denote the first executed region, the rounds filled with right-diagonal line indicate the second executed region, and the nonfilled rounds refer to the final executed region. The filled rounds are included in circular-pattern subgraphs and u_7 is the regional juncture. Then, the subgraph matching is iteratively conducted by our circular-pattern first matching order.

5. Experimental Evaluation

In this section, we verify the effectiveness and scalability of algorithms that are experimented on synthetic and real datasets, and we mainly analyze the experiments with current memory-based distributed SPARQL query processing strategies.

FIGURE 10: Physical storage of partitioned RDF graph.

FIGURE 11: Ordered query graph.

5.1. Experimental Settings. All experiments are conducted on a distributed cluster including six identical computing nodes. Each computing node uses an Intel(R) Core(TM) i7-7700@3.60 GHz 8-core processor, and the node communication is deployed on the Ethernet of 1000 Mbps. The physical memory is 16 GB, and the hard disk size is 1 T.

Experimental evaluation employs the four generated scales of synthetic data set LUBM (Lehigh University BenchMark) and the real data set YAGO2 (yet another great ontology 2). The related information of datasets is shown in Table 3, where #*T*, #*S*, #*O*, and #*P* represent the numbers of triples, subjects, objects, and different predicates, respectively.

5.1.1. Datasets. The synthetic data set LUBM [34] was developed by Lehigh University, which is a standard and systematic semantic Web repository evaluation benchmark for university ontology. This benchmark aims to evaluate the extended queries of a single real ontology on a large data set. The two datasets of different sizes are generated from the data generator UBA 1.7\footnote {<http://swat.cse.lehigh.edu/projects/lubm>}. YAGO2\footnote {<http://yago-knowledge.org>} is a linked data knowledge base that mainly integrates data from three sources: Wikipedia, WordNet, and

GeoNames which contains 120 million triples and more than 10 million entities (such as individuals, organizations, and cities).

5.2. Analysis of Experimental Results. We compare the query performance of our DP-SM algorithm with TriAD [4] and Wukong [5]. The query performances are deployed on six calculated nodes (including a master node) and evaluated on the LUBM-2560 dataset. A benchmark is used to generate different scales of query graph, which is employed in the research of many distributed RDF systems and is published in [12].

Experiments are evaluated into two groups of query graphs, illustrated in Figure 12. The first group of query graph L1, L2, and L3 correspond to Q1, Q3, and Q7, respectively, in [19], our PD-SM algorithm is faster by 1.4–2.2 times than WuKong algorithm. Actually, the final results of L2 are empty. Even though there exists a large number of predicate relationships mapped to the query graph L2, the verified candidates of query nodes are empty. The candidate verification of our algorithm can previously find the candidates of all query nodes before the matching processing is executed, while the algorithms of WuKong and TriAD need to find the candidates with a time-consuming traversal on large intermediate results. The final results of L1 and L3 are conducted as 65,000 and 1,000 data subgraphs, respectively. The experiments verify that the graph-based exploration method has nearly one order of magnitude faster than the relationship-based joining model. A circular-pattern first matching order is employed to guide the iterative processing of subgraph matching in our algorithm, which can prune the redundant intermediate results previously. Thus, our algorithm obtains a greater improvement of matching performance.

The second group of query graphs L4, L5, and L6 corresponds to the extended Q2, Q1, and Q7 in [19], which employs the more complex and denser topological structures than the L1, L2, and L3. The query graph of L4 is a noncircular topological structure and the intermediate results are larger without the verification of partial subgraph isomorphism. Thus, our algorithm DP-SM has a small improvement than TriAD. Compared with algorithm WuKong, the improved matching performance of our algorithm benefits from the strategy for postponing the cluster-connected calculations of Cartesian products. The query graph L6 contains more dense circular topological structures than L5. The circular-pattern first matching strategy can speed up the acquisition of subgraph results.

The average matching time on YAGO2 dataset is evaluated in Figure 13, where the simple and complex query graphs are denoted in [8]. Similar to the experimental evaluation on LUBM dataset, the matching performance of complex query graphs Y4, Y5, and Y6 is similar to the experimental evaluation of complex query graphs on LUBM dataset, illustrated in Figure 13(b). Our algorithm of DP-SM proposed is 1.5–2.5 times faster than algorithms WuKong and TriAD. The difference is that the matching time-performances of simple query graphs Y1, Y2, and Y3

Input: a pattern graph $\mathcal{Q}(\mathcal{V}, \mathcal{E})$
Output: a dominance-partitioned pattern hypergraph $\mathcal{Q}_d(\mathcal{V}_d, \mathcal{E}_d)$

- (1) $i = 0$, $\text{DFS}[u_r] = i$, $\mathcal{V}_d \leftarrow u_r$;
- (2) **For** $u \in u_r.\text{successors}$ **do**
- (3) **If** u is unvisited **then**;
- (4) $\text{DFS}[u] = i++$, Continue to $u_r.\text{successors}$;
- (5) **Else if** u is visited **then**
- (6) $\mathcal{V}_d \leftarrow u$;
- (7) **For** $\text{DFS}[u] = I$ from $|\mathcal{V}_d| - 1$ to 0 **do**
- (8) **For** $u' \in u.\text{precursors}$ **do**
- (9) **If** $\text{DFS}[u] \in \text{DFS}[u']$ **then**
- (10) $\mathcal{P}(u) \leftarrow (u, u')$, Continue to $u'.\text{precursors}$;
- (11) **Else if** $\text{sdom}[u] > \text{sdom}[u']$ **then**
- (12) $\mathcal{P}(u) \leftarrow (u, u')$, $\text{sdom}[u] = \text{sdom}[u']$;
- (13) $\mathcal{P}(u) \leftarrow [\text{sdom}[u], u]$;
- (14) **For** $u \in \mathcal{V}_d$ in descending order of DFS **do**
- (15) **For** $u' \in \mathcal{V}_d - u$ **do**
- (16) **If** $\text{DFS}[u] < \text{DFS}[u'] < \text{DFS}[\text{sdom}[u]]$ **then**
- (17) Removing u from $\mathcal{V}_d$;
- (18) **For** $u \in \mathcal{V}_d$ **do**
- (19) $\mathcal{T}(u) \leftarrow (u, u.\text{precursors})$, $\mathcal{PT}(u) = \mathcal{P}(u) \cup \mathcal{T}(u)$;
- (20) **For** $u \in \mathcal{V}_d$ and $u' \in \mathcal{V}_d - u$ **do**
- (21) **If** $\mathcal{PT}(u) \wedge \mathcal{PT}(u') \neq \emptyset$ **then**
- (22) $\mathcal{E}_d \leftarrow (u, u')$;
- (23) **Return** $\mathcal{Q}_d(\mathcal{V}_d, \mathcal{E}_d)$;

ALGORITHM 2: DPPG-construction algorithm.

Input: a dominant connected pattern graph $\mathcal{Q}_d(\mathcal{V}_d, \mathcal{E}_d)$, an RDF graph $G(V, E, L, \varphi)$
Output: a set of k -partitioned data subgraph $G_k = \{G'_1, \dots, G'_k\}$

- (1) Initializing weighted matrix W of size $|\mathcal{V}_d| \times |\mathcal{V}_d|$;
- (2) **For each** $t \in G$ **do**
- (3) $W[i, i] = \sum_{p \in \mathcal{PT}_i} |\langle p, t \rangle|$;
- (4) $W[i, j] = \sum_{p \in \mathcal{PT}_i \wedge \mathcal{PT}_j} \sum_{t \in G} |\langle p, t \rangle|$;
- (5) **For** $i < |\mathcal{V}_d|$ **do**;
- (6) $D[i, i] = \sum_{j=1}^{|\mathcal{V}_d|} W[i, j]$;
- (7) $f = D^{-(1/2)}(D - W)D^{-(1/2)}$, $F \leftarrow k \times |\mathcal{V}_d|$;
- (8) $C \leftarrow K\text{-Means}(F)$, $C = \{C_1, \dots, C_2\}$;
- (9) **For** $C_i \in C$ and $t \in G$ **do**
- (10) **If** $t \in (C_i)$, **then**
- (11) $G'_k \leftarrow t$;
- (12) **Return** G_k ;

ALGORITHM 3: Dominance-driven k -partition algorithm.

TABLE 2: Dictionary encoding mapping table.

v-id	p/type/attribute		dir
0 index	0 predicate		0 in
1 Person_A	1 advisorBy	9 ResearchAssistant	1 out
2 Person_B	2 teachingOf	10 FullProfessor	
3 Publication_A	3 takesCourse	11 Publication	
4 Department_A	4 memberOf	12 Department	
5 University_A	5 workFor	13 University	
6 Course_A	6 subOf	14 Course	
7 Course_B	7 pubAuthor	15 GraduateCourse	
8 123@163.com	8 rdf:type	16 hasEmail	

```

Input: a query graph  $q(u'_i, \mathcal{V}', \mathcal{E}')$ 
Output: the subgraph mappings  $\mathcal{M}$  of  $q$  on  $G_k$ 
(1) If  $i = |\mathcal{V}'|$  then;
(2)   Output  $M \rightarrow \mathcal{M}$ ;
(3) Else
(4)   If  $i = -1$  then;
(5)     Continue to  $u_i.\text{successor}$ ;
(6) Else
(7)   Foreach  $\langle v, u_i \rangle \in M(u_i, G_k)$  do
(8)     If  $\text{candidateValid}(v, u_i)$  then
(9)        $\langle v, u_i \rangle \rightarrow M$ ;
(10)       $\text{subMatching}(q(u_i.\text{successor}))$ ;
(11)       $\text{subMatching}(q(u_i.\text{precursor}))$ ;

```

ALGORITHM 4: Subgraph matching algorithm on k -partitioned RDF graph ($\text{subMatching}()$).

TABLE 3: Related information of Datasets.

Datasets	#T (M)	#S (M)	#O (M)	#P
YAGO2	120	10	54	99
LUBM-40	5.3	0.8	0.55	17
LUBM-160	21	3.5	2.5	17
LUBM-640	85	14	10	17
LUBM-2560	346	55	41	17

FIGURE 12: The average matching time of algorithms on DataSet LUBM-2560.

are faster than the simplex ones on LUBM datasets, because the nodes of simple query graphs are limited by constant values that can conduct the smaller search space of intermediate results, shown in Figure 13(a). Compared with Wukong and SDSM, since our algorithm has a time-consumption in the orchestration of matching order, it is negligible with the overall running time of matching algorithms.

5.3. Experimental Scalability. The scalability of algorithms is evaluated based on the number of machines and the size of dataset.

The scalability based on the number of machines are evaluated in Figure 14, where the number of machines is gradually increased from 2 to 6. The experimental results show that the matching time-performances of query graph L1, L3, L4, L5, and L6 gradually improved in increasing order of the machine number. The trend of experimental evaluation proved that our DP-SM algorithm can effectively conduct the subgraph results in distributed environments. Since the candidates of L2 are verified as empty in the previous candidate verification, the matching time appears as a constant trend. For complex queries L4, L5, and L6, the decreasing magnitudes in matching time-performances are slightly lower than ones of L1 and L3, because the query

FIGURE 13: The average matching time of algorithms on DataSet YAGO2.

FIGURE 14: Performance of DP-SM algorithm on the number of different cluster nodes.

graphs crossing multiple partitioned pattern subgraphs increase the time-consumption of transmission on the partitioned RDF graphs.

The scalability based on the size of dataset LUBM is evaluated in Figure 15, where the number of machines is fixed as 6. The different scales of LUBM dataset are generated to evaluate the matching time-performances of algorithms, which are located in the range from 5.3 M to 346 M. The matching time-performances of our DP-SM algorithm can maintain a nearly linear growth without the complex topological structures of the query graph. Our algorithm employs a circular-pattern first matching strategy to

previously prune the redundant RDF and postpone the subgraph-connected calculation of Cartesian products. Then, the partial intermediate results are linked slightly without the huge matching time-consumption on noncircular pattern subgraphs.

6. Conclusions

In this paper, we propose a dominance-partitioned subgraph matching on a large RDF graph. Firstly, a dominance-connected pattern graph is extracted from a pattern graph to construct a dominance-partitioned pattern hypergraph,

FIGURE 15: Performance of DP-SM algorithm on different scales of Data set LUBM.

which divides a pattern graph as multiple fish-shaped pattern subgraphs. Secondly, a dominance-driven spectrum clustering strategy is used to gather the pattern subgraphs into multiple clusters. Thirdly, a dominance-partitioned subgraph matching algorithm is designed to conduct all isomorphic subgraphs on a cluster-partitioned RDF graph. Finally, experimental evaluation verifies that our strategy has higher time-efficiency of complex queries, and it has better scalability on multiple machines and different data scales.

Data Availability

The LUBM data used to support the findings of this study have been deposited in the web repository (<http://swat.cse.lehigh.edu/projects/lubm>). The YAGO2 data used to support the findings of this study have been deposited in the web repository (<http://yago-knowledge.org>).

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant no. 61976032).

References

- [1] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, New York, NY, USA, 1979.
- [2] T. Neumann and G. Weikum, "Rdf-3X: a RISC-style engine for RDF," *Proceedings of the VLDB Endowment*, vol. 1, no. 1, pp. 647–659, 2008.
- [3] C. Weiss, P. Karras, and A. Bernstein, "Hexastore: sextuple indexing for semantic web data management," *Proceedings of the VLDB Endowment*, vol. 1, no. 1, pp. 1008–1019, 2008.
- [4] S. Gurajada, S. Seufert, I. Miliaraki, and M. Theobald, "TriAD: a distributed shared-nothing RDF engine based on asynchronous message passing," in *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, pp. 289–300, Snowbird, UT, USA, June 2014.
- [5] J. Shi, Y. Yao, R. Chen, H. Chen, and F. Li, "Fast and concurrent RDF queries with RDMA-based distributed graph exploration," in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation*, pp. 317–332, Savannah, GA, USA, November 2016.
- [6] L. He, B. Shao, Y. Li et al., "Stylus: a strongly-typed store for serving massive RDF data," *Proceedings of the VLDB Endowment*, vol. 11, no. 2, pp. 203–216, 2017.
- [7] Q. Xu, X. Wang, J. Li, Y. Gan, L. Chai, and J. Wang, "StarMR: an efficient star-decomposition based query processor for SPARQL basic graph patterns using MapReduce," in *Proceedings of the Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data*, pp. 415–430, Macau, China, July 2018.
- [8] X. Wang, L. Chai, Q. Xu et al., "Efficient subgraph matching on large RDF graphs using MapReduce," *Data Science and Engineering*, vol. 4, no. 1, pp. 24–43, 2019.
- [9] F. Bi, L. Chang, X. Lin, L. Qin, and W. Zhang, "Efficient subgraph matching by postponing cartesian products," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 1199–1214, San Francisco, CA, USA, June 2016.
- [10] J. Z. Pan, "Resource description framework," *Handbook on Ontologies*, pp. 71–90, Springer, Berlin, Germany, 2019.
- [11] D. J. Abadi, A. Marcus, S. R. Madden, and K. Hollenbach, "SW-store: a vertically partitioned DBMS for semantic web data management," *The VLDB Journal*, vol. 18, no. 2, pp. 385–406, 2009.
- [12] T. Neumann and G. Weikum, "The RDF-3X engine for scalable management of RDF data," *The VLDB Journal*, vol. 19, no. 1, pp. 91–113, 2010.
- [13] P. Peng, L. Zou, L. Chen, and D. Zhao, "Adaptive distributed RDF graph fragmentation and allocation based on query workload," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 4, pp. 670–685, 2019.
- [14] J. Huang, D. J. Abadi, and K. Ren, "Scalable SPARQL querying of large RDF graphs," *Proceedings of the VLDB Endowment*, vol. 4, no. 11, pp. 1123–1134, 2011.

- [15] K. Rohloff and R. E. Schantz, "Clause-iteration with Map-Reduce to scalably query datagraphs in the SHARD graph-store," in *Proceedings of the ACM 4th International Workshop on Data-Intensive Distributed Computing*, pp. 35–44, San Jose, CA, USA, June 2011.
- [16] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 359–392, 1998.
- [17] L. Lai, L. Qin, X. Lin, Y. Zhang, L. Chang, and S. Yang, "Scalable distributed subgraph enumeration," *Proceedings of the VLDB Endowment*, vol. 10, no. 3, pp. 217–228, 2016.
- [18] A. Schätzle, M. Przyjaciół-Zablocki, S. Skilevic, and G. Lausen, "S2Rdf: RDF querying with SPARQL on spark," *Proceedings of the VLDB Endowment*, vol. 9, no. 10, pp. 804–815, 2016.
- [19] M. Atre, V. Chaoji, M. J. Zaki, and J. A. Hendler, "Matrix bit loaded: a scalable lightweight join query processor for RDF data," in *Proceedings of the ACM 19th International Conference on World Wide Web*, pp. 41–50, Raleigh, NC, USA, April 2010.
- [20] P. Yuan, P. Liu, B. Wu, H. Jin, W. Zhang, and L. Liu, "TripleBit: fast and compact system for large scale RDF data," *Proceedings of the VLDB Endowment*, vol. 6, no. 7, pp. 517–528, 2013.
- [21] L. Zou, J. Mo, L. Chen, M. T. Özsu, and D. Zhao, "gStore: answering SPARQL queries via subgraph matching," *Proceedings of the VLDB Endowment*, vol. 4, no. 8, pp. 482–493, 2011.
- [22] L. Zeng and L. Zou, "Redesign of the gStore system," *Frontiers of Computer Science*, vol. 12, no. 4, pp. 1–19, 2018.
- [23] K. Zeng, J. Yang, H. Wang, B. Shao, and Z. Wang, "A distributed graph engine for web scale RDF data," *Proceedings of the VLDB Endowment*, vol. 6, no. 4, pp. 265–276, 2013.
- [24] J. Xu, J. Chen, and L. Yin, "Multi-objective spatial keyword query with semantics: a distance-owner based approach," *Distributed and Parallel Databases*, vol. 38, no. 3, pp. 625–647, 2020.
- [25] Z. Wang, Y. Yuan, X. Zhou, and H. Qin, "Effective and efficient community search in directed graphs across heterogeneous social networks," in *Proceedings of the 31st Australasian Database Conference*, pp. 161–172, Melbourne, Australia, February 2020.
- [26] X. Chen, J. Xu, R. Zhou et al., "S2R-tree: a pivot-based indexing structure for semantic-aware spatial keyword search," *GeoInformatica*, vol. 24, no. 1, pp. 3–25, 2020.
- [27] Y. Cheng, L. Chen, Y. Yuan, and G. Wang, "Rule-based graph repairing: semantic and efficient repairing methods," in *Proceedings of the 34th IEEE International Conference on Data Engineering*, pp. 773–784, Paris, France, April 2018.
- [28] Y. Cheng, Y. Yuan, L. Chen, G. Wang, C. Giraud-Carrier, and Y. Sun, "DistR: a distributed method for the reachability query over large uncertain graphs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 11, pp. 3172–3185, 2016.
- [29] Y. Hu, T. Nie, D. Shen, and G. Yu, "A deep aggregated model for protein secondary structure prediction," *International Journal of Data Mining and Bioinformatics*, vol. 22, no. 3, pp. 231–249, 2019.
- [30] T. Cai, J. Li, A. S. Mian, R. Li, T. Sellis, and J. X. Yu, "Target-aware holistic influence maximization in spatial social networks," *IEEE Transactions on Knowledge and Data Engineering*, no. 99, p. 1, 2020.
- [31] J. Li, T. Cai, K. Deng, X. Wang, T. Sellis, and F. Xia, "Community-diversified influence maximization in social networks," *Information Systems*, vol. 92, Article ID 101522, 2020.
- [32] Y. Sun, G. Li, J. Du, B. Ning, and H. Chen, "A subgraph matching algorithm based on subgraph index for knowledge graph," *Frontiers of Computer Science*, 2020.
- [33] T. Lengauer and R. E. Tarjan, "A fast algorithm for finding dominators in a flowgraph," *ACM Transactions on Programming Languages and Systems*, vol. 1, no. 1, pp. 121–141, 1979.
- [34] Y. Guo, Z. Pan, and J. Heflin, "LUBM: a benchmark for OWL knowledge base systems," *Journal of Web Semantics*, vol. 3, no. 2-3, pp. 158–182, 2005.