International Journal of Geo-Information

MDPI

Article

Spatiotemporal RDF Data Query Based on Subgraph Matching

Xiangfu Meng [1], Lin Zhu [1,2,*], Qing Li [2] and Xiaoyan Zhang [1]

[1] School of Electronic and Information Engineering, Liaoning Technical University, Huludao 125105, China; mengxiangfu@lntu.edu.cn (X.M.); zhangxiaoyan@lntu.edu.cn (X.Z.)
[2] School of Computer and Communication Engineering, Northeastern University (Qinhuangdao), Qinhuangdao 066004, China; liqingneuq@hotmail.com
* Correspondence: zhulin@neuq.edu.cn; Tel.: +86-131-3356-8068

Abstract: Resource Description Framework (RDF), as a standard metadata description framework proposed by the World Wide Web Consortium (W3C), is suitable for modeling and querying Web data. With the growing importance of RDF data in Web data management, there is an increasing need for modeling and querying RDF data. Previous approaches mainly focus on querying RDF. However, a large amount of RDF data have spatial and temporal features. Therefore, it is important to study spatiotemporal RDF data query approaches. In this paper, firstly, we formally define spatiotemporal RDF data, and construct a spatiotemporal RDF model st-RDF that is used to represent and manipulate spatiotemporal RDF data. Secondly, we present a spatiotemporal RDF query algorithm stQuery based on subgraph matching. This algorithm can quickly determine whether the query result is empty for queries whose temporal or spatial range exceeds a specific range by adopting a preliminary query filtering mechanism in the query process. Thirdly, we propose a sorting strategy that calculates the matching order of query nodes to speed up the subgraph matching. Finally, we conduct experiments in terms of effect and query efficiency. The experimental results show the performance advantages of our approach.

Keywords: subgraph matching; RDF; spatiotemporal data; query

Citation: Meng, X.; Zhu, L.; Li, Q.; Zhang, X. Spatiotemporal RDF Data Query Based on Subgraph Matching. *ISPRS Int. J. Geo-Inf.* **2021**, *10*, 832. https://doi.org/10.3390/ijgi10120832

Academic Editor: Wolfgang Kainz

Received: 28 September 2021
Accepted: 10 December 2021
Published: 12 December 2021

1. Introduction

Resource Description Framework (RDF), as a standard metadata description framework proposed by the World Wide Web Consortium (W3C) [1], can be used in various fields such as intelligent software agents, privacy preferences, and privacy policies. Based on the advantages of RDF in data and knowledge representation, many researchers have proposed to use RDF in the management of temporal data in recent years. In the representation of temporal data, some researchers have tried to introduce the time dimension into standard RDF or add time stamp information after the predicate or the entire triplet [2], to achieve RDF-based temporal data modeling. Gutierrez et al. [3,4] propose a framework that incorporates temporal reasoning into RDF to generate temporal RDF graphs, and at the same time provides a semantic for these graphs, including a grammar that uses RDF vocabulary and time labels to integrate the grammatical framework into standard RDF graphs. Since then, in response to the query requirements of temporal RDF data, Pugliese et al. [5] propose the tGRIN index structure, which establishes a special index for the temporal RDF physically stored in the RDBMS. Motik [6] raises a logic-based method to represent the effective time in RDF and OWL and devises a query evaluation algorithm. Recently, Zhang et al. [7] put forward a new RDF-based temporal data representation model, RDFt, and its query method. This model is suitable for querying temporal data in practical applications.

There are information and datasets in the real world having spatial attributes such as geographic location [8]. Kolas et al. [9] come up with ontology types that can support geographic spatial semantics systems in order to realize the analysis and query operations

of spatial data and expound the motivation of each ontology type and the potential areas of geospatial community standardization. Smart et al. [10] use qualitative spatial reasoning to support geographic ontology management system development tools, and introduce the realization of the spatial reasoning engine, where the Web ontology language (OWL) and its associated reasoning tools are applied, and the space rules engine extension of the OWL associated reasoning tool is used to represent spatial reasoning and integrity rules. Subsequently, Batsakis and Petrakis [11,12] propose SOWL, demonstrating how spatial and spatiotemporal information and spatiotemporal evolution can be effectively represented in OWL. Recently, Ademaj et al. [13] introduce a novel spatial consistency model that applies to all geometrically based popular stochastic channel models. Cui et al. [14] focus on modeling the spatial point process of random vehicle positions in large and small cities, and experimentally verify the real location data of moving taxi tracks recorded by the global positioning system (GPS), thus forming a spatial RDF model.

With the development of temporal data models and spatial data models, some researchers have begun to consider integrating temporal attributes and spatial attributes, and are committed to constructing spatiotemporal RDF data models that can simultaneously represent temporal and spatial information. Some relevant departments provide real RDF data sets integrating time information and spatial information. These data sets include YAGO2, OpenStreetMap, and GovTrack, where YAGO2 is an RDF dataset based on Wikipedia and WordNet (Semantic Web) [15,16]. At the same time, Koubarakis and Kyzirakos [17] developed a constrained data model stRDF, which has the ability to represent spatial and temporal data. Wang et al. [18,19] designed a spatiotemporal data model of spatiotemporal RDF quintuples, and spatiotemporal RDF data can be organized in the form of graphs containing spatiotemporal features based on the model. Lu et al. [20] take the fuzziness of the data into account and conceived a novel fuzzy spatiotemporal data model DPS. Xu et al. [21] constituted a Bayesian spatiotemporal stochastic model to fully explain the spatiotemporal correlation in RSI. Sun et al. [22] proposed a framework for unifying geospatial data, which can effectively organize geospatial data. Kyzirakos et al. [23] proposed a new RDF store that supports the state of the art semantic geospatial query languages stSPARQL and GeoSPARQL.

The work of querying spatiotemporal RDF data has also received widespread attention, apart from constructing a spatiotemporal RDF data model to realize the representation function of spatiotemporal data. Vlachou et al. [24] compressed the spatiotemporal information of each RDF entity into a unique integer value and use this code in the filtering and optimization framework to effectively query spatiotemporal RDF data. Wu et al. [25] proposed a keyword-based spatiotemporal RDF data query method kSPT, which does not rely on the use of a structured query language. Eom et al. [26] used the method of constructing an index to query the spatiotemporal data. However, since the spatiotemporal RDF data model is modeled based on the RDF data model, it can also be represented and stored in the form of graphs. With the rapid development of graph query algorithms based on subgraph matching, it is necessary to clarify the concepts of RDF graph mapping, RDF graph equivalence, and graph isomorphism [8]. The representative subgraph matching algorithms include VF2 [27], Quicks I [28], GraphQL [29], GADDI [30], SPATH [31], and TurbOiso [32]. In order to improve query performance, Lee et al. [33] constructed a general framework of a subgraph matching algorithm.

Although current efforts have advantages of querying RDF data, they mainly focus on temporal RDF or spatial RDF. In order to model and query spatiotemporal RDF data, not only do their respective temporal and spatial features need to be considered, but the overall spatiotemporal features over time also need to be considered, which is not a straightforward task. Consequently, a range of research for spatiotemporal RDF data management is investigated. However, they mainly focus on spatiotemporal RDF data representations formally, or they are not well compatible with efficient querying. The existing spatiotemporal models or standard vocabularies dedicated to spatiotemporal data will generate redundant information when querying spatiotemporal RDF data based

on subgraph matching. In this paper, we propose st-RDF model firstly. We introduce a temporal component to RDF datasets, and after a spatial component to RDF datasets, which are finally mixed in a spatiotemporal component. An illustrative example is shown after the corresponding definitions and figures. The temporal component is defined as an interval and the spatial component is defined as a pair of coordinates (rectangles). After, a spatiotemporal query graph is defined and illustrated with an example. Two operations are defined for intervals and coordinates: intersection and merging. Intersection for intervals is defined as usual, and merging of intervals is defined as the smallest interval contains the merged internals. The same can be said for coordinates. The intersection of rectangles and the smallest rectangle containing the merged rectangles. On the basis of the proposed model, we propose a subgraph matching approach for querying spatiotemporal RDF data. Experimental results verify that our approach has performance advantages, and application discussions give general steps on how to use our approach in spatiotemporal applications. The main contributions of this paper are described as follows:

- We construct a spatiotemporal RDF model st-RDF. On the basis of the data model, it can represent and operate data with temporal and spatial attributes.
- We propose a preliminary query filtering mechanism. For queries whose spatiotemporal range exceeds the data graph, this mechanism can quickly determine that the query result is empty and we feed it back to the user.
- We propose a spatiotemporal RDF query algorithm stQuery based on subgraph matching. At the same time, we sort nodes according to the degree of association between query nodes and candidate nodes.
- We use the control variable method to compare stQuery with the other three query algorithms to test the query efficiency.

The rest of this paper is organized as follows. We propose definitions and concepts about the spatiotemporal RDF model st-RDF in Section 2. In Section 3, a spatiotemporal RDF query algorithm stQuery based on subgraph matching is proposed. We show the performance of our algorithm through experimental design and result analysis in Section 4. Section 5 gives application discussions and Section 6 draws conclusions and explains future research work.

2. Spatiotemporal Data Model Based on RDF

This section introduces the temporal and spatial features of spatiotemporal data and proposes a spatiotemporal RDF data model. The representative features of spatiotemporal data are spatial feature and temporal feature, and the most representative feature of spatial feature is coordinate feature. For simplicity, we only consider latitude and longitude as spatial features. Of course, there are several other spatial features such as area and other geometries, but such features can be easily extended by our representation. On the other hand, our subgraph matching on spatiotemporal RDF data mainly relates to locations. As a result, in this paper, we mainly consider latitude and longitude to space.

2.1. The Temporal Feature of Spatiotemporal Data

Time is closely related to our lives, and there are a lot of instances with temporal features around us at the same time. Instances with temporal attributes include temporal entities and temporal facts. We focus on the following four types of temporal entities: person, groups, artifact, and events.

(i) Person: uses the temporal predicates “wasBornOnDate” and “diedOnDate” to identify the temporal attributes.
(ii) Groups: (bands, football clubs, companies, etc.). Use the temporal predicates “wasCreatedOnDate” and “wasDestroyedOnDate” to identify the temporal attributes.
(iii) Artifact: (buildings, songs, cities, etc.). Use the temporal predicates “wasCreatedOnDate” and “wasDestroyedOnDate” to identify the temporal attributes.

(iv) Events: (sports events, wars, etc.). Use the temporal predicates "startedOnDate" and "endedOnDate" to identify the temporal attribute.

We combine the temporal predicates "wasBornOnDate", "wasCreateOnDate" and "startedOnDate" into the temporal predicate "startsExistingOnDate". The temporal predicates "DiedOnDate", "WasdestroyOnDate" and "EndEndOnDate" are united into the temporal predicate "EndSExistingOnDate". Furthermore, the temporal predicate "beginning existence" and the temporal predicate "ending existence" can be merged to obtain the valid existence temporal predicate "hasExistingTemporal " of the temporal entity, and the corresponding object is usually a valid temporal period. The above process is described in detail in Figure 1.

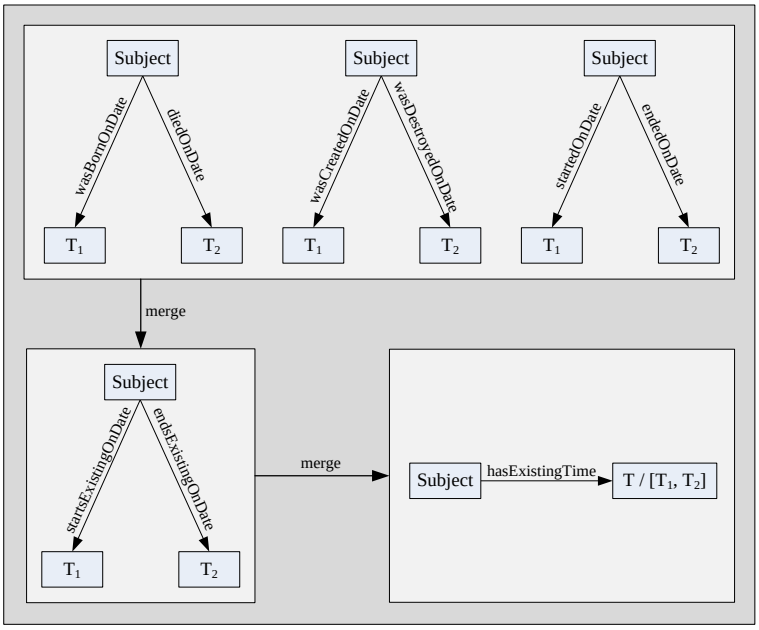


Figure 1. Representation model of temporal entities.

Temporal entities can be represented by a special RDF triplet et-RDF. Definition 1 introduces the temporal entity model et-RDF.

Definition 1 (et-RDF). *The et-RDF triple is denoted as* $(s, p^T, o^T) \in (E \cup B) \times U \times (T \cup B)$, *where:*

- s stands for subject, p^T stands for temporal predicate, and o^T stands for temporal object;
- E is the entity set and $E \subset U$, B stands for the blank node set, and T stands for the temporal label set.

In Definition 1, T can be further expressed as temporal interval $[T_S, T_E]$, where T_S represents the starting existence time, T_E represents the ending existence time, and $T_S \leq T_E$, $T \subset L$. The et-RDF triple can be represented as a directed graph with two nodes and an edge. An et-RDF triple instance is represented in the purple elliptical dashed line as shown in Figure 2. The yellow node ([1936-05-17,2013-02-02]) represents the temporal entity, and the information inside the node stands for the entity information. The blue node (Juan_B_Tudela) stands for the temporal object, the information inside the node stands for the temporal label, and the temporal predicate is represented by a directed arc from the temporal entity to the temporal object, including the temporal attribute marked on the directed arc. For example, the location information of Saipan is 15.21233 longitude and 145.7545 latitude; Juan_B._Tudela's lifetime is from 1936-05-17 to 2013-02-02; Juan_B._Tudela has a name from 1936-05-17 to 2013-02-02.

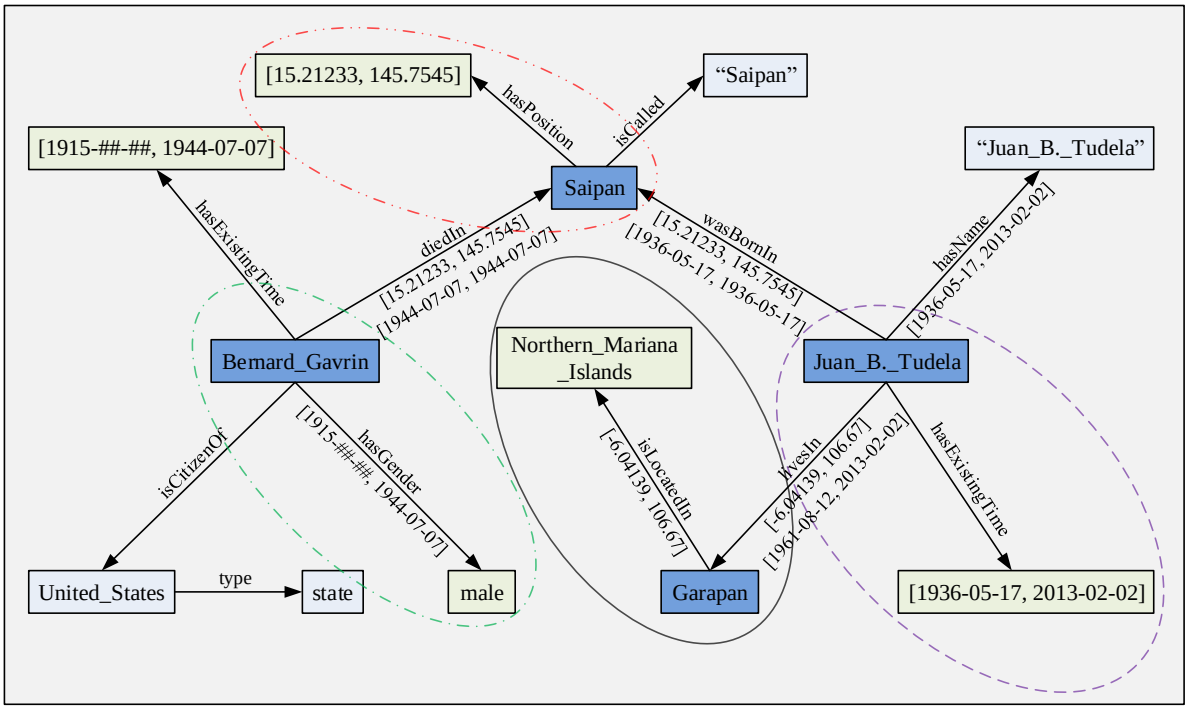


Figure 2. Spatiotemporal RDF data graph.

A fact can be represented in the form of an RDF triple. Similarly, the temporal fact can be represented in the form of the corresponding RDF triple with temporal information. We define the concept of temporal fact model ft-RDF.

Definition 2 (ft-RDF). *The ft-RDF triplet with a temporal label is denoted as (s, p, o) [T], where:*

- *s* stands for subject, *p* stands for predicate, *o* stands for object, and $(s, p, o) \in (U \cup B) \times U \times (U \cup B \cup L)$ stands for a fact;
- T represents the temporal label.

According to Definition 2, T is represented by the temporal interval $[T_S, T_E]$, where T_S represents the starting time, T_E represents the ending time, and $T_S \leq T_E$. In particular, $[T_S, T_E]$ represents a moment when $T_S = T_E$. The ft-RDF triples can be converted into graphs. As shown in Figure 2, there is an instance of a ft-RDF triple in the green elliptical dotted line. The yellow node (male) represents the subject of the statement, and the information within the node is the label of the subject. The blue node (Bernard_Gavrin) represents the object of the statement, and the information within the node is the label of the object. The predicate in this statement is represented by a directed arc from the subject to the object, and the predicate label is marked on the directed arc. The temporal attribute is the attribute of the whole statement. In the ft-RDF graph, the corresponding temporal label is marked on the directed arc together with the predicate label.

2.2. Spatial Features of Spatiotemporal Data

Each physical object has a position on the earth, and we represent their spatial attributes with the position information. The same as temporal attributes, instances with spatial attributes also include spatial entities and spatial facts. The latitude and longitude coordinates of the physical objects on the earth are used to identify the spatial features.

In terms of spatial entities, spatial predicates "hasLatitude" (to identify the latitude coordinates) and "hasLongitude" (to identify the longitude coordinates) are used to identify the spatial attributes. The spatial predicates "hasLatitude" and "hasLongitude" are merged to obtain the geographic spatial predicate "hasPosition" of the spatial entity in order to represent the spatial information of the spatial entity more effectively. The corresponding

object is a geographic coordinate represented by longitude and latitude. This process is described in detail in Figure 3. Definition 3 describes the spatial entity model es-RDF.

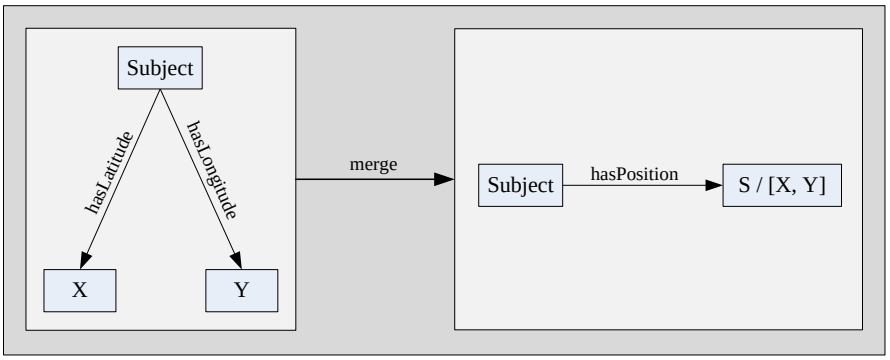


Figure 3. Representation model of spatial entities.

Definition 3 (es-RDF). *An es-RDF triplet is denoted as* $(s, p^S, o^S) \in (E \cup B) \times U \times (S \cup B)$*, where:*

- s stands for subject, p^S stands for spatial predicate, and o^S stands for spatial object;
- E stands for the set of entities and $E \subset U$, B stands for the set of blank nodes, S stands for the set of spatial labels.

As for Definition 3, an es-RDF triple can be represented as a directed graph with two nodes and one edge, as shown in Figure 2. An instance of an es-RDF triple is shown in the red elliptical double-dotted line. The yellow node ([15.21233,145.7545]) represents the spatial entity, and the information inside the node is the entity information. The blue node (Saipan) represents the spatial object, and the information inside the node is the spatial label. The spatial predicate is represented by a directed arc from the spatial entity to the spatial object, and the spatial attribute is marked on the directed arc.

Spatial facts can be represented as a statement with spatial attributes. Similarly, a spatial fact can be represented by an RDF triple with spatial attributes to represent the geo-location information.

Definition 4 (fs-RDF). *A fs-RDF triple with a spatial label is denoted as (s, p, o) [S], where:*

- s stands for subject, p stands for predicate, and o stands for object, where $(s, p, o) \in (U \cup B) \times U \times (U \cup B \cup L)$;
- S represents the spatial label.

According to Definition 4, S is represented by the longitude and latitude coordinates $[X, Y]$, where X represents the longitude value and Y represents the latitude value. The fs-RDF triples can also be converted into graphs. An fs-RDF triple is represented by a directed graph with two nodes, where spatial attributes are attached to a directed arc between the two nodes. As shown in Figure 2, there is an instance of a fs-RDF triple in the black elliptical solid line. The blue node (Garapan) represents the subject of the statement, and the information within the node is the label of the subject. The yellow node (Northern_Mariana_Islands) represents the object of the statement, and the information within the node is the label of the object. The predicate in this statement is represented by a directed arc from the subject to the object, and the predicate label is marked on the directed arc. The spatial attribute is the attribute of the entire statement. In the fs-RDF graph, the corresponding spatial label is labeled along with the predicate label on the directed arc.

2.3. The Spatiotemporal RDF Data Model

In this subsection, we merge et-RDF, ft-RDF, es-RDF, and fs-RDF models to form a spatiotemporal RDF data model st-RDF, which can represent temporal and spatial attributes simultaneously.

Definition 5 (st-RDF). *A spatiotemporal RDF triple with a temporal label and a spatial label is denoted as (s, p, o) [S] [T], where:*

- s stands for subject, p stands for predicate, and o stands for object, where $(s, p, o) \in (U \cup B) \times U \times (U \cup B \cup L)$;
- S represents the spatial label;
- T represents the temporal label.

In Definition 5, p contains temporal predicates, spatial and ordinary predicates. o contains temporal objects, spatial and ordinary objects. S is represented by the longitude and latitude coordinates $[X, Y]$, where X represents the longitude value and Y represents the latitude value. T is represented by the temporal interval $[T_S, T_E]$, where T_S represents the starting time, T_E represents the ending time, and $T_S \leq T_E$. Therefore, the spatiotemporal RDF triplet can also be denoted as (s, p, o) $[X, Y]$ $[T_S, T_E]$. In st-RDF, the information in "[" and "]" can be omitted according to whether the triple (i.e., the corresponding statement) has spatiotemporal attributes. If the triple does not have a temporal attribute, then $[T]$ is null. Similarly, if the triple does not have a spatial attribute, then we have $[S]$ is null.

A spatiotemporal RDF graph is a directed graph composed of multiple spatiotemporal RDF triples, in which the subject s and object o are presented in the form of nodes, and the predicate p is represented as a directed arc that s points to o. When the triples have spatiotemporal attributes, the spatiotemporal attributes of the triples are attached to the directed arc. When the entity has a spatiotemporal attribute, it is represented by a new triple, in which the subject is the entity, the predicate is the spatiotemporal predicate, and the object is the spatiotemporal information. For the query of spatiotemporal RDF data, we further divide the spatiotemporal RDF graph into a spatiotemporal RDF data graph and spatiotemporal RDF query graph. Definition 6 and Definition 7 define the concepts of spatiotemporal RDF data graph and spatiotemporal RDF query graph.

Definition 6 (spatiotemporal RDF data graph). *An RDF data graph is denoted as stG = (V, E, L, F_{st}), where:*

- $V = V_L \cup V_E \cup V_C \cup V_B \cup V_S \cup V_T$ represents a set of vertices;
- E represents the set of edges between two nodes;
- $L = L_V \cup L_E$ is the label set of all vertices and edges;
- F_{st}: $V \cup E \rightarrow L$ is the mapping function from vertices and edges to the label set L.

In Definition 6, V_L, V_E, V_C and V_B represent text vertex, entity vertex, class vertex and blank vertex, respectively. V_S represents space node and V_T represents temporal node. L_V = {URI} ∪ {Literal Value} ∪ L_T ∪ L_S is as the set of all vertex labels, and $L_E = L_R \cup L_T \cup L_S$ is as the label set of all edge labels, where L_R is as the label set of ordinary relational predicate, L_T is as the temporal label set, and L_S is as the spatial label set. $F_{st}(V)$: $V \rightarrow L_V$ is the mapping function from vertices to vertex labels, and $F_{st}(E)$: $E \rightarrow L_E$ is the mapping function from edges to edge labels. For different types of vertices, the mapping relationship is as follows:

(i) $v \in V_L \Leftrightarrow F_{st}(v) \in$ {Literal Value};
(ii) $v \in V_E \cup V_C \Leftrightarrow F_{st}(v) \in$ {URI};
(iii) $v \in V_B \Leftrightarrow F_{st}(v) =$ NULL;
(iv) $v \in V_S \Leftrightarrow F_{st}(v) \in$ {Spatial Label};
(v) $v \in V_T \Leftrightarrow F_{st}(v) \in$ {Temporal Label}.

Table 1 shows 12 spatiotemporal RDF triples, containing Id, subject, predicate, object, position, and time. The corresponding spatiotemporal RDF data graph is shown in Figure 2.

Table 1. Spatiotemporal RDF data.

Id	Subject	Predicate	Object	Position	Time
1	Bernard_Gavrin	hasExistingTime	[1915-##-##, 1944-07-07]		
2	Bernard_Gavrin	hasGender	male		[1915-##-##, 1944-07-07]
3	Bernard_Gavrin	diedIn	Saipan	[15.21233, 145.7545]	[1944-07-07, 1944-07-07]
4	Bernard_Gavrin	isCitizenOf	United_States		
5	United_States	type	state		
6	Saipan	hasPosition	[15.21233, 145.7545]		
7	Saipan	isCalled	"Saipan"		
8	Juan_B._Tudela	hasExistingTime	[1936-05-17, 2013-02-02]		
9	Juan_B._Tudela	wasBornIn	Saipan	[15.21233, 145.7545]	[1936-05-17, 1936-05-17]
10	Juan_B._Tudela	hasName	"Juan_B._Tudela"		[1936-05-17, 2013-02-02]
11	Juan_B._Tudela	livesIn	Garapan	[−6.04139, 106.67]	[1961-08-12, 2013-02-02]
12	Garapan	isLocatedIn	Northern_Mariana _Islands	[−6.04139, 106.67]	

Definition 7 (spatiotemporal RDF query graph). *The spatiotemporal RDF query graph is denoted by* $stQ = (V^q, E^q, L^q, F_{st}{}^q)$*, where:*

- $V^q = V_L \cup V_E \cup V_C \cup V_B \cup V_S \cup V_T \cup V_P$ represents a set of vertices;
- E^q represents the set of edges between two nodes;
- $L^q = L_V \cup L_E$;
- $F_{st}{}^q$: $V^q \cup E^q \rightarrow L^q$ is a mapping function from vertices and edges to a set of labels L^q.

According to Definition 7, the meanings of V_L, V_E, V_C, V_B, V_S, V_T and L^q are the same as those in definition 6, and V_P represents the parameter vertices in the RDF query graph. $F_{st}{}^q(V^q)$: $V^q \rightarrow L_V$ is a mapping function from vertices to vertex labels, and $F_{st}{}^q(E^q)$: $E^q \rightarrow L_E$ is the mapping function from edges to edge labels. For different types of vertices, the mapping relationship is the same as the spatiotemporal RDF query graph, except $v \in V_P \Leftrightarrow F_{st}{}^q(v) = \text{NULL}$.

It is noted that the variable "V" can appear at any position in the query graph but cannot appear in the relevant temporal period or coordinates.

Figure 4 shows an example of a spatiotemporal RDF query graph that contains a total of seven vertices, three of which are parametric vertices. "$?x$" died in Saipan at a position of 15.21233 longitude and 145.7545 latitude; Saipan has a position "$?z$" and is called "Saipan".

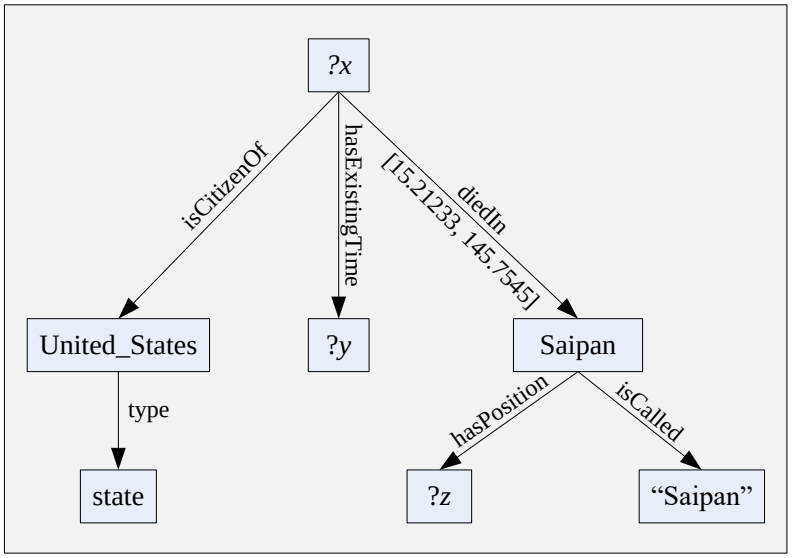


Figure 4. Spatiotemporal RDF query graph.

3. Spatiotemporal RDF Data Query Based on Subgraph Matching

Based on the spatiotemporal RDF data model proposed in the previous section, this section proposes a spatiotemporal RDF data query approach based on subgraph matching.

3.1. Preliminary Spatiotemporal Determination

The matching degree of related spatiotemporal intervals is determined firstly when performing subgraph matching, i.e., the degree of spatiotemporal matching between the query graph and the data graph is determined. The temporal interval matching and spatial interval matching are introduced below.

3.1.1. Temporal Interval Matching

In terms of temporal interval matching, τ is the temporal interval function, and $\tau(stG)$ is the temporal span of stG in the spatiotemporal RDF graph. Accordingly, $\tau(e)$ is the temporal span of the spatiotemporal RDF triples, where $e \in \{e \mid e \in E \text{ in } stG\}$. Each $e \in E$ has $\tau(e) \subset \tau(stG)$. For the temporal relationships between the spatiotemporal RDF data graph and the spatiotemporal RDF query graph, the concepts of temporal intersection operation, temporal merger operation, and temporal span are given.

Definition 8 (temporal intersection operation $\wedge_t$). *Let $t_s = Max(t_{s1}, t_{s2})$ and $t_e = Min(t_{e1}, t_{e2})$ be temporal segments $[t_{s1}, t_{e1}]$ and $[t_{s2}, t_{e2}]$. If and only if $t_s \leq t_e$, the intersecting operation of the temporal segments is $[t_{s1}, t_{e1}] \wedge_t [t_{s2}, t_{e2}] = [t_s, t_e]$. Otherwise, the intersecting operation is $[t_{s1}, t_{e1}] \wedge_t [t_{s2}, t_{e2}] = \emptyset$, when $t_s > t_e$.*

In Definition 8, if there is the following temporal segment T_1 = [1961-02-05, 1982-06-10], T_2 = [1970-07-05, 1992-06-15], and T_3 = [1989-10-03, 1992-06-15], then

(i) $T_1 \wedge_t T_2$ = [1961-02-05, 1982-06-10] $\wedge_t$ [1970-07-05, 1992-06-15] = [1970-07-05, 1982-06-10];
(ii) $T_1 \wedge_t T_3$ = [1961-02-05, 1982-06-10] $\wedge_t$ [1989-10-03, 1992-06-15] = $\emptyset$.

Definition 9 (temporal merger operation $\vee_t$). *Let $t_s = Min(t_{s1}, t_{s2})$ and $t_e = Max(t_{e1}, t_{e2})$ be temporal segments $[t_{s1}, t_{e1}]$ and $[t_{s2}, t_{e2}]$, then the intersecting operation of the temporal segments is $[t_{s1}, t_{e1}] \vee_t [t_{s2}, t_{e2}] = [t_s, t_e]$, where $t_s \leq t_e$.*

According to Definition 9, for the temporal segments T_1 = [1961-02-5, 1982-06-10] and T_2 = [1970-07-05, 1992-06-15], there is a union $T_1 \vee_t T_2$ = [1961-02-05, 1982-06-10] $\vee_t$ [1970-07-05, 1992-06-15] = [1961-02-05, 1992-06-15].

Definition 10 (temporal span τ). *There is a temporal span $\tau(stG) = \{ [t_{si}, t_{ei}] \mid 1 \leq i \leq |E|, t_{si} \leq t_{ei} \}$ for the spatiotemporal RDF graph stG. Let $t_s = Min_{1 \leq i \leq |E|}(t_{si})$ and $t_e = Max_{1 \leq i \leq |E|}(t_{ei})$, then the temporal span of spatiotemporal RDF graph stG is $[t_{s1}, t_{e1}] \vee_t [t_{s2}, t_{e2}] \vee_t \ldots \vee_t [t_{sn}, t_{en}] = [t_s, t_e]$, where $n = |E|$, and $t_s \leq t_e$.*

As for Definition 10, there is temporal span $\tau(stG)$ = [1944-07-07, 1944-07-07] $\vee_t$ [1936-05-17, 1936-05-17] $\vee_t$ [1936-05-17, 2013-02-02] $\vee_t$ [1915-##-##, 1944-07-07] $\vee_t$ [1961-08-12, 2013-02-02] = [1915-##-##, 2013-02-02] for the spatiotemporal RDF data graph *stG* given in Figure 2. For the spatiotemporal RDF query graph *stQ* given in Figure 4, if the temporal attributes of all spatiotemporal RDF triples are empty, the temporal span $\tau(stQ)$ is an infinite set *T*. The span indicates the merging operation and from the set of intervals and rectangles of data or query graph.

For a spatiotemporal RDF data graph *stG* and a spatiotemporal RDF query graph *stQ*, it is possible for the *stQ* to have a matching subgraph in *stG* when $\tau(stG) \wedge_t \tau(stQ)$ is not null. Otherwise, *stQ* must have no matching subgraph in *stG*. Considering the spatiotemporal RDF data graph *stG* in Figure 2 and the spatiotemporal RDF query graph *stQ* in Figure 4, there is $\tau(stG) \wedge_t \tau(stQ)$ = [1915-##-##, 2013-02-02] $\wedge_t T$ = [1915-##-##, 2013-02-02]. If the result is not empty, it is preliminarily determined that the spatiotemporal RDF query graph *stQ* is likely to find a matching subgraph in the spatiotemporal RDF data graph *stG*.

3.1.2. Spatial Interval Matching

For spatial interval matching, ρ is the spatial interval function, and $\rho(stG)$ is the spatial span of the spatiotemporal RDF data graph. Accordingly, $\rho(e)$ represents the spatial span of spatiotemporal RDF triples, where $e \in \{e \mid e \in E \text{ in } stG\}$, and each $e \in E$ has $\rho(e) \subset \rho(stG)$. In terms of the spatial relationship between the spatiotemporal RDF data graph and the spatiotemporal RDF query graph, the concepts of the spatial intersection operation, spatial merger operation, and spatial span are defined.

As shown in Figure 5a, the spatial coordinates are denoted as $A(P_{xA}, P_{yA})$ and $B(P_{xB}, P_{yB})$, where $P_{x1} = \text{Min}(P_{xA}, P_{xB})$, $P_{x2} = \text{Max}(P_{xA}, P_{xB})$, $P_{y1} = \text{Min}(P_{yA}, P_{yB})$, and $P_{y2} = \text{Max}(P_{yA}, P_{yB})$. When latitude interval $P_{xAB} = [P_{x1}, P_{x2}]$ and longitude interval $P_{yAB} = [P_{y1}, P_{y2}]$, the region consisting of points A and B is P_{AB} (P_{xAB} & P_{yAB}). For Figure 5b, the other spatial coordinates are denoted as $C(P_{xC}, P_{yC})$ and $D(P_{xD}, P_{yD})$, where $P_{x3} = \text{Min}(P_{xC}, P_{xD})$, $P_{x4} = \text{Max}(P_{xC}, P_{xD})$, $P_{y3} = \text{Min}(P_{yC}, P_{yD})$, and $P_{y4} = \text{Max}(P_{yC}, P_{yD})$. If the longitude interval $P_{xCD} = [P_{x3}, P_{x4}]$ and latitude interval $P_{yCD} = [P_{y3}, P_{y4}]$, the region composed of points C and D is $P_{CD}(P_{xCD}$ & $P_{yCD})$.

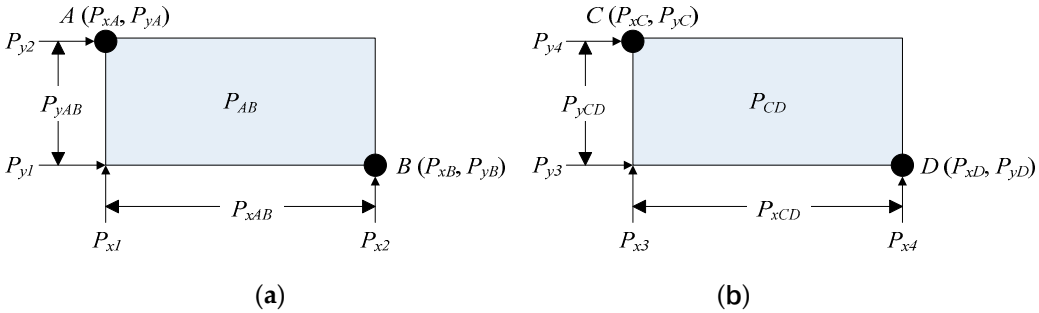


Figure 5. Spatial region graph. (**a**) The spatial region consisting of points A and B; (**b**) the spatial region consisting of points C and D.

Definition 11 (spatial intersection operation $\wedge_s$). *Let $P_{xi} = Max(P_{x1}, P_{x3})$, $P_{xj} = Min(P_{x2}, P_{x4})$, $P_{yi} = Max(P_{y1}, P_{y3})$, and $P_{yj} = Min(P_{y2}, P_{y4})$, then the intersecting operation of spatial regions P_{AB} and P_{CD} is $P_{AB} \wedge_s P_{CD} = (P_{xAB}$ & $P_{yAB}) \wedge_s (P_{xCD}$ & $P_{yCD}) = (P_x$ & $P_y)$, where the longitude interval $P_x = [P_{xi}, P_{xj}]$ and latitude interval $P_y = [P_{yi}, P_{yj}]$.*

The spatial intersection operation can be represented as shown in Figure 6.

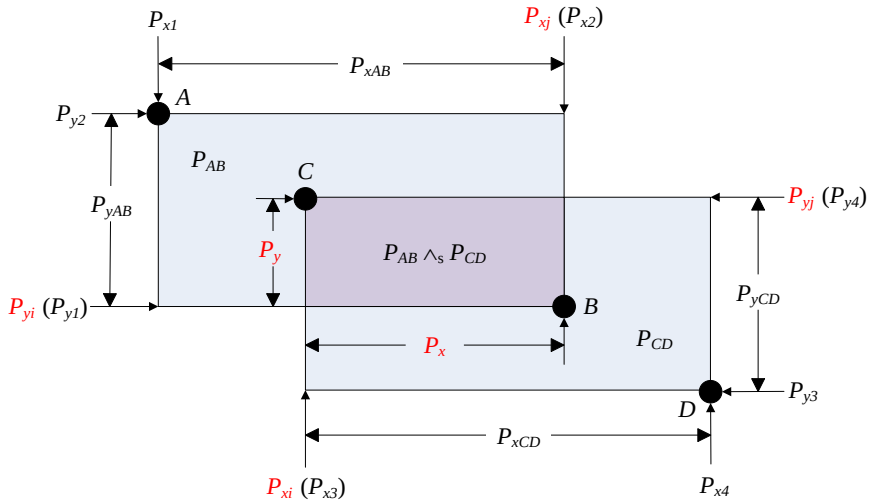


Figure 6. Spatial intersection operation.

Definition 12 (spatial merger operation $\vee_s$). *Let $P_{xi} = Min(P_{x1}, P_{x3})$, $P_{xj} = Max(P_{x2}, P_{x4})$, $P_{yi} = Min(P_{y1}, P_{y3})$, and $P_{yj} = Max(P_{y2}, P_{y4})$, then the intersection of spatial regions P_{AB} and P_{CD} is $P_{AB} \vee_s P_{CD} = (P_{xAB}$ & $P_{yAB}) \vee_s (P_{xCD}$ & $P_{yCD}) = (P_x$ & $P_y)$, where the longitude interval $P_x = [P_{xi}, P_{xj}]$ and latitude interval $P_y = [P_{yi}, P_{yj}]$.*

The spatial merger operation is shown in Figure 7.

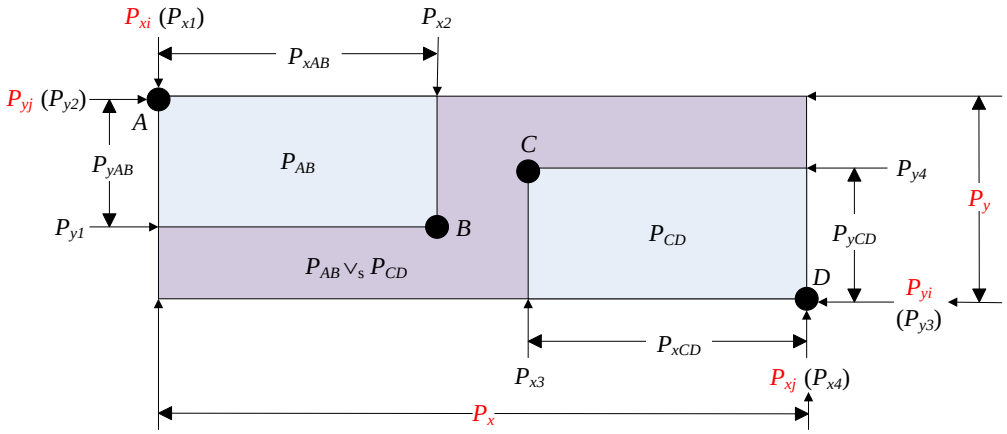


Figure 7. Spatial merger operation.

Definition 13 (spatial span ρ). *The spatial span $\rho(stG) = \{ [P_{xi}, P_{xj}]$ & $[P_{yi}, P_{yj}] \mid 1 \leq i, j \leq |E|, P_{xi} \leq P_{xj}, P_{yi} \leq P_{yj} \}$ for the spatiotemporal RDF graph stG. Let $P_{xmin} = Min_{1 \leq i \leq |E|} (P_{xi})$, $P_{xmax} = Max_{1 \leq i \leq |E|} (P_{xi})$, $P_{ymin} = Min_{1 \leq i \leq |E|} (P_{yi})$, and $P_{ymax} = Max_{1 \leq i \leq |E|} (P_{yi})$, then the spatial span of the graph stG is $(P_{x12}$ & $P_{y12}) \vee_s (P_{x23}$ & $P_{y23}) \vee_s \ldots \vee_s (P_{x(n-1)n}$ & $P_{y(n-1)n})$ $= ([P_{xmin}, P_{xmax}]$ & $[P_{ymin}, P_{ymax}])$, where $n = |E|$, and $P_{xmin} \leq P_{xmax}, P_{ymin} \leq P_{ymax}$.*

As for Definition 13, there is a spatial span $\rho(stG)$ = ([-6.04139, 15.21233] & [106.67, 145.7545]) and $\rho(stQ)$ = ([15.21233, 15.21233] & [145.7545, 145.7545]) for the spatiotemporal RDF data graph *stG* in Figure 2 and the spatiotemporal RDF query graph *stQ* in Figure 4, respectively.

For a spatiotemporal RDF data graph *stG* and a spatiotemporal RDF query graph *stQ*, it is possible for *stQ* to have a matching subgraph in *stG*, if and only if $\rho(stG) \wedge_s \rho(stQ)$ is non-null. For the spatiotemporal RDF data graph *stG* in Figure 2 and the spatiotemporal RDF query graph *stQ* in Figure 4, there is $\rho(stG) \wedge_s \rho(stQ)$ = ([−6.04139, 15.21233] &

[106.67, 145.7545]) $\wedge_s$ ([15.21233, 15.21233] & [145.7545, 145.7545]) = ([15.21233, 15.21233] & [145.7545, 145.7545]). When the result is non-null, it is preliminarily determined that the spatiotemporal RDF query graph *stQ* is likely to find a matching subgraph in the spatiotemporal RDF data graph *stG*.

3.2. Calculation of the Matching Order

The query process based on subgraph matching can be carried out if $\tau(stG) \wedge_t \tau(stQ)$ and $\rho(stG) \wedge_s \rho(stQ)$ are non-null. Only considering the spatiotemporal RDF query graph, the matching orders of query nodes are calculated in the process of subgraph matching. In order to clarify the matching order of query nodes, the node query candidate regions are given in the following.

For the matching order, $D(u)$ is the query candidate region of node u in the spatiotemporal RDF data graph, where $D(u)$ contains all data nodes that may match u. Node u and any node v in $D(u)$ should meet the following conditions:

(i) $\deg(u) \leq \deg(v)$;
(ii) $\text{deg-in}(u) \leq \text{deg-in}(v)$;
(iii) $\text{deg-out}(u) \leq \text{deg-out}(v)$.

The deg function $\deg(u)$ represents the degree of node u, where indegree function $\text{deg-in}(u)$ represents the indegree of node u, and outdegree function $\text{deg-out}(u)$ represents the outdegree of node u. Outdegree and indegree are numbers of outcoming and incoming edges from a node. If $\text{deg-in}(u) \leq \text{deg-in}(v)$ and $\text{deg-out}(u) \leq \text{deg-out}(v)$ are satisfied for nodes u and v, then $\deg(u) \leq \deg(v)$. In addition, if there is an edge connecting the query nodes u_1 and u_2 in two spatiotemporal RDF query graphs, a node v_1 in the query candidate region $D(u_1)$ of u_1 must be adjacent to a node v_2 in the query candidate region $D(u_2)$ of u_2, i.e., there is an edge between v_1 and v_2. The rule is called as the principle of AC (Arc Consistency). This means that if a node in the spatiotemporal RDF data graph exists in the query candidate region of a node u in the spatiotemporal RDF query graph, but not meeting the AC principle, then it should be removed from $D(u)$.

The first query node should be determined to query spatiotemporal RDF data by a subgraph matching algorithm and the first query node is selected according to the following rules:

(i) Select the node in the smallest query candidate region (i.e., the least number of nodes in the query candidate region) as the first query node. When the query candidate region with two or more nodes is the smallest, the approach in (ii) is adopted to select these nodes.
(ii) Select the node with the largest degree as the first node. When there are two or more query nodes with the same degree, the approach in (iii) is adopted to select these nodes.
(iii) Select the node with the maximum outdegree. When there are two or more nodes with the same outdegree, any node is selected as the first query node.

It is assumed that there are n query nodes in the spatiotemporal RDF query graph, and the order of the remaining n-1 query nodes is determined by the degree of association with the nodes that have been sorted. The query node with the largest degree of association that already exist in the partial matching order is ranked earlier. An approach similar to an RI algorithm is adopted to sort subsequent query nodes [34]. $\zeta_i = \{ u_1, u_2, \ldots, u_i \}$ represents a partial query order consisting of i nodes, where $i < n$. ξ_I is the collection of nodes that do not participate in sorting. Three sets about the candidate query node u are defined to select the next node in the sort:

(i) $V_{u,\,vis}$: The set of adjacency nodes belonging to u in i query nodes of ζ_I;
(ii) $V_{u,\,neig}$: The set of query nodes in ζ_i that are adjacent to at least one node in ξ_i and connected to u;
(iii) $V_{u,\,unv}$: The set of adjacent nodes of u that are not in ζ_i and are not adjacent to any node in ζ_i.

Select the next node in the sort as follows:

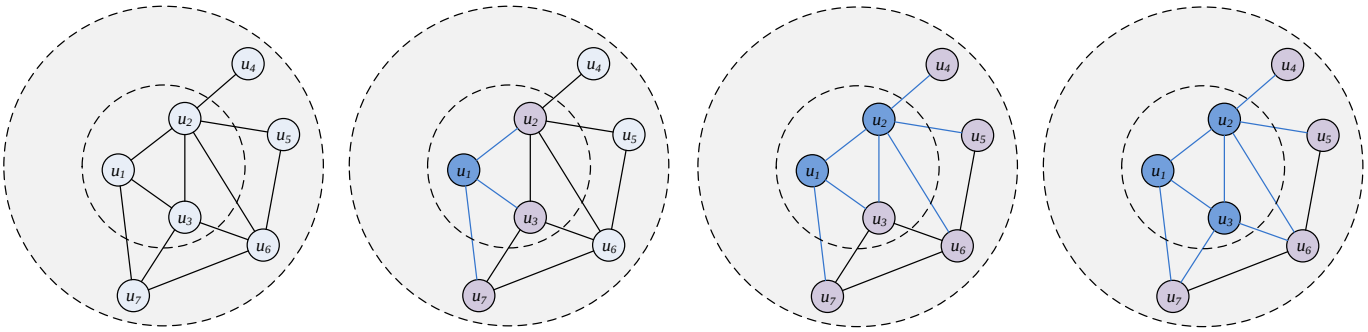
(i) Firstly, choose the node whose value of $|V_{u,\,vis}|$ is the maximum;

(ii) If the values of $| V_{u,\,vis} |$ are the same, then choose the node whose value of $| V_{u,\,neig} |$ is the maximum;

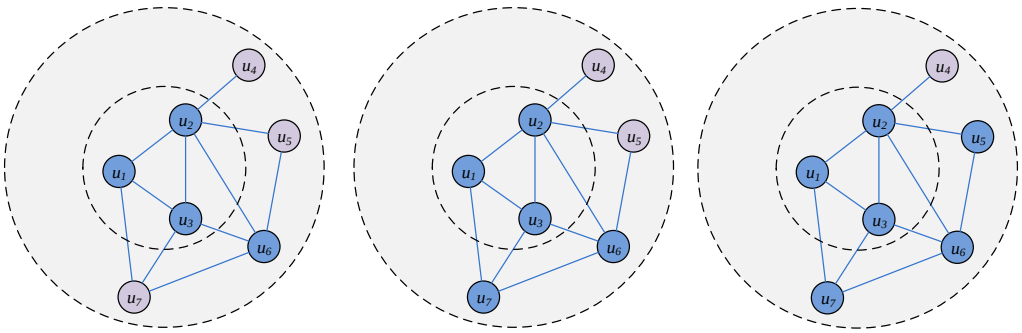
(iii) If the values of $| V_{u,\,neig} |$ are the same, then choose the node whose value of $| V_{u,\,unv} |$ is the maximum;

(iv) If the values of $| V_{u,\,unv} |$ are the same, then select any of the nodes.

Taking Figure 8a as an example, if the first query node u_1 has been selected, $\zeta_1 = \{ u_1 \}$ and $\xi_6 = \{ u_2, u_3, u_4, u_5, u_6, u_7 \}$. When selecting the next query node, consider that $V_{u2,\,vis} = \{ u_1 \}$, $V_{u3,\,vis} = \{ u_1 \}$, and $V_{u7,\,vis} = \{ u_1 \}$. If $V_{u4,\,vis}$, $V_{u5,\,vis}$ and $V_{u6,\,vis}$ are all $\emptyset$, so $| V_{u2,\,vis} | = | V_{u3,\,vis} | = | V_{u7,\,vis} | > | V_{u4,\,vis} | > | V_{u5,\,vis} | > | V_{u6,\,vis} |$, then the next query node can be considered in u_2, u_3, and u_7. Consider $V_{u2,\,neig} = \{ u_1 \}$, $V_{u3,\,neig} = \{ u_1 \}$, $V_{u7,\,neig} = \{ u_1 \}$, and $| V_{u2,\,neig} | = | V_{u2,\,neig} | = | V_{u7,\,neig} | = 1$, but the next query node is still not determined. Then continue to determine $| V_{u,unv} |$ value, including $V_{u2,\,unv} = \{ u_4, u_5, u_6 \}$, $V_{u3,\,unv} = \{ u_6 \}$, and $V_{u7,\,unv} = \{ u_6 \}$. Because $| V_{u2,\,unv} | = 3$, and $| V_{u3,\,unv} | = | V_{u7,\,unv} | = 1$, then $| V_{u2,\,unv} | > | V_{u3,\,unv} | = | V_{u7,\,unv} |$, which can determine the next query node for u_2. After updating the sets ζ_i and ξ_i, there is $\zeta_2 = \{ u_1, u_2 \}$ and $\xi_5 = \{ u_3, u_4, u_5, u_6, u_7 \}$, then the next query node can be selected, and the final query sequence is $\zeta_7 = \{ u_1, u_2, u_3, u_6, u_7, u_5, u_4 \}$.



(a) Query node order (b) The first query node selected (c) The second query node selected (d) The third query node selected



(e) The fourth query node selected (f) The fifth query node selected (g) The sixth query node selected

Figure 8. Query node selection order.

Figure 8b–g shows the node state at each sorting step, where the blue node has participated in sorting, and the remaining nodes have not participated in sorting. The purple node is the adjacent node of the node that has participated in sorting. The specific sorting process is explained as follows:

(i) When $\zeta_1 = \{ u_1 \}$ and $\xi_6 = \{ u_2, u_3, u_4, u_5, u_6, u_7 \}$, the node status is shown in Figure 8b. When $V_{u2,\,vis} = \{ u_1 \}$, $V_{u3,\,vis} = \{ u_1 \}$, $V_{u7,\,vis} = \{ u_1 \}$, and $V_{u4,\,vis}$, $V_{u5,\,vis}$, $V_{u6,\,vis} = \emptyset$, the node is selected in u_2, u_3, and u_7; When $V_{u2,\,neig} = \{ u_1 \}$, $V_{u3,\,neig} = \{ u_1 \}$, and $V_{u7,\,neig} = \{ u_1 \}$, the node is selected in u_2, u_3, and u_7; When $V_{u2,\,unv} = \{ u_4, u_5, u_6 \}$, $V_{u3,\,unv} = \{ u_6 \}$, and $V_{u7,\,unv} = \{ u_6 \}$, u_2 is removed from the unordered set and added to the partial query order set.

(ii) When $\zeta_2 = \{ u_1, u_2 \}$ and $\xi_5 = \{ u_3, u_4, u_5, u_6, u_7 \}$, the node state is shown in Figure 8c. When $V_{u3,\,vis} = \{ u_1, u_2 \}$, $V_{u4,\,vis} = \{ u_2 \}$, $V_{u5,\,vis} = \{ u_2 \}$, $V_{u6,\,vis} = \{ u_2 \}$, and $V_{u7,\,vis} = \{ u_1 \}$, u_3 *is* removed from the unordered set and added to the partial query order set.

(iii) When $\zeta_3 = \{ u_1, u_2, u_3 \}$and $\xi_4 = \{ u_4, u_5, u_6, u_7 \}$, the node status is shown in Figure 8d. When $V_{u4,\,vis} = \{ u_2 \}$, $V_{u5,\,vis} = \{ u_2 \}$, $V_{u6,\,vis} = \{ u_2, u_3 \}$, and $V_{u7,\,vis} = \{ u_1, u_3 \}$, the node is selected between u_6 and u_7; When $V_{u6,\,neig} = \{ u_1, u_2, u_3 \}$ and $V_{u7,\,neig} = \{ u_1, u_2, u_3 \}$, the node is selected between u_6 and u_7; When $V_{u6,\,unv} = \emptyset$ and $V_{u7,\,unv} = \emptyset$, the node is selected between u_6 and u_7; One of the nodes u_6 is selected and deleted from the unsorted set, and it is added to the partial query order set.

(iv) When $\zeta_4 = \{ u_1, u_2, u_3, u_6 \}$ and $\xi_3 = \{ u_4, u_5, u_7 \}$, the node state is shown in Figure 8e. When $V_{u4,\,vis} = \{ u_2 \}$, $V_{u5,\,vis} = \{ u_2, u_6 \}$and $V_{u7,\,vis} = \{ u_1, u_3, u_6 \}$, u_7 is removed from the unordered set and added to the partial query order set.

(v) When $\zeta_5 = \{ u_1, u_2, u_3, u_6, u_7 \}$and $\xi_2 = \{ u_4, u_5 \}$, the node status is shown in Figure 8f. When $V_{u4,\,vis} = \{ u_2 \}$ and $V_{u5,\,vis} = \{ u_2, u_6 \}$, u_5 is removed from the unsorted set and added to the partial query order set.

(vi) $\zeta_6 = \{ u_1, u_2, u_3, u_6, u_7, u_5 \}$ and $\xi_1 = \{ u_4 \}$. As shown in Figure 8b, only u_4 is the node that does not participate in sorting at this time, so the next selected query node is u_4. Thus, the final query sequence $\zeta_7 = \{ u_1, u_2, u_3, u_6, u_7, u_5, u_4 \}$. In this case, ξ set is empty.

The node with the largest out-degree in the smallest query candidate domain is selected as the first query vertex. The purpose is to find the most favorable result with the greatest probability and reduce the useless traversal. After determining the first query node, the order of the remaining n-1 query nodes are determined according to the degree of association with the sorted node. The significance of this sorting is to comprehensively consider the closeness of the relationship between the nodes, so as to facilitate the generation of the best query results.

For example, considering the variable "*?x*" in the query in spatiotemporal RDF query graph, we set u_1 = *"Antonio"*, u_2 = *"Miguel"*, u_3 = *"Benito"*, u_4 = *"Mateo"*, u_5 = *"Federico"*, u_6 = *"Luis"*, and u_7 = *"Lope"*. The node u_1 = *" Antonio"* is selected as the first query node, so there are $\zeta_1 = \{$ *Antonio* $\}$and $\xi_6 = \{$ *Miguel*, *Benito*, *Mateo*, *Federico*, *Luis*, *Lope* $\}$. When selecting the next query node, we consider $V_{u2,\,vis} = \{$ *Antonio* $\}$, $V_{u3,\,vis} = \{$ *Antonio* $\}$, and $V_{u7,\,vis} = \{$ *Antonio* $\}$. When $V_{u4,\,vis}$, $V_{u5,\,vis}$, $V_{u6,\,vis}$ are all empty, there is $| V_{u2,\,vis} | = | V_{u3,\,vis} | = | V_{u7,\,vis} | > | V_{u4,\,vis} | > | V_{u5,\,vis} | > | V_{u6,\,vis} |$. The next query node is generated in *"Miguel"*, *"Benito"* and *"Lope"*. Considering $V_{u2,\,neig} = \{$ *Antonio* $\}$, $V_{u3,\,neig} = \{$ *Antonio* $\}$, $V_{u7,\,neig} = \{$ *Antonio* $\}$, and $| V_{u2,\,neig} | = | V_{u2,\,neig} | = | V_{u7,\,neig} | = 1$, we cannot sure the next query node. We continue to judge the value of $| V_{u,unv} |$, where $V_{u2,\,unv} = \{$ *Mateo*, *Federico*, *Luis* $\}$, $V_{u3,\,unv} = \{$ *Luis* $\}$, and $V_{u7,\,unv} = \{$ *Luis* $\}$. Because of $| V_{u2,\,unv} | = 3$ and $| V_{u3,\,unv} | = | V_{u7,\,unv} | = 1$, we have $| V_{u2,\,unv} | > | V_{u3,\,unv} | = | V_{u7,\,unv} |$. After updating the collections ζ_i and ξ_i, we have $\zeta_2 = \{$ *Antonio*, *Miguel* $\}$ and $\xi_5 = \{$ *Benito*, *Mateo*, *Federico*, *Luis*, *Lope* $\}$, and then we can continue to select the next query node. The final query sequence is $\zeta_7 = \{$ *Antonio*, *Miguel*, *Benito*, *Luis*, *Lope*, *Federico*, *Mateo* $\}$.

When querying vertices, we define the chain query pattern, star query pattern and loop query pattern (Section 4.2). For example, Tom goes to school, then visits the library, and finally goes home. The query is about where Tom goes from school. This kind of query belongs to a chain query pattern. Tom's hobbies are running, playing basketball, and playing football. The query is about what Tom's hobbies are. This kind of query belongs to a star query pattern. Tom visits his teacher Traka while going to the library, and Traka goes to the English corner. The English corner is in the library. The query is about inquiring where Tom is. This kind of query belongs to a loop query pattern. Therefore, the proposed approach is interesting and the three kinds of queries are of value for spatiotemporal real applications.

The whole process of selecting the query node matching order is given by Algorithm 1, in which the ChooseFirstVertex function is used to select the first query node. Details on Algorithm 1 are described as follows:

Algorithm 1: Calculating the Matching Order Algorithm OrderMatchNodes.

Input: Spatiotemporal RDF query graph *stQ*
Output: matching order *Ord*
1: *ChooseFirstVertex*(*stQ*)
2: if $num(V_Q) > 1$
3: if $Max(degree(V_Q)) > 1$
4: $Ord \leftarrow Max(Outdegree(V_Q))$
5: end if
6: end if
7: $\xi \leftarrow V_Q$
8: while $|Ord| < |V_Q|$ do
9: for each u in ξ
10: $V_{u,vis}, V_{u,neig}, V_{u,unv} \leftarrow \emptyset$
11: for each u' in V_Q
12: if u' in Ord
13: if u' in $N(u)$
14: $V_{u,vis} = V_{u,vis} \cup \{ u' \}$
15: else if u' in $N(\xi \cap N(u))$
16: $V_{u,neig} = V_{u,neig} \cup \{ u' \}$
17: else if u' in $N(u)$ && u' not in $N(Ord)$
18: $V_{u,unv} = V_{u,unv} \cup \{ u' \}$
19: end for
20: end for
21: $M_{vis} = \max_{u \in Ord} | V_{u,vis} |$
22: $M_{neig} = \max_{u \in Mvis} | V_{u,neig} |$
23: $u_{max} = \text{random}(\max_{u \in Mneig} | V_{u,unv} |)$
24: append(Ord, u_{max})
25: $\xi = \xi \setminus \{ u_{max} \}$
26:end while

3.3. Spatiotemporal RDF Subgraph Matching

The querying of spatiotemporal RDF data is the process of finding the isomorphic subgraph to the spatiotemporal RDF query graph in the spatiotemporal RDF data graph. The concept of spatiotemporal RDF subgraph isomorphism is defined as follows.

Definition 14 (spatiotemporal RDF subgraph isomorphism). *The spatiotemporal RDF subgraph isomorphism means that there exists the injective function* f: $V \rightarrow V^q$, *which satisfies:*

- For any vertex $u \in V^q$, there is $F_{st}{}^q(u) \subseteq F_{st}(f(u))$;
- For any edge $(u_1, u_2) \in E^q$, there are $(f(u_1), f(u_2)) \in E$, and $F_{st}{}^q(u_1, u_2) = F_{st}(f(u_1), f(u_2))$.

In Definition 14, the injective function f: $V \rightarrow V^q$ is used for the spatiotemporal RDF data graph $stG(V, E, L, F_{st})$ and the spatiotemporal RDF query graph $stQ(V^q, E^q, L^q, F_{st}{}^q)$. In order to find the matching subgraphs (isomorphic subgraphs) corresponding to the spatiotemporal RDF query graph *stQ* embedded in the spatiotemporal RDF data graph *stG* and complete the query of spatiotemporal RDF data, a spatiotemporal RDF query algorithm stQuery based on the general framework of subgraph matching algorithms is proposed. Algorithm 2 outlines the overall process of spatiotemporal RDF data query algorithm stQuery based on subgraph matching.

Algorithm 2: Spatiotemporal RDF Query Algorithm stQuery

Input: Spatiotemporal RDF data graph *stG*, spatiotemporal RDF query graph *stQ*
Output: All subgraphs in *stG* that match *stQ*
1: $M \leftarrow \emptyset$
2: ST_G = *GetSTSpan*(*stG*)
3: ST_Q = *GetSTSpan*(*stQ*)
4: if $ST_G \wedge ST_Q$ is not null
5: u = *ChooseFirstVertex*(*stQ*)
6: $D(u) \leftarrow$ *GetCanddidate*(*stG*, u)
7: if $D(u)$ is not null
8: *Ord* ← *OrderMatchNodes*(*stQ*)
9: for each $v \in D(u)$
10: *UpdateState*(u, v, M)
11: *SubgraphSearch*(*stG*, *stQ*, *Ord*, u, v)
12: report M
13: *RestoreState*(u, v, M)
14: end for
15: end if
16: end if
17: *OrderMatchNodes*(*stQ*)
18: for each $v \in stQ$
19: if $num(Max(\mid V_{u,\,vis} \mid)) > 1$
20: for each $v \in Max(\mid V_{u,\,neig} \mid)$
21: $U \leftarrow add(v)$
22: end for
23: end if
24: end for
25: *Ord* ← U

In the process of query, the set M of matched subgraphs is initially assigned to null (line 1). Then, the spatiotemporal spans of *stG* and *stQ* are obtained by the *GetSTSpan* function. If the spatiotemporal span intersection between *stG* and *stQ* is empty, it means that there is no subgraph matching with *stQ* in *stG*. On the other hand, the next step of the query process is continued (lines 2–4). Next, the first query node in *stQ* is selected by the *ChooseFirstVertex* function, and the candidate regions of this node are obtained by the *GetCanddidate* function (lines 5–6). If the candidate region is not empty, the nodes other than the initial query node in *stQ* are sorted. Each node of the candidate region in turn performs a *SubgraphSearch* algorithm (in Algorithm 3). When a query node u and a matching of the data node v are found, (u, v) is added to M, ending up with an updated matching subgraph set M (lines 9–13). Finally, M contains all the matched subgraphs of *stQ* in *stG*.

The core of the subgraph matching process of spatiotemporal RDF query is the recursive process based on a backtracking strategy. Algorithm 3 gives the spatiotemporal RDF subgraph matching algorithm SubgraphSearch.

When the number of matched nodes and edges is equal to the number of nodes and edges in the query graph, a matching subgraph M (lines 1–2) that matches *stQ* in *stG* can be returned. Otherwise, the next node is denoted as u', and the candidate region of u' is found. If the node u is located before node u' in the sorting ζ, then the candidate node set $C(u')$ matching node u' is obtained from $Neighbor(f(u)) \cap D(u')$ (line 5). Next, the *CheckFeasibility* function is needed to verify whether the query nodes and data nodes meet the feasibility conditions. When all the query nodes are matched, a match for the query graph *stQ* is found in the data graph *stG* and added to the set of matching results. Traceback means deleting the last matching pair of query nodes and target nodes from M and deleting the mappings between such nodes. This algorithm returns all the matches found.

Algorithm 3: Subgraph Matching Algorithm SubgraphSearch

Input: Spatiotemporal RDF data graph *stG*, spatiotemporal RDF query graph *stQ*, matching order set *Ord*, data node v, query node u
Output: The subgraph M matched with *stQ* in *stG*

```
if |V| = |V_Q| && |E| = |E_Q|
    return M
else
    u' ← NextQueryVertex( )
    D(u') ← Neighbor(f(u)) ∩ Canddidate(stG, u')
    if D(u') is not null
        for each v' ∈ D(u')
            if (u, u') ∈ E_Q && (v, v') ∉ E_G
                D(u') ← D(u') \ { v'}
        for each v' ∈ D(u') such that v' is not matched do
                CheckFeasibility(u', v')
                UpdateState(u', v', M)
                SubgraphSearch(stG, stQ, u', v')
                RestoreState(u', v', M)
        end for
    end if
end if
```

4. Experiments

In this section, we evaluate the spatiotemporal RDF data query approach. The experiments are all carried out under the windows 10_64 bit operating system. The processor is Intel(R) Core(TM) i5-8265U CPU @ 1.60 GHz 1.80 GHz, and RAM is 16.0 GB.

4.1. Experimental Dataset

The dataset in the experiment is extracted from yago (version 3.1), which is from the real world, and its accuracy has been manually evaluated. Many facts and entities are endowed with temporal and spatial attributes to form spatiotemporal data. In this subsection, four sub-datasets are selected to extract spatiotemporal datasets, and they are yagoFacts dataset, yagoMetaFacts dataset, yagoDateFacts dataset and yagoGeonamesOnlyData dataset, which are introduced as follows:

(i) yagoFacts dataset: The dataset is 972 MB in size, involving a total of 12,430,700 pieces of data and including all the instance data in the yago dataset (no spatiotemporal information);
(ii) yagoMetaFacts dataset (395 MB): This dataset is 395 MB in size and contains 3,824,875 pieces of data, including all the temporal and spatial metadata in the yago dataset;
(iii) yagoDateFacts dataset: The data set is 412 MB in size, involving a total of 4,190,241 pieces of data and including data only with time attribute in the yago dataset, in which the temporal information is presented in the form of date (year, month, day);
(iv) yagoGeonamesOnlyData dataset: The dataset is 4.11 GB in size, involving a total of 61,605,695 pieces of data and including all the data with spatial attributes in the yago dataset, as well as a large number of relevant information data, in which the spatial information is presented in the form of geographic coordinates (longitude and latitude).

Table 2 introduces the original datasets and indicates the specific information contained in each dataset.

Table 2. Statistics of the four real datasets.

Datasets	Number of Data	Temporal Information	Spatial Information
yagoFacts	12,430,700	N	N
yagoMetaFacts	3,824,875	Y	Y
yagoDateFacts	4,190,241	Y	N
yagoGeonamesOnlyData	61,605,695	N	Y

After taking a series of extraction and integration operations, a synthetic dataset is obtained. The subsequent experiments in this paper are conducted on this Dataset. The Dataset is 481.65 MB in size and contains temporal data (involving only temporal information), spatial data (involving only spatial information), spatiotemporal data (involving both temporal and spatial information), and non-spatiotemporal data (involving neither temporal nor spatial information). This dataset is described in Table 3.

Table 3. Statistics of the synthetic datasets.

Data Type	Number
temporal data	417,963
spatial data	3,965,854
spatiotemporal data	7901
non-spatiotemporal data	442,607

4.2. The Experimental Setup

This subsection divides experiments into two parts and conducts the corresponding experiments on the effect and efficiency of the query approach.

The first part is the experiment to test the effects of stQuery through the query about different types of data. Four groups of queries with different types of data are set. Each group of queries includes three similar samples (the same number of query vertices and similar query contents) and twelve query samples to reduce the contingency of experimental results. The specific contents of queries in these four groups are as follows:

(i) Temporal queries (test sample 1, test sample 2, test sample 3): Queries containing only temporal data, involve temporal entities or facts;
(ii) Spatial queries (test sample 4, test sample 5, test sample 6): Queries containing only spatial data, involve spatial entities and facts;
(iii) Spatiotemporal queries (test sample 7, test sample 8, test sample 9): Queries containing temporal data and spatial data at the same time, include temporal data and spatial information;
(iv) Non-spatiotemporal queries (test sample 10, test sample 11, test sample 12): Queries involve neither temporal nor spatial information.

The second part is the experiment to test the query performance. Through the comparative experiments based on the control variable method, stQuery is compared with the current more advanced query algorithms st-SDS [26], $\text{Turbo}_{\text{HOM++}}$ [35] and f-ASM [36]. The control variable method refers to the method of turning the problem of multiple factors into a problem of multiple single factors, and changing only one of them so as to study the influence of this factor during the experiments. A control variable is any factor that is controlled or held constant during an experiment. The query efficiency is tested by comparing the query response time for different query graph patterns. Three query graph patterns are the chain query pattern, star query pattern, and loop query pattern. Figure 9 shows simple examples of these three query graph patterns. Figure 9a is a sample of chain query pattern with a double-hop path. Figure 9b is a sample of star query pattern with six nodes. Figure 9c is a sample of loop query pattern with four nodes. We divide this part of the experiment into three groups. Each group contains three different sizes of test sample queries, and each test sample use stQuery, st-SDS, $\text{Turbo}_{\text{HOM++}}$ and f-ASM to query.

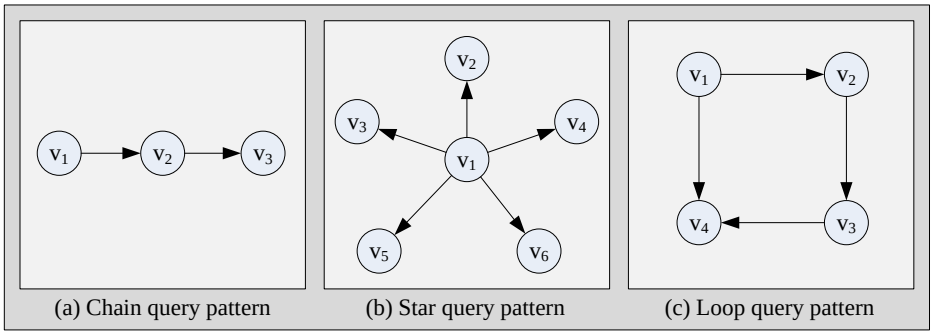


Figure 9. Query pattern graph.

4.3. Experimental Results

This subsection presents the experimental results into two parts: the effect and performance analysis of the algorithm.

4.3.1. The Effect of the Algorithms

Twelve test samples are of similar size in order to ensure the uniqueness of variables, which contain seven query nodes. The experimental results are shown in Tables 4–6. Table 4 describes the experimental effects of temporal and spatial queries. Table 5 describes the experimental effects of spatiotemporal and non-spatiotemporal queries. Taking the above query results into comprehensive consideration, Table 6 describes the average experimental effects about queries of each group.

Table 4. stQuery implementation effect of temporal and spatial queries.

Temporal Query	Response Time (s)	Spatial Query	Response Time (s)
test sample 1	105.0624387	test sample 4	108.7450422
test sample 2	100.1976586	test sample 5	108.8466299
test sample 3	102.9692515	test sample 6	109.6339075

Table 5. stQuery implementation effect of spatiotemporal and non-spatiotemporal queries.

Spatiotemporal Query	Response Time (s)	Non-Spatiotemporal Query	Response Time (s)
test sample 7	114.3038049	test sample 10	97.1463715
test sample 8	111.7024177	test sample 11	96.9176396
test sample 9	111.7339461	test sample 12	98.7803164

Table 6. Average response time.

Query Category	Temporal Query	Spatial Query	Spatiotemporal Query	Non-Spatiotemporal Query
Response Time (s)	102.6697829	109.0751932	112.5800562	97.6147758

As shown in Table 4, the query response time of the temporal query test sample is approximately between 100 s and 106 s, while that of the spatial query test sample is approximately between 108 s and 110 s. It can be seen that the efficiency of spatial query is slightly faster than that of temporal query. The main reason lies in the influence of data sets. There are more spatial data than temporal data, so it takes more time to match the corresponding spatial data in the spatiotemporal RDF data graph in the process of spatial query.

For Table 5, the query response time of the spatiotemporal query test sample is approximately between 111 s and 115 s, while that of the non-spatiotemporal query test sample is

approximately between 96–99 s. Therefore, it can be known that the efficiency of the spatiotemporal query is generally faster than that of the non-spatiotemporal query. It is mainly caused by the complexity of spatiotemporal data. Compared with non-spatiotemporal data, spatiotemporal data has more attribute information, which also requires more complex matching in the process of query.

The average query response time is shown in Table 6. The average query time is represented roughly: spatiotemporal query > spatial query > temporal query > non-spatiotemporal query. This is mainly caused by the complexity of the query information and the composition of the dataset. Therefore, the more complex the query information is, the lower the query efficiency is. In the case of similar complexity of query information, the more relevant the data, the lower the query efficiency.

In the experiment of spatiotemporal query, besides the above query tests, the query whose spatiotemporal range exceeds the spatiotemporal RDF data graph is also tested to verify the effectiveness of the "preliminary spatiotemporal determination" method. Experiments show that for the general spatiotemporal RDF query graph, when the spatiotemporal range exceeds the spatiotemporal RDF data graph, the feedback that the query result is empty can always be obtained within 90 s, which avoids a large amount of time consumption in the subgraph matching process.

4.3.2. The Performance Analysis of the Algorithms

In the performance analysis part, stQuery is compared with st-SDS, $\text{Turbo}_{\text{HOM++}}$ and f-ASM in three aspects: chain query pattern, star query pattern and loop query pattern.

(a) chain query pattern

As shown in Figure 10, when the query graph is a chain query pattern, five groups of experiments are carried out on the query graphs with nodes 3, 4, 5, 6, and 7. The $\text{turbo}_{\text{HOM++}}$ algorithm has the shortest query response time and the highest query efficiency when the query node is 3. With the increase of query nodes, the query response time of this algorithm grows faster, surpassing the other three algorithms. The query response time of the stQuery algorithm almost does not change significantly in each group of experiments, and the query efficiency is relatively high, which indicates that the algorithm has a good performance to the query graph of chained query pattern.

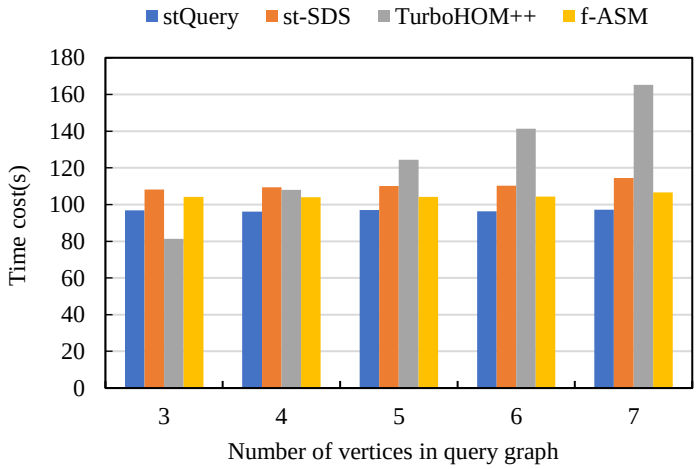


Figure 10. Experimental comparison of the chain query pattern.

(b) star query pattern

In Figure 11, when the query graph is a star query pattern, five groups of experiments are carried out for the query graphs with nodes of 4, 5, 6, 7, and 8. The experimental results show that the stQuery has a lower time cost than st-SDS and TurboHOM++, and has slightly better query efficiency than f-ASM.

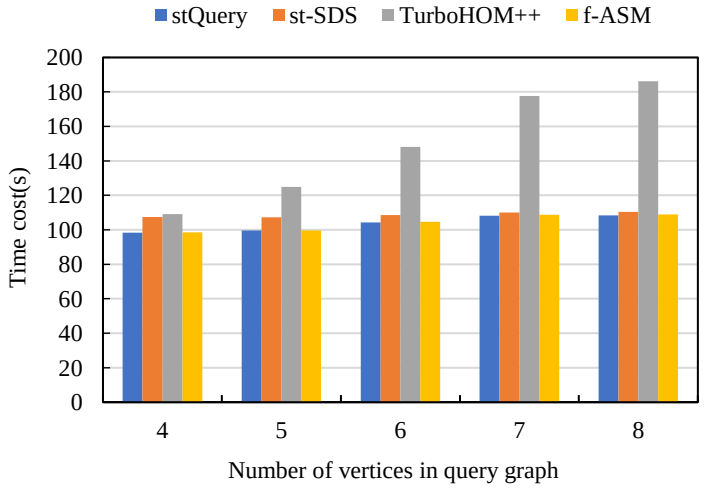


Figure 11. Experimental comparison of the star query pattern.

(c) loop query pattern

According to Figure 12, when the query graph is the loop query pattern, five groups of experiments are carried out on the query graphs with nodes 3, 4, 5, 6, and 7. Experimental results show that $\text{Turbo}_{\text{HOM++}}$ algorithm has the best query efficiency when the query node is 3. When the query nodes are increased to 4, 5, and 6, stQuery has the greatest advantage in query efficiency, which is generally better than $\text{Turbo}_{\text{HOM++}}$ and slightly better than f-ASM. The overall query efficiency of stQuery is relatively stable. The query performance of the stQuery algorithm has slight advantages over f-ASM in loop query patterns, but it is generally better than st-SDS and TurboHOM++.

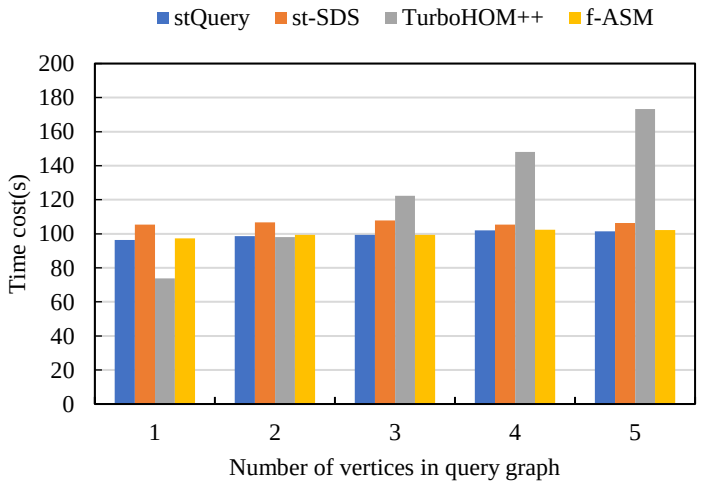


Figure 12. Experimental comparison of the loop query pattern.

5. Application Discussion

In order to better apply the technology of spatiotemporal RDF data query based on subgraph matching, in this section, we give general steps on how to use our approach in spatiotemporal applications.

Step 1: We can formally represent spatiotemporal data with temporal features according to Definitions 1 and 2, represent spatiotemporal data with spatial features according to Definitions 3 and 4, and represent spatiotemporal data with both temporal features and spatial features according to Definition 5.

Step 2: According to Algorithm 1, we can calculate the matching order. In this process, we can perform temporal interval matching according to Definitions 8–10, and perform spatial interval matching according to Definitions 11–13.

Step 3: On the basis of Steps 1 and 2, for a specific query in spatiotemporal applications, we can obtain the desired query results according to Algorithms 2 and 3.

6. Conclusions

In this paper, a spatiotemporal RDF model st-RDF is proposed. Based on this data model, spatiotemporal data with temporal and spatial attributes can be represented and operated. This paper then proposes a spatiotemporal RDF query algorithm stQuery, and mainly uses the sorting method of RI algorithm to sort the query node according to the correlation degree between the query node and the candidate node, which promotes the further improvement of the query efficiency. Experiments show that the proposed spatiotemporal RDF model and the corresponding query approach have relatively good performances. In future work, we plan to add a predicate index to investigate the query scope of the spatiotemporal RDF query. On the other hand, spatiotemporal RDF graphs are more likely to violate predefined spatial and temporal constraints due to the dynamic changes of spatiotemporal data. As a result, based on our model, we can define some constraint definitions, rules, and algorithms for checking and fixing inconsistencies, as well as solve consistency problems due to updates.

Author Contributions: Conceptualization, Xiangfu Meng and Lin Zhu; Methodology, Xiangfu Meng, Lin Zhu and Qing Li; Validation, Lin Zhu, Qing Li and Xiaoyan Zhang; Investigation: Xiangfu Meng, Lin Zhu and Qing Li; Writing (Original Draft), Xiangfu Meng, Lin Zhu and Qing Li; Writing (Review and Editing), Lin Zhu and Xiaoyan Zhang; Supervision, Xiangfu Meng; Project Administration, Xiangfu Meng. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China (No. 61772249), the General Research Foundation of the Liaoning Education Department, China (LJ2019QL017), and the Scientific Research Fund of the Liaoning Education Department (LJKZ0355).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lassila, O.; Swick, R.R. *Resource Description Framework (RDF) Model and Syntax Specification*. W3C. 1998. Available online: https://www.w3.org/TR/1999/REC-rdf-syntax-19990222/ (accessed on 1 September 2021).
2. Perry, M.; Jain, P.; Sheth, A.P. Sparql-st: Extending Sparql to Support Spatiotemporal Queries. In *Geospat. Semant. Semant. Web*; Springer: Boston, MA, USA, 2011; Volume 12, pp. 61–86. [CrossRef]
3. Gutierrez, C.; Hurtado, C.; Vaisman, A. Temporal rdf. In Proceedings of the European Semantic Web Conference, Crete, Greece, 29 May–1 June 2005. [CrossRef]
4. Gutierrez, C.; Hurtado, C.A.; Vaisman, A. Introducing Time into RDF. *IEEE Trans. Knowl. Data Eng.* **2006**, *19*, 207–218. [CrossRef]
5. Pugliese, A.; Udrea, O.; Subrahmanian, V.S. Scaling RDF with time. In Proceedings of the 17th International Conference on World Wide Web, Beijing, China, 21–25 April 2008. [CrossRef]
6. Motik, B. Representing and querying validity time in RDF and OWL: A logic-based approach. *J. Web Semantics* **2012**, *12*, 3–21. [CrossRef]
7. Zhang, F.; Wang, K.; Li, Z.; Cheng, J. Temporal Data Representation and Querying Based on RDF. *IEEE Access* **2019**, *7*, 85000–85023. [CrossRef]
8. Hayes, J. *A Graph Model for RDF*; Darmstadt University of Technology: Darmstadt, Germany; University of Chile: Santiago de Chile, Chile, 2004.
9. Kolas, D.; Dean, M.; Hebeler, J. Geospatial Semantic Web: Architecture of Ontologies. In Proceedings of the 2006 IEEE Aerospace Conference, Big Sky, MT, USA, 4–11 March 2006; Volume 3799, pp. 183–194. [CrossRef]
10. Smart, P.D.; Abdelmoty, A.I.; El-Geresy, B.A.; Jones, C.B. A framework for combining rules and geo-ontologies. In *International Conference on Web Reasoning and Rule Systems*; Springer: Berlin/Heidelberg, Germany, 2007. [CrossRef]
11. Batsakis, S.; Petrakis, E.G. SOWL: Spatio-temporal representation, reasoning and querying over the semantic web. In Proceedings of the 6th International Conference on Semantic Systems, Graz, Austria, 1–3 September 2010. [CrossRef]
12. Batsakis, S.; Petrakis, E.G. SOWL: A framework for handling spatio-temporal information in OWL 2.0. In *International Workshop on Rules and Rule Markup Languages for the Semantic Web. Proceedings of the 5th International Symposium, RuleML 2011–Europe, Barcelona, Spain, 19–21 July 2011*; Springer: Berlin/Heidelberg, Germany, 2011. [CrossRef]
13. Ademaj, F.; Schwarz, S.; Berisha, T.; Rupp, M. A Spatial Consistency Model for Geometry-based Stochastic Channels. *IEEE Access* **2019**, *7*, 183414–183427. [CrossRef]

14. Cui, Q.; Wang, N.; Haenggi, M. Vehicle Distributions in Large and Small Cities: Spatial Models and Applications. *IEEE Trans. Veh. Technol.* **2018**, *67*, 10176–10189. [CrossRef]
15. Hoffart, J.; Berberich, K.; Weikum, G. YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. *Artif. Intell.* **2013**, *194*, 28–61. [CrossRef]
16. Hoffart, J.; Suchanek, F.M.; Berberich, K.; Lewis-Kelham, E.; De, M.G.; Weikum, G. Yago2: Exploring and querying world knowledge in time, space, context, and many languages. In Proceedings of the 20th International Conference Companion on World Wide Web, Hyderabad, India, 28 March 2011. [CrossRef]
17. Koubarakis, M.; Kyzirakos, K. Modeling and querying metadata in the semantic sensor web: The model stRDF and the query language stSPARQL. In *Extended Semantic Web Conference. Proceedings of the 7th Extended Semantic Web Conference, ESWC 2010, Crete, Greece, 30 May–3 June 2010*; Springer: Berlin/Heidelberg, Germany, 2010. [CrossRef]
18. Wang, D.; Zou, L.; Zhao, D. gst-Store: An Engine for Large RDF Graph Integrating Spatiotemporal Information. In Proceedings of the 17th International Conference on Extending Database Technology (EDBT 2014), Athens, Greece, 24–28 March 2014. [CrossRef]
19. Wang, D.; Zou, L.; Zhao, D. gst-store: Querying large spatiotemporal RDF graphs. *Data Inf. Manag.* **2017**, *1*, 84–103. [CrossRef]
20. Lu, X.; Hu, T.; Yin, F. A Novel Spatiotemporal Fuzzy Method for Modeling of Complex Distributed Parameter Processes. *IEEE Trans. Ind. Electron.* **2018**, *66*, 7882–7892. [CrossRef]
21. Xu, L.; Wong, A.; Clausi, D.A. A Novel Bayesian Spatial–temporal Random Field Model Applied to Cloud Detection from Remotely Sensed Imagery. *IEEE Trans. Geosci. Remote Sens.* **2017**, *55*, 4913–4924. [CrossRef]
22. Sun, K.; Zhu, Y.; Pan, P.; Hou, Z.; Wang, D.; Li, W.; Song, J. Geospatial data ontology: The semantic foundation of geospatial data integration and sharing. *Big Earth Data* **2019**, *3*, 269–296. [CrossRef]
23. Kyzirakos, K.; Karpathiotakis, M.; Koubarakis, M.S. A semantic geospatial DBMS. In Proceedings of the ISWC, Boston, MA, USA, 11–15 November 2012. [CrossRef]
24. Vlachou, A.; Doulkeridis, C.; Glenis, A.; Santipantakis, G.M.; Vouros, G.A. Efficient spatio-temporal RDF query processing in large dynamic knowledge bases. In Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, Limassol, Cyprus, 8–12 April 2019. [CrossRef]
25. Wu, D.; Zhou, H.; Shi, J.; Mamoulis, N. Top-k relevant semantic place retrieval on spatiotemporal RDF data. *VLDB J.* **2020**, *29*, 893–917. [CrossRef]
26. Eom, S.; Shin, S.; Lee, K.H. Spatiotemporal query processing for semantic data stream. In Proceedings of the 9th International Conference on Semantic Computing, Anaheim, CA, USA, 7–9 February 2015. [CrossRef]
27. Cordella, L.P.; Foggia, P.; Sansone, C.; Vento, M. A (Sub) Graph Isomorphism Algorithm for Matching Large Graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* **2004**, *26*, 1367–1372. [CrossRef] [PubMed]
28. Shang, H.; Zhang, Y.; Lin, X.; Yu, J.X. Taming verification hardness: An efficient algorithm for testing subgraph isomorphism. In Proceedings of the VLDB Endowment, Auckland, New Zealand, 1 August 2008. [CrossRef]
29. He, H.; Singh, A.K. Graphs-at-a-time: Query language and access methods for graph databases. In Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, Vancouver, BC, Canada, 9 June 2008. [CrossRef]
30. Zhang, S.; Li, S.; Yang, J. GADDI: Distance index based subgraph matching in biological networks. In Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, Saint Petersburg, Russia, 24 March 2009. [CrossRef]
31. Zhao, P.; Han, J. On graph query optimization in large networks. In Proceedings of the VLDB Endowment, Singapore, 1 September 2010. [CrossRef]
32. Han, W.S.; Lee, J.; Lee, J.H. Turboiso: Towards ultrafast and robust subgraph isomorphism search in large graph databases. In Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, New York, NY, USA, 22 June 2013. [CrossRef]
33. Lee, J.; Han, W.S.; Kasperovics, R.; Lee, J.H. An in-depth comparison of subgraph isomorphism algorithms in graph databases. In Proceedings of the VLDB Endowment, Trento, Italy, 1 December 2012. [CrossRef]
34. Bonnici, V.; Giugno, R. On the Variable Ordering in Subgraph Isomorphism Algorithms. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **2016**, *14*, 193–203. [CrossRef] [PubMed]
35. Kim, J.; Shin, H.; Han, W.S.; Hong, S.; Chafi, H. Taming subgraph isomorphism for RDF query processing. *Proc. VLDB Endow.* **2015**, *8*, 1238–1249. [CrossRef]
36. Li, G.; Yan, L.; Ma, Z. An approach for approximate subgraph matching in fuzzy RDF graph. *Fuzzy Sets Syst.* **2019**, *376*, 106–126. [CrossRef]