Big Geospatial Data and Geospatial Semantic Web: Current State and Future Opportunities

Chuanrong Zhang \textsuperscript{a}, Tian Zhao \textsuperscript{b}, Weidong Li \textsuperscript{a}

\textsuperscript{a} Department of Geography & Center for Environmental Sciences and Engineering, University of Connecticut, Storrs, CT, USA

\textsuperscript{b} Department of Computer Science, University of Wisconsin-Milwaukee, Milwaukee, WI, USA

Abstract A huge part of big data is geospatial data. With the rapid development of remote sensing, geospatial positioning systems (GPS), Web mapping, cellular communications (such as smart phones), and wiki-based collaboration technologies, geospatial data are being captured at an unprecedented scale and rate. The size of geospatial data is growing rapidly every year. Although recent developments of geographic information and Internet technologies have enabled accessing and sharing big geospatial data, we still face huge challenges for transparent geospatial data exchange and sharing because of the lack of fast and interoperating algorithms and tools along with heterogeneous semantic problems. The Geospatial Semantic Web was proposed to overcome semantic and spatial computation limitations. By associating spatial data content from the Web with ontologies that would supply context and meaning, the vision of a Geospatial Semantic Web is to extract geospatial knowledge from the Web regardless of geospatial data formats or sources; thus it can facilitate transparent geospatial data exchange, sharing, and query. For example, the Geospatial Semantic Web will be able to search for needed geospatial information not based on keywords but on data contents and context, e.g. understanding what the word “near” means based on different spatial data contents and context. However, it still is a long way to go to realize this goal. For another example, spatial queries from big geospatial data are complex and time-consuming. Spatial query processing over big geospatial data requires intensive disk input/output (I/O) access and spatial computation. New efficient parallel algorithms and cloud computing approaches are needed to support efficient parallel execution of big geospatial data query.

In this book chapter, we investigate the challenges and opportunities of the Geospatial Semantic Web brought for sharing and utilizing big geospatial data. For example, it is very inefficient to query big geospatial data using the standard Semantic Web query language—GeoSPARQL because the data in the knowledge base of the Geospatial Semantic Web are no longer indexed to support efficient big geospatial data query. A new query strategy is needed to reduce the runtime costs of GeoSPARQL queries through on-the-fly spatial indexing and parallel execution.

1. Big Geospatial Data
Because of technology development, datasets are rapidly growing in data volume. Many technologies such as ubiquitous information-sensing mobile devices, aerial sensory technologies, cameras, social media networks, remote sensing imagery, and wireless sensor networks are increasingly used to gather various data sets. Large volumes of data are being created and continue to increase every day. A large part of big data can be geo-referenced. In fact, 80% of big data are geospatial data (Morais 2012). Geospatial data contain information about a physical object that can be represented by numerical values in a geographic coordinate system. Geospatial data represents the location, size, and shape of an object on Earth such as a school, house, sea, park or county. Remote sensing, GPS, GIS, and VGI are major modern geospatial technologies to collect and handle geospatial data. In recent years, these geospatial technologies have generated large volumes of geospatial data about things and human beings, which are available for better understanding our coupled human and environmental systems.

Remote sensing is the science of deriving information about an object from measurements made at a distance from the object, i.e., without actually coming in contact with it (Campbell and Wynne 2011). Remote sensing data are undergoing rapid growth because of recent advances in remote sensing techniques. Every day many spaceborne and airborne sensors from many different countries collect a massive amount of remotely sensed data. Remote sensing devices have been widely used to observe our planet from different angles and perspectives for different purposes. The remote sensing data collected by a single satellite data center are dramatically increasing by several terabytes per day (Gamba et al. 2011). Global archived remotely sensed data would probably exceed one exabyte based on the statistics of the Open Geospatial Consortium (OGC). A huge amount of remotely sensed data are now freely available from the National Aeronautics and Space Administration (NASA) Open Government Initiative. Only one of NASA’s archives, the Earth Science Data and Information System (ESDIS), holds 7.5 (Petabyte) of data with nearly 7,000 unique datasets and a million users in 2013. Remotely sensed images include a variety of data from multiple sources (laser, radar, optical, etc) and are multi-temporal (collected on different dates) and multi-resolution (from high-resolution to medium and coarse-resolution). The advent of high-resolution earth observation has led to the high dimensionality of remotely sensed data. Large-scale environmental monitoring research has generated multi-temporal and multi-sensor remotely sensed data at multiple scales: local, regional, and global scales. Those remotely sensed data are used for different applications such as global climate change, urban planning, and natural disaster management. Large remote sensing applications overwhelmed with massive remotely sensed data are regarded as typical big geospatial data-intensive challenges (Ma et al. 2015).

GPS is an electronic system that uses satellites to determine the position of an object such as a vehicle or a person. GPS technology has been developed to provide precise positional and velocity data and global time synchronization for air, sea, and land travel. Advances in GPS technologies have helped to generate a big volume of geospatial information for different applications. A growing number of cell phones, personal digital assistants (PDAs), digital cameras, and other handheld devices use GPS technology to provide users with information based on their locations. For example, GPS-enabled devices and mobile applications have generated huge volumes of health data by capturing daily activities of patients. A GPS-enabled tracker can produce a large volume of health data by recording inhaler usage by asthmatics. For another example, GPS has been extensively used for generating data tracking of suspected criminals and for land surveying.
GPS plays many important roles in bridging a variety of Web applications and end users. The availability of smartphones with built-in GPS and developments of web technology called ‘Web 2.0’ allow end users to produce, visualize, and analyze ‘mashups’ of data from different sources over the Web. Many GPS-enabled web applications have been created. Nowadays many websites provide web services based on users’ geographic location. Innovative GPS enabled device use will continue to grow as a means to help get insights into different applications through real-time data generation and complex data analytics that would otherwise be hard to piece together. GPS has been and will continue to be used in many fields such as transportation. For example, raw GPS logs have been used to understand transportation modes such as walking and driving, which is a type of human behavior data that can provide pervasive computing systems with more contextual information of a user’s mobility (Zheng et al. 2010). For another example, GPS-equipped taxis have been regarded as mobile sensors probing the travel speed of each road (Zheng et al. 2013). GPS enabled navigation devices can integrate real-time traffic patterns and alerts with navigation maps and suggest the best routes to drivers.

Advances in GPS, Web mapping, cellular communications, and wiki technologies have led to VGI. VGI is a special case of user-generated content with explicit or implicit embedded geospatial information. VGI is created by volunteers through crowdsourcing, and represents a new phenomenon arising from Web 2.0 technologies (Goodchild 2007; Sui et al. 2012). VGI is dramatically altering the way in which ordinary citizens can create digital geospatial data. VGI enables both experts and amateur enthusiasts to create and share geospatial information. Complementing the traditional authoritative geographic information collected by governmental agencies or private organizations, VGI has become another popular source of geospatial data for many applications in recent years. For example, there were more than 20 million geographic features in the database of Wikimapia in 2014 (Gao et al. 2014), which is more than many of the world’s largest gazetteers.

VGI is more than just a new type of data; it establishes a new paradigm for continuous monitoring of the changing landscape, behaviors, and social interactions (Jiang and Thill 2015). Many VGI datasets are available at very fine spatial and temporal resolutions. VGI collects data using three main methods: 1) by using geo-aware mobile devices, 2) by annotating geographic features using Geoweb mapping interfaces, and 3) by extracting or inferring location information from ambient geospatial data in social media (such as photos, videos, blog posts, tweets) (Cinnamon and Schuurman 2013; Stefanidis et al, 2013). Examples of VGI include Open Street Map, Geolocated Twitter ‘tweet’ datasets, Wikimapia, Microsoft’s Virtual Earth and Foursquare Venue data.

VGI datasets may have the following characteristics: 1) large in volume, 2) subject to dynamic changes and updates, 3) collected through crowdsourcing architectures using different devices and technologies, and 4) contain a mixture of structured and unstructured information. By making it possible for more people to produce more data in digital form, VGI can dramatically increase the volume of existing geospatial data. With a high and increasing degree of heterogeneity, VGI may create shifts in the content and characteristics of existing geospatial data. Compared with authoritative geospatial data, VGI datasets face challenges such as data quality, accuracy, and validity. However, VGI may be good data sources for augmenting, updating, or completing existing authoritative spatial databases. VGI calls for new procedures and infrastructures for handling distributed big geospatial data.

GIS is a system designed to capture, store, manipulate, analyze, manage, and present all types of geospatial data. GIS has been applied to many different application areas for geospatial
analysis and decision-making such as climate change, disaster response, banking, retail and E-commerce, political campaigns, insurance, and fraud analysis. Currently, several commercial desktop GIS software systems dominate the geographical information (GI) industry, such as Esri ArcGIS, Hexagon Geospatial Geomedia, MapInfo Professional, Global Mapper, Manifold GIS, GE Smallworld, Bentley Map, Golden Software MapViewer, and Clark Laboratories IDRISI. Recently, many free, open-source GIS packages have been developed such as GRASS GIS, gvGIS, ILWIS, JUMP GIS, MapWindown GIS, QGIS, SAGA GIS, and uDig. The development of the World Wide Web creates a unique environment for developing GIS. Besides many of commercial Internet GIS programs such as Esri’s ArcGIS Online, a number of open source Web map servers, such as GeoServer, MapGuide Open Source, Mapnik, MapServer, have been developed to offer better tools for managing geospatial data over the Web. These online GIS programs have freed many users from the need to store large geospatial datasets on their own servers. It is possible to perform basic spatial data analyses and transmit these data back to the office by using these online GIS programs.

However, as the size of geospatial datasets increase, these conventional GIS software packages, which are normally based on single central processing unit (CPU) architecture and single-user interfaces, face tremendous challenges for managing big geospatial data. The existing GIS framework and tools, either desktop or web-server based, do not support (1) shared access to terabyte-scale data, (2) real-time collaborative spatial analysis of massive data, or (3) user-friendly integration between big geospatial data access and spatial analysis (Wang 2010; Wang et al. 2013). Big geospatial data need new GIS tools and software to search, sift, and sieve data from multiple and disparate data sources for spatial analysis. New GIS tools and software should be developed to extend the capabilities of the traditional GIS over new technologies such as the cloud, embedded sensors, mobile and social media, and thus can access disparate big geospatial data in real-time for geospatial data analysis. New GIS tools and systems should support geospatial analysis of big unstructured data in real-time and be able to aggregate terabytes or more geospatial datasets for better understanding of spatial patterns, trends, or relationships.

2. Geospatial Semantic Web

Although the recent developments of the aforementioned geographic information technologies and the proliferation of cost effective and ubiquitous positioning technologies (GPS) have enabled capturing spatially oriented data at an unprecedented scale and rate and have proven useful for gathering information, the increasing volumes of geospatial data present problems for transparent geospatial data exchange and sharing because of heterogeneous semantic problems.

To facilitate the exchange and sharing of geospatial data built on initial expenditures, spatial data infrastructures (SDIs) have been developed in many countries in the past two decades. SDIs based on open standards and OGC web service technologies offer the potential to overcome the heterogeneous problems of legacy GIS databases and facilitate sharing geospatial data in a cost effective way. The fast development of SDIs and OGC web service technologies has undoubtedly improved the sharing and synchronization of big geospatial information across diverse sources.

However, literature about SDI shows that there are limitations in the current SDIs, and it might be still difficult to find proper data from the current SDIs and big data sources. Current SDIs only emphasize technical data interoperability via web services and standard interfaces and cannot resolve semantic heterogeneity problems in big geospatial data sharing. However,
differences in semantics used in diverse big data sources are one of the major problems in big geospatial data sharing and data interoperability. Secondly, with the currently implemented SDIs, it is only possible to search and access geospatial data and services by metadata keywords and it is impossible to directly search and access geospatial data and services based on their contents (Farrugia and Egenhofer 2002). This causes a problem for novice portal users who may not know which keywords to use or may not even know they should try many keywords (Wiegand and García 2007). In addition, a keyword-based search may have a low recall if a different terminology is used and/or a low precision if terms are homonymous (Lutz 2007). On the other hand, the keyword search may sometimes bring an overwhelming number of search results. As a result, users may have to spend a lot of time sifting through undesirable query results before finding the desired data set (Wiegand and García 2007).

Thirdly, although SDIs aim to make discovery and access to distributed geographic data more efficient, the catalogue services currently used in SDIs for discovering geographic data do not allow expressive queries and also do not take into account more than one data source that might be required to answer a question (Lutz and Kolas 2007). It is not possible to automatically discover several data sources that only in combination can provide the information required to answer a given question. However, it is unrealistic to expect that one web service or one data source can fulfill exactly the needs of a user’s request. Therefore, SDIs need a semantic-based approach that can reason about a service’s capability to a level of detail that permits their automatic discovery and composition. In addition, SDIs and the many geospatial portals have large amounts of geospatial data that tend to be only available for download but not for analysis due to the lack of a comprehensive data science environment.

The concept of the Geospatial Semantic Web was proposed to overcome semantic heterogeneity problems of geospatial data (Egenhofer 2002; Zhang et al. 2015). The Geospatial Semantic Web is an extension of the current Web, where geospatial information is given well-defined meaning by applied ontology and thus geospatial contents can be discovered, queried, and consumed automatically by software (Zhang et al. 2015). The Geospatial Semantic Web aims to add computer-processable meaning (semantics) to the geospatial information on the World Wide Web.

Semantic heterogeneity refers to disagreements about meaning, interpretation, or intended use of the same or related data. By associating spatial data content from the Web with ontologies that would supply context and meaning, the vision of Geospatial Semantic Web is to extract geospatial knowledge from the Web regardless of geospatial data formats or sources; thus it can facilitate transparent geospatial data exchange, sharing, and query. For example, the Geospatial Semantic Web will be able to search for needed geospatial information not based on keywords, but on data contents and context, e.g., understanding what the word “near” means based on different spatial data contents and context.
Figure 1 illustrates a Geospatial Semantic Web architecture for geospatial data sharing. For instant remote data access and exchange, the ontology-based web services are used to access and manipulate geospatial data over the Web from heterogeneous databases. The architecture is based on Service-Oriented Architecture (SOA) and is essentially a collection of ontology-based OGC web services, which communicate with each other by simple data passing or coordinating some activities. It has four major elements: service provider, service broker, service client, and ontology server. The service provider supplies the ontology-based geospatial data; the service client searches and integrates the ontology-based geospatial data from the service providers; and the service broker provides a registry for the available ontology-based web services. The ontology server ensures semantic interoperability of ontologies from the service providers and clients.

Based on the work in geospatial ontologies, some recent research focuses on how to query heterogeneous spatial data over the Web through ontologies. Many studies involve the concepts and feasibility of the Geospatial Semantic Web. However, many of these studies are at the initial stage of proposing frameworks for making queries in the Geospatial Semantic Web using the concept of global ontology and local ontology, or ontology integrations. It still is a long way to go to realize the aforementioned goals of Geospatial Semantic Web particular for big geospatial data. There are many challenges to implement a workable Geospatial Semantic Web system. The
major challenges and future directions of the Geospatial Semantic Web in the context of big geospatial data will be discussed in the following sections.

3. Challenges and Future Directions of Geospatial Semantic Web in the context of Big Geospatial Data

3.1. Ontology

The introduction of big geospatial data from diverse sources over the Web makes semantic heterogeneous problem more complex. It becomes a greater challenge to share, integrate, analyze, and manage big geospatial data because of the semantic heterogeneity. An ontology defines and represents the concepts in a domain and uses a shared vocabulary to represent knowledge. When ontologies are populated with valid data, they can provide a knowledge base that supports the analytics of these data. Ontologies are used to resolve the semantic heterogeneous problem over the Geospatial Semantic Web, and ontology development forms the backbone of the Geospatial Semantic Web (Wiegand and García 2007). Ontologies offer the potential to address the semantic challenges presented by big geospatial data.

Ontologies provide the semantic congruity, consistency, and clarity to support big geospatial data analysis and knowledge extraction. Ontologies may make it possible to exploit big geospatial data in the context of their relationships with other existing data. Ontology quality and ontology matching are important for big geospatial data querying and sharing. However, ontology quality and ontology matching remain a challenge for development of the Geospatial Semantic Web. The challenge is further exacerbated with the big geospatial data effect.

Big geospatial data not only introduce efficiency problems for accessing, integrating, and querying ontology data from the Geospatial Semantic Web, but also create visualization problems for communicating and representing large, complex, and varied domain ontologies and knowledge. The high variety and velocity of the big geospatial data means a large number of concepts, properties, and domain knowledge bases. Exploration with the large number of concepts, relationships, and attributions of ontologies with high complexity for big geospatial data is difficult because of the mismatch/gap between users’ or developers’ understanding of large domain ontologies for big geospatial data over the Geospatial Semantic Web.

Ontology quality may be the most important challenge for sharing big geospatial data over the Geospatial Semantic Web. Although some studies have created some specific application ontologies, currently these ontologies are typically built by a small number of people, in most cases by researchers, using ontology tools and editors such as Protégé. For example, The USGS (United States Geological Survey) have been working on developing ontologies for The National Map for many years and has published RDF triple data derived from The National Map to support geospatial knowledge queries (e.g. Varanka 2011; Usery and Varanka 2012; Varanka and Usery 2015). Although these ontology tools and editors supporting ontological modeling have been improved over the last few years and many functions are available now, such as ontology consistency checking, import of existing ontologies, and visualization of ontologies. Manual ontology building for big geospatial data has proven to be a very difficult and error-prone task and becomes the bottleneck of knowledge acquiring processes from big geospatial data. It is especially challenging to build high quality large ontologies for big geospatial data. For instance, it is unrealistic for non-domain-experts to use these tools to build high quality ontologies for a variety of big geospatial data. Although transformation algorithms have been proposed by Zhang et al. (2008) to automatically transform existing Unified Modeling Language
(UML) to Web Ontology Language (OWL) so as to avoid errors and provide a cost efficient method for the development of high quality ontologies, there are many issues yet to be resolved due to the differences between UML and OWL. It is extremely challenging to develop the automatic transform approaches for building high quality large ontologies for big geospatial data.

Ontology matching is another important challenge for sharing big geospatial data over the Geospatial Semantic Web. On the Geospatial Semantic Web, different users communicate and collaborate based on what different ontologies connected with different knowledge expressions mean. The differences between ontologies from varied sources can be handled by ontology matching. Ontology matching is a solution to the semantic heterogeneity problem over the Web by finding correspondences between semantically related entities of ontologies, and is inevitable for ensuring interoperability of big geospatial data. The past several years have witnessed impressive progress in the development of ontology matching tools. However there are many issues waiting to be solved for reliable ontology matching for sharing big geospatial data over the Geospatial Semantic Web.

The first issue is that ontology matching needs to work on a larger scale. In the era of big geospatial data, ontology matching needs fast algorithms to analyze and integrate a large set of ontologies. However, existing ontology matching tools have not demonstrated that they can handle a large set of ontologies. The second issue is that the performance of existing ontology matching tools needs to be improved. For dynamic sharing and utilizing of big geospatial data over Geospatial Semantic Web, performance is particularly important because users cannot wait too long for the Web to respond. However, large ontology matching may become a bottleneck for dynamic big geospatial data applications if the matching techniques perform slowly. The third issue is missing background knowledge. Missing background knowledge is one important reason for the failure of large ontology matching for big geospatial data applications. Ontologies are developed with specific background knowledge and in a specific context. However, the background knowledge and context information may not be available for matching tools. The lack of background knowledge and context information may generate ambiguities and thus increase uncertainties and errors of large ontology matching. Strategies are needed to resolve the missing background knowledge and context information in large ontology matching.

In addition, the lack of sufficient metadata annotations of large ontologies for big geospatial data is also a challenge. While ontologies are important for semantic interoperability and communication among big geospatial data sources, ontologies are always developed by groups or individuals in isolation. There is a lack of metadata annotations of ontologies, which causes difficulty for large ontology matching and sharing. The various ontologies without metadata are usually developed using different techniques. There is no enforced standard convention for describing the contents and context of large ontologies. However, metadata annotations of large ontologies should facilitate ontology discovery and matching for sharing big geospatial data over the Geospatial Semantic Web.

Finally, it is a challenge to provide a dynamic ontology matching support infrastructure at the web scale for sharing and utilizing big geospatial data. The matching life cycle is tightly related to the ontology life cycle: as soon as ontologies evolve, new matching tools have to be produced following the ontology evolution. This may be achieved by recording the changes made to ontologies and transforming these changes into matching processes, which can be used for computing new matching that will update the precious ones.

In general, nowadays web sites are no longer static web pages serving contents and images any more. They have become more responsive, adaptive, and dynamic. There is inherent
uncertainty in the ontology creation, maintenance, and matching processes particularly for sharing and utilizing big geospatial data. The use of newer data formats in big geospatial data (many of which are without schema) makes it harder to use existing ontology creation, matching and alignment techniques for big geospatial data applications. Under these conditions, existing approaches for ontology creation, maintenance, and matching need to be modified and new perspectives for solving this problem. Since the existing approaches based on deterministic assumptions will not perform well in situations that are non-deterministic, probabilistic methods based on approximate sampling of big geospatial data may be explored to overcome this problem.

In dynamic settings of big geospatial data sources, it is natural that data are constantly changing. Thus approaches that attempt to automatically tune and adapt ontology creation and matching solutions to the settings in which an application operates are of high importance. However, it is hard to perform automatic tuning and adapting of large ontologies. It is too cumbersome for one person or a small group of people to resolve the problem. Many people need to work together for creating high quality large ontologies and matching correct large ontologies. Crowd sourcing and other collaborative and social approaches that allow easily sharing and reusing ontologies may be used to aid in large ontology creating and matching.

### 3.2 GeoSPARQL queries

The GeoSPARQL protocol was proposed by the OGC as an extension of SPARQL for querying geographic RDF data. GeoSPARQL queries are dominated by spatial join operations due to the fine-grained nature of the RDF data model. Lack of spatial indices causes additional performance problems for GeoSPARQL queries. One reason for the poor performance problems is caused by the way that spatial attributes are stored in RDF datasets. Spatial attributes are usually stored as string literals that conform to certain formats such as WKT or GML. The GeoSPARQL query engine that implements spatial operators and filter functions has to parse these strings to recover the spatial coordinates for spatial computation. A naïve implementation of a spatial operator or a filter function in GeoSPARQL treats its spatial inputs as plain strings and has to parse the strings to retrieve spatial contents such as x and y coordinates. Repeated parsing of the spatial inputs imposes a very large runtime overhead. The second reason for the poor performance problems is due to the lack of parallelization. Since spatial objects are not indexed, a GeoSPARQL query engine cannot partition ontology data into subsets to be processed in parallel. As a result, a GeoSPARQL query can only be processed as a single-threaded program. Even with pre-computed spatial indices, partitioning spatial ontology data is not easy since the targeted data may not be evenly distributed in the indices.

In fact, different parallel approaches have been widely used for improving the query performance for a long time as reported in literature. However, past research on improving query performance using parallelization has been centered on relational databases (e.g. Boral et al. 1990; DeWitt et al. 1986; Kitsuregawa et al. 1983). Optimizing techniques for parallel relational databases do not specialize on the triple model of RDF and triple patterns of SPARQL queries for query engines based on the RDF- and SPARQL-specific properties (Groppe and Groppe 2011). Although there are studies to query heterogeneous relational databases using SPARQL and parallel algorithms (e.g. Miao and Wang 2009; Castagna et al. 2009), parallel relational databases have inherent limitations such as scalability. A SPARQL query can be parallelized by treating each triple statement in the query as a parallel task and the results of all the triple statement sub-queries can be joined together after all the parallel tasks have completed (Groppe and Groppe 2011). Unfortunately, this approach does not work efficiently when spatial...
predicates exist in the triple statements. There are also studies to propose methods for efficiently parallelizing joint query of RDF data using Map-Reduce systems (e.g. Ravindra et al. 2011).

However, to the best of our knowledge, there are only a few studies that deal with parallelizing spatial join computations to support efficient spatial RDF queries, which is an important issue for the development of a Geospatial Semantic Web (Zhang et al. 2015; Zhao et al. 2015). Zhao et al. (2015) proposed a query strategy to improve the query performance of a geospatial knowledge base by creating spatial indexing on-the-fly to prune the search space for spatial queries and by parallelizing the spatial join computations within the queries. Their initial experiments show that the proposed strategy can greatly reduce the runtime costs of GeoSPARQL queries through on-the-fly spatial indexing and parallel execution. Zhang et al. (2015) introduced a MapReduce based parallel approach for improving the query performance of a geospatial ontology for disaster response. Their approach makes full use of data/task parallelism for spatial queries and focuses on parallelizing the spatial join computations of GeoSPARQL queries. The results of initial experiments show that the proposed approach can reduce individual spatial query execution time by taking advantage of parallel processes. Their proposed approach, therefore, may afford a large number of concurrent spatial queries in disaster response applications.

Although some insights into how the execution of GeoSPARQL queries can be improved through a parallel process have been gained based on the aforementioned studies, these results are limited since they were based on the use of a small- and a medium sized spatial datasets. To implement a workable Geospatial Semantic Web system for big geospatial data for efficient spatial knowledge queries, more studies are needed. Spatial queries from big geospatial data are complex and time-consuming. The geospatial data objects are normally nested and more complex than other data types. They are stored as multi-dimensional geometry objects, e.g. points, lines, and polygons. Spatial queries are based not only on the value of alphanumeric attributes but also on the spatial location, extent, and measurements of spatial objects in different reference systems. Therefore, spatial query processing over big geospatial data requires intensive disk I/O accesses and spatial computation. The I/O and computation capabilities of traditional GIS, VGI, and SDI can hardly meet the high performance requirement of spatial queries or spatial analyses over big geospatial data. While the emerging Key Value Store (KVS) systems, such as Bigtable, HBase, and Cassandra, are proved to be helpful for some I/O intensive applications, these KVS systems cannot process spatial queries efficiently because the data in KVSs are organized regardless of geographic proximity and are indexed by key-based structure rather than a spatial index.

Spatial analysis for big geospatial data involves complex queries such as spatial cross-matching, overlaying of multiple sets of spatial objects, spatial proximity computations between objects and queries for spatial pattern discovery. These queries often involve millions or billions of spatial objects and heavy geometric computations, which not only are used for computing measurements or generating new spatial objects but also as logical operations for topology relationships. Therefore, novel approaches are needed to support efficient parallel execution of GeoSPARQL queries. Future studies may explore how to improve the performance of GeoSPARQL queries in distributed platforms using cloud-based web services and cluster platforms. Future studies should focus on studying 1) how to implement extensions to the RDF query engine (such as Jena) to build and cache spatial indices on-the-fly, and 2) how to utilize distributed and paralleled computing resources including computing clusters (through libraries such as Spark) and Graphic Processing Units (GPU) to accelerate GeoSPARQL.
3.3. Geospatial indexing

Performing a spatial query of a knowledge base of the Geospatial Semantic Web can be very inefficient if it contains a large number of spatial objects. Spatial indexing algorithms such as R-tree (Guttman 1984), Quad-tree (Samet and Webber 1985), and KD-tree (Bentley 1975) are used in traditional spatial databases to improve query performance. These methods may be used to improve query performance over the Geospatial Semantic Web in the context of big geospatial data. With spatial indexing, spatial queries of big geospatial data over Geospatial Semantic Web can be answered more efficiently since the computation involves far fewer spatial objects.

There are three important spatial indexing algorithms in literature: R-tree, Quad-tree, and KD-tree. All three algorithms can be used to reduce the runtime costs of range queries, spatial joins, and nearest neighborhood (kNN) queries of geospatial objects. However, Quad-tree and KD-tree may be more suitable for indexing point objects while R-tree and its variations work well for all types of spatial objects including lines and polygons.

R-tree and its variations are the preferred algorithms for spatial indexing (Sellis et al. 1987; Beckmann et al. 1990). R-tree performs well for all kinds of spatial objects including points, lines, and polygons. It is a height-balanced tree where each spatial object is inserted into the tree leaf using its minimum bounding rectangle (MBR) as a guide. Each node of a R-tree has a MBR that encloses the MBRs of its children. The number of the children of each node is maintained within a range so that if there is an overflow of children at a node, then the node is split into two, and if there is an underflow, then two or more nodes are merged. If the insertion of a spatial object causes the root node to split, then the tree will grow by adding a new root.

R-tree is a dynamic tree whose shape depends on the order in which the spatial objects are inserted. The MBRs of the children of a node may also overlap so that search operations may require recursive descend into multiple children of a node. Various heuristics can be applied to sub-tree insertion and node splitting.

Quad-tree is a tree data structure where each tree node has four children with each child representing a quadrant in a two dimensional space. Quad-tree for points can be constructed by recursively adding tree nodes until each leaf contains at most one point. To search a Quad-tree for the points within a range, the tree nodes can be recursively searched and only quadrants that intersect the range will be visited. For each visited leaf quadrant, the points of the leaf that are within the range will be returned.

The height of a Quad-tree depends on the smallest distance between two points. The time complexity of searching the nearest neighbor of a point using a Quad-tree is linear to the depth of the tree. Quad-trees may not be balanced, however, and the shape depends on the distribution of spatial objects. In extreme cases, the depth of a Quad-tree can be linear to the number of points. The advantage of the Quad-tree approach is its simplicity and it can be efficient for evenly distributed points.

A common type of Quad-tree is the region Quad-tree, which divides the space into four equally-sized quadrants. Region Quad-trees are suitable for indexing points. There are other types of Quad-trees (with more complexity) for indexing lines and polygons.

KD-tree is a binary tree with a k-dimensional point in each node. For a node at depth $d$, the dimension $d \mod k$ of the node's point is used as the key to separate the node's sub-trees. A value at each dimension (such as the medium value of the data points) needs to be chosen for data separation. Finding one nearest neighbor in a balanced KD-tree of randomly distributed points takes log-time on average. However, in general, multiple sub-trees of each node may be
explored in a nearest neighbor search. To search a KD-tree for points within a range, the splitting hyperplane may be used to decide which sub-trees to visit.

There are some challenges in introducing spatial indexing to the Geospatial Semantic Web such as how to retrofit an ontology query engine to build spatial indices and how to take advantage of the spatial indices in answering spatial queries. An ontology query engine such as Jena can be modified to build spatial indices when spatial ontology data are initially loaded. To implement this, the query engine should provide means to identify the ontology class whose instances are spatial objects that should be indexed. In addition, when new ontology instances are added or updated, the related spatial indices should be updated as well. Extensions to spatial ontology constructs may be needed to ensure that the ontology data are written in formats that can be parsed and indexed by the query engine.

Answering ontology queries using spatial indices is more intricate since each query may or may not involve spatial attributes that can be processed with indices. There are several possible implementation. One choice is to implement low-level extensions to the ontology query engine to intercept calls to spatial functions or predicates of a query and generate results using spatial indices if available. Another choice is to analyze the entire query first, separate the spatial components of the query, and then attempt to answer it with spatial indices. The third choice is to build spatial indices on demand when answering queries such as those that contain operators corresponding to spatial joins, and cache the indices for subsequent use.

3.4. Spatial Join

Queries over the Geospatial Semantic Web may require joining two or more types of spatial objects. In spatial databases, spatial join algorithms are used to improve runtime performance and the same algorithms can also be applied to improve query performance over the Geospatial Semantic Web. The choice of these algorithms depends on whether one or more spatial indices are present.

**Nested loop** is the simplest approach to join two sets of spatial objects and it works with or without spatial indices (Mishra and Eich 1992). It simply takes one from each of the two sets of spatial objects via nest loops to check which pairs satisfy the join conditions. The set of objects that are indexed should be placed in the inner loop so that the index can be used to check the spatial join relation for each object of the outer loop. The advantage of this approach is that it is very simple, does not require indices, and is efficient for small sets of spatial objects. This algorithm can be easily parallelized. The disadvantage is that it can be very slow for large data sets where the spatial relation is costly to compute.

**Hierarchical traversal algorithms** can be used if both sets of objects are indexed using R-tree or similar data structures (Brinkhoff et al. 1993). The basic algorithm takes two sets of tree nodes as inputs. For each pair of nodes with one from each set, it compares the MBRs of the nodes to determine whether or not they are disjoint. If they are not, then the algorithm recursively calls itself with the children of the non-leaf nodes as inputs. The recursion stops when both nodes are leaves and returns the pair of leaves that are related by the spatial join predicate. This approach is very efficient when the MBRs of an R-tree do not have many overlaps. Like a nest loop, this algorithm is parallelizable. The drawback is the requirement of available indices on both sets of input objects, which may not be possible when the input sets are dynamically generated.
The plane sweep algorithm does not use spatial indices (Arge et al. 1998). The algorithm works by sweeping along one dimension, e.g., x-dimension, so that if a MBR of a set is detected, it becomes active and is inserted into a sweep structure for that set. Once the sweep line passes a MBR, it is removed from the corresponding sweep structure. Each MBR in one sweep structure is compared with the active MBR of the other set to find out whether or not they are spatially related using other dimensions (e.g. y-dimension). The plane sweep algorithm works for smaller data sets that can be fitted into memory. The advantage of the approach is that it does not require indices and it is more scalable than the nested loop algorithm when the objects are evenly distributed. The drawback is that it requires all objects to fit in memory for efficient processing. Therefore, its applicability is limited by the size of the memory and the sizes of the data sets.

For larger data sets, the partition-based spatial merge-join algorithm may be used. This algorithm recursively partitions the pair of input data sets into pairs of smaller sets until each pair can fit in memory that the plane-sweep algorithm can be applied. The partition can be based on a grid but MBRs that intersect grid lines must be duplicated.

To improve performance, the spatial join process can be split into two steps: filtering and refinement. The filtering step approximates the spatial join using the MBRs to obtain a set of candidate pairs and the refinement step checks spatial relations of the candidate pairs to obtain the final result. The filtering step processes more pairs of spatial objects but it is efficient to compute join relations for MBRs. The refinement step spends more time on computing the join relations of candidate pairs using their exact geometries though it involves a smaller number of spatial object pairs. The aforementioned spatial join algorithms can be used for the filtering step.

3.5. Distributed Geospatial Computing

Spatial computation such as spatial indexing, KNN (K-nearest neighbors) searching, range query, and spatial joins is very expensive for large spatial data sets. To improve performance, several systems have been developed to implement spatial computation over distributed platforms such as computing clusters and cloud-based servers. Two of these systems -- Hadoop-GIS (Aji et al. 2013) and SpatialHadoop (Eldawy and Mokbel 2015) are directly based on the Hadoop framework, which provides distributed file system that supports MapReduce computation over a large number of computing nodes. The other two systems, GeoSpark (Yu et al. 2015) and SpatialSpark (Zhang et al. 2015) are based on a more recent framework Spark, which provides better performance than Hadoop due to Spark's in-memory capability. The crucial abstraction in Spark is Resilient Distributed Dataset (RDD), which is used to hold distributed data.

All the aforementioned four systems have implemented spatial computation such as KNN, range queries, and spatial join. All systems provide distributed spatial indexing capability, which can greatly improve the performance of some spatial operations. SpatialHadoop supports spatial indexing as an integrated component of the Hadoop distributed file system, which can optimize skewed spatial data distributions. However, GeoSpark has better performance than SpatialHadoop over spatial joins with or without indexing. It is unclear how SpatialSpark compares with other systems due to its limited number of implementation. The challenging issues with implementing distributed spatial computation focus on distributed indexing of spatial data on the computing nodes and revision of sequential spatial algorithms to use the map-reduce framework. Memory consumption of Spark-based systems is also a challenging issue. There are at least four choices with spatial indexing over the Map-Reduce framework. The
second choice is to build an index on-demand. The third choice is to build a global index on the distributed datasets. The last choice is to have both a global index and a local index for each local dataset. Hadoop-GIS builds global indices but has local indices on demand while SpatialHadoop builds both global and local indices. GeoSpark builds global indices but local indices are based on needs. When partitioning data over spatial indices such as R-tree, one must consider spatial objects across the boundaries of the partitions. A common solution is to replicate the spatial objects for all partitions with boundaries that overlap the objects. However, the subsequent computation is needed to remove the duplicated objects from the results. Implementing range queries, KNN, and spatial joins on map-reduce frameworks seems straightforward with or without spatial indices as the algorithms can be made data parallel. These frameworks demonstrated that spatial computation can be made scalable with respect to the number of computing nodes. This makes it possible to support spatial computation on very large spatial datasets of Geospatial Semantic Web.

Other than map-reduce based frameworks, GPUs, multi-core CPUs, and Vector Processing Unit (VPU) have been used in accelerating spatial computation in the literature. Examples include building R-tree indices and using R-tree indices for spatial querying (You et al. 2013), point in polygon, point to polygon computation (Zhang and You 2014), spatial join (You et al. 2015), and spatial-temporal aggregation (Zhang et al. 2014). The experiments in these studies have revealed the potential of many-core processing units such as GPU and VPU in reducing the runtime cost of certain data-parallel geospatial computation. The main burden of taking advantage of the computing power of GPU and VPU is the more complex programming interface provided by these platforms when compared with map-reduce frameworks. There is also the cost of copying data between CPU and GPU memory, which, if occurs frequently, can substantially diminish the performance gain.

While the aforementioned researches have demonstrated the effectiveness of scalable spatial computation over distributed systems, more work is needed to improve query performance of large geospatial data sets over the Geospatial Semantic Web. The primary challenge is how to distribute the geospatial data within a knowledge base over computing nodes to enable global and local spatial indices and to distribute the spatial computation workload. One possible research direction is to have virtual representation of spatial objects within a geospatial knowledge base while storing the concrete data in original distributed servers. This requires modification to ontology query engine to redirect the spatial computation of a query that involves virtual spatial objects to the distributed servers and to integrate the results with that of the rest of the query. Another possible direction is to partition the geospatial knowledge base over distributed servers, perform ontology query over each computing nodes, and then integrate the results. The non-spatial portion of the knowledge may need to be replicated across the computing nodes so that the results of the ontology query at each computing node are completed except the geospatial data, which needs to be joined.

4. Conclusion

Modern geospatial technologies such as remote sensing, GPS, GIS, and VGI have been used to collect and handle geospatial data. In recently years, these geospatial technologies have generated large volumes of geospatial data about things and human beings. Although recent development of Geographic Information and Internet technologies have enabled accessing and sharing big geospatial data, we still face huge challenges for transparent geospatial data exchange and sharing because of the lack of fast and interoperating algorithms and tools along
with heterogeneous semantic problems. The concept of a Geospatial Semantic Web was proposed to overcome semantic and spatial computation limitations. However, there still is a long way to go to support efficient query of big geospatial data over the Geospatial Semantic Web. In this book chapter, we investigate the challenges and opportunities of Geospatial Semantic Web brought for sharing and utilizing big geospatial data. Ontologies provide the semantic congruity, consistency, and clarity for support of big geospatial data analysis and knowledge extraction. However, ontology quality and ontology matching remains a challenge for development of the Geospatial Semantic Web. The challenge is further exacerbated with the big geospatial data effect. It is very inefficient to query big geospatial data using the standard Semantic Web query language—Geo-SPARQL because the data in the knowledge base of the Geospatial Semantic Web are no longer indexed to support efficient big geospatial data query. A new query strategy is needed to reduce the runtime costs of Geo-SPARQL query through on-the-fly spatial indexing and parallel execution. Three important spatial indexing algorithms in the literature-- R-tree, Quad-tree, and KD-tree-- can be used to improve query performance from big geospatial data sources over Geospatial Semantic Web. However, Quad-tree and KD-tree may be more suitable for indexing point objects while R-tree and its variations may work well for all types of spatial objects including lines and polygons. To get the needed information from big geospatial data sources over the Geospatial Semantic Web, spatial queries may require joining two or more types of spatial objects. The spatial join algorithms for spatial databases may be used to obtain the needed information from big geospatial data sources over the Geospatial Semantic Web. The choice of these algorithms depends on whether one or more spatial indices are present. To improve performance, several systems such as Hadoop-GIS, SpatialHadoop, GeoSpark, and Spatial Spark have been developed to implement spatial computation over distributed platforms such as computing clusters and cloud-based servers. Each of these implemented distributed geospatial computing system has its own advantages and disadvantages. However, the challenge still exists to implement distributed spatial computation for sharing and utilizing big geospatial data over the Geospatial Semantic Web.

In summary, although big geospatial data are available for better understanding our coupled human and environmental systems, there are still many challenges of Geospatial Semantic Web techniques for sharing and utilizing big geospatial data. New methods for the Geospatial Semantic Web and distributed spatial computing are needed to make full use of available big geospatial data.

References:


• You, S., Zhang, J., & Gruenwald, L. (2015, April). Large-scale spatial join query processing in cloud. In Data Engineering Workshops (ICDEW), 2015 31st IEEE International Conference on (pp. 34-41). IEEE.


• Zhang, J., You, S., & Gruenwald, L. (2015). Large-scale spatial data processing on GPUs and GPU-accelerated clusters. SIGSPATIAL Special, 6(3), 27-34.