A Fully Distributed Scheme for Discovery of Semantic Relationships

Juan Li, *Member*, *IEEE*, Hui Wang, *Member*, *IEEE*, Samee Ullah Khan, *Senior Member*, *IEEE*, Qingrui Li, *Student Member*, *IEEE*, and Albert Y. Zomaya, *Fellow*, *IEEE*

Abstract—The availability of large volumes of Semantic Web data has created the potential of discovering vast amounts of knowledge. Semantic relation discovery is a fundamental technology in analytical domains, such as business intelligence and homeland security. Because of the decentralized and distributed nature of Semantic Web development, semantic data tend to be created and stored independently in different organizations. Under such circumstances, discovering semantic relations faces numerous challenges, such as isolation, scalability, and heterogeneity. This paper proposes an effective strategy to discover semantic relationships over large-scale distributed networks based on a novel hierarchical knowledge abstraction and an efficient discovery protocol. The approach will effectively facilitate the realization of the full potential of harnessing the collective power and utilization of the knowledge scattered over the Internet.

Index Terms—Distributed systems, distributed discovery, Semantic Web, semantic relation

1 Introduction

With the development of Semantic Web technologies, more and more Semantic Web data are generated, which is being used in web applications and enterprise information systems. To effectively utilize the large amount of semantic data, efficient search mechanisms customized for Semantic Web data, especially for ontologies, have been proposed for both humans and software agents. For instance, the semantic search [11] scans objects to capture instances in a given data set. By utilizing keywords, the Swoogle search engine [10] retrieves semantic entities as uniform resource identifiers (URI). To support complex queries over resource description framework (RDF) bases, query languages, such as SPARQL [58], have been used to express various restrictions on semantic entities and relationships. The technologies listed above effectively assist users (human and software agents) to locate desirable information from large amount of semantic data on the web. However, hidden knowledge of great potential value in large semantic data sets may not be discovered by the aforementioned approaches. For instance, one such type of knowledge is to determine the complex relationships between multiple semantic entities, i.e., "how resources X and Y are related."

- *J. Li and Q. Li are with the Computer Science Department, North Dakota State University, Fargo, ND 58108-6050. E-mail: {J.Li, qingrui.li}@ndsu.edu.*
- *H. Wang is with the Department of Computer Science, Stevens Institute of Technology, Castle Point on Hudson, Hoboken, NJ 07030. E-mail: Hui.Wang@stevens.edu.*
- *S.U. Khan is with the Electrical and Computer Engineering Department, North Dakota State University, 1411 Centennial Blvd, Fargo, ND 58102. E-mail: samee.khan@ndsu.edu.*
- *A.Y. Zomaya is with the School of Information Technologies, University of Sydney, Building J12, Sydney, NSW 2006, Australia. E-mail: albert.zomaya@sydney.edu.au.*

Manuscript received 9 Aug. 2011; revised 15 Mar. 2012; accepted 29 Apr. 2012; published online 8 May 2012.
For information on obtaining reprints of this article, please send e-mail to: tsc@computer.org, and reference IEEECS Log Number TSC-2011-08-0078.
Digital Object Identifier no. 10.1109/TSC.2012.16.

Automatic discovery of semantic relationships between entities is a key issue in analytical domains, such as business intelligence, and homeland security, where "...the focus is on trying to uncover obscured relationships or associations between entities and very limited information about the existence and nature of any such relationship is known to the user..." [1]. In literature, we find many applications that are built on the aforementioned types of association. For example, detecting conflict of interest (COI) relationships among potential reviewers and authors of scientific papers [33] and detecting a connection between two suspected passengers on the same flight in the context of aviation safety [34].

Most of the prior work (e.g., [1], [33], and [34]) on semantic association query and discovery assumed that there is a global data set, where all of the entities and relationships are available for analysis. Because Semantic Web data are created and stored by different organizations or individuals that are geographically distributed, such an assumption is impractical. Moreover, moving and merging these data sources to a central location require transmittal of terabytes of data sets over long distances. Furthermore, some repositories may not be allowed to be merged either for legal reasons, for the risk of revealing business secrets, or for posing other social challenges. On the other hand, analyzing a local knowledge base can only obtain limited knowledge that is constricted by spatial and temporal constraints. Therefore, there is an increasing demand and interest to collectively discover knowledge from distributed sources. In light of the aforementioned discussion, new challenges (that we outline below) have emerged:

- Because of the lack of a global view or unified understanding of the distributed semantic data, it is difficult to achieve global optimum of semantic association discovery with dispersed local operations.
- The relationships between two entities may span over multiple distributed knowledge bases, which

requires efficient discovery protocols to forward search requests between knowledge bases and later to gather the search results.

- Because of the complexity of semantic-relation queries, it may be difficult to achieve scalability and low latency when considering large-scale distributed systems.

To the best of our knowledge, there is only one previously reported work [15] on discovering semantic relationships in distributed environments. That work presents a superpeer-based approach to discover semantic relations in a peer-to-peer (P2P) network environment. In the proposed system, peers register with superpeers that are also peers but likely to be more powerful in terms of computing capacity, memory storage, and bandwidth. Superpeers are connected with each other through semantic links. The system is built on the assumption that each superpeer knows how to reach other superpeers. Therefore, relational discovery can be performed by finding semantic paths at the superpeer layer. However, it was unspecified: 1) How do the related nodes can locate the same superpeer? 2) How do superpeers communicate in the network? As a consequence, it is difficult to guarantee the scalability of the system without the design of aforementioned communication components.

In this paper, we focus on a discovery system that breaks the traditional barriers of the centralized scheme into the realm of decentralized and distributed strategies. Our work aims to support the discovery of semantic relationships over geographically distributed knowledge bases on an unprecedented scale. The proposed approach is fully decentralized and scalable. Moreover, it not only efficiently solves the semantic relation discovery problem, but also improves the traditional search and discovery of semantic knowledge. Therefore, our proposed methodology also improves the effectiveness and efficiency of semantic sharing, in general.

The rest of the paper is organized as follows: In the following section, we detail the background knowledge. Section 3 provides an overview of the discovery system framework and presents some preliminary techniques. Section 4 describes the design of the discovery system. In Section 5, we evaluate the proposed methods and show their effectiveness with a comprehensive set of simulations. Related work and concluding remarks are provided in Sections 6 and 7, respectively.

2 Background

The RDF is a World Wide Web Consortium (W3C) recommendation for describing web resources. The RDF provides a basic data model, such as the entity-relationship model for writing simple statements about web objects. The RDF can make statements about resources in the form of *subject-predicate-object* expressions, termed *triples* in the RDF terminology. The *subject* denotes the resource that has a URI. The *predicate* denotes traits or aspects of the resource and expresses a relationship between the *subject* and *object*. *Predicates* in the RDF also are identified by URIs. The *object* is the actual value that can either be a resource or a literal.

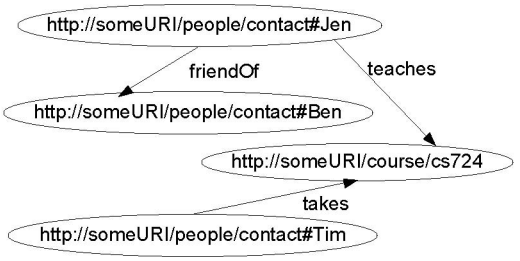


Fig. 1. Example of part of an RDF graph.

The RDF also can represent simple statements about resources as a directed labeled graph with typed edges and nodes. In this model, a directed edge labeled with a property name connects the *subject* to the *object*. For instance, the group of statements, "There is a person identified by *http://someURI/contact#Jen*. She is a friend of another person, called Ben, and she teaches a course CS724 that is taken by a student, called Tim," could be represented as the RDF graph that is depicted in Fig. 1.

As depicted in Fig. 1, paths of the RDF graph represent semantic relationships among the participating resources (entities), explicitly or implicitly. The work reported in [1] proposed a ρ-path query as a way of expressing semantic associations between entities in the RDF graph. A path $\rho = e_1, p_1, e_2, p_2, e_3, \ldots, e_{n-1}, p_{n-1}, e_n$ is a sequence of RDF statements, where each triple e_i, p_i, and e_{i+1} represents a single statement in which p_i is the predicate and one of e_i or e_{i+1} is the subject and the other is the object. As opposed to the idea reported in [1] in which a ρ-path is defined as a directed path, in this paper, we treat paths as undirected routes for the following reasons: 1) As pointed out in the work reported in [49], it is difficult to generally determine which relationships are of relevance in a certain situation and which ones are irrelevant. Each relationship may be deemed valuable in a certain situation and useless in another situation. Therefore, our discovery mechanism aims to find as many relationships as possible. The filtering of the search results needs to consider the context and preferences of the user or an application, which is beyond the scope of this paper. Therefore, in this research, two resources x and y are said to be ρ-path *associated* if there exists an undirected path ρ of length $n > 0$ between them.

3 Design Framework

In this section, we present the basic idea of the discovery system and the techniques that enable the relational discovery within a distributed environment.

3.1 Design Overview

Because path discovery needs to locate not only the entities but also all of the paths connecting them, it is considered much more difficult than the entity discovery mechanisms. Our proposed solution is inspired by the strategy of path planning [4], [26], [27]. Consider the following analogous problem of traveling by a car from Seattle, Washington, USA to Montreal, Quebec, Canada. More specifically, the problem set is: "How to travel from 1000 Union Street in Seattle, Washington to 2000 Avenue Du Parc in Montreal, Quebec?" Given a detailed road map of North America, it

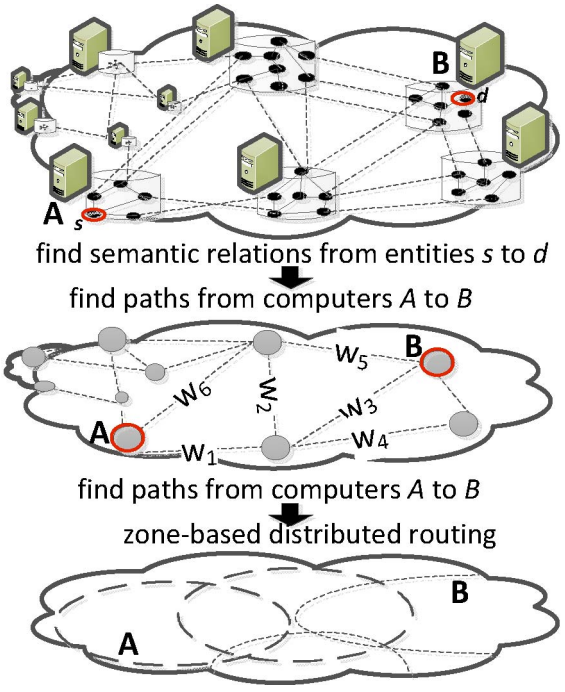


Fig. 2. An overview of the abstraction-based approach.

may be a computationally expensive exercise to compute the travel route with all possible roads annotated with driving distances. Therefore, instead of working at such a low level of detail, a human travel planner would prefer to do the route planning at the state (province) level first. That is to say, once "the path" reaches a state boundary, compute the best route from state to state. For the aforementioned example, a human traveler would probably prefer to do the planning in three steps: 1) Travel from 1000 Union Street in Seattle, Washington to a major highway leading out of Seattle, Washington. 2) Plan a route from Seattle, Washington to Montreal, Quebec. 3) Travel from the incoming highway in Montreal, Quebec to 2000 Avenue Du Parc in Montreal, Quebec. Steps 1 and 3 would require a detailed road map of each of the cities. Step 2 could be completed with a high-level map, with roads connecting states, abstracting away all of the minor details.

We adopt a similar abstraction strategy (as discussed above) for semantic relationship discovery. As shown in Fig. 2, instead of starting from millions of semantic entities and relationships at the lowest level, we consider each knowledge base that contains multiple entities and relations as an abstract unit. We assume that each peer hosts an individual knowledge base. We treat these knowledge bases as black boxes and ignore the detailed semantic entities and the corresponding relations within the knowledge bases. Thereafter, through ontology mapping (or integration) [50], [51], [52], [53], [54] or interlinking with the same URIs, we can connect knowledge bases to form a graph at a higher level. The knowledge base-level semantic graph will act as the blueprint of our distributed semantic knowledge bases. As a consequence, the semantic path discovery problem is analogous to the route planning problem. The aforementioned abstraction may drastically reduce the size of a potentially huge search space. However, locating multiple paths at the knowledge base level is still a challenging problem due to the large number of such knowledge bases and their distributed nature. To further improve the efficiency of path discovery at this level, we propose a novel discovery protocol that constructs zones on top of the knowledge bases. Built at the top of the knowledge base-level semantic graph, the zone layer uses a network routing strategy to facilitate path discovery.

With the different levels of semantic graph constructed, the path-finding problem is constricted to the following three steps: 1) Locate the source and destination entities at the entity-level graph. 2) Search at the level of peers or knowledge bases for paths that originate from the source peer (containing the source entity) and end at the destination peer (containing the goal entity). 3) Retrieve the semantic path at the level of entities. Such an abstraction equates a much faster search. To efficiently locate the source and destination semantic entities at the entity level, we adopt a distributed hash table (DHT)-based overlay. Such an overlay is used to index the entity-level semantic graph, which allows efficient identification of semantic entities. The main focus of our work is to discover semantic paths between semantic entities. The basic idea is to create different levels of abstractions and apply appropriate discovery scheme on each of them. In the subsequent text, we present each part of the relation discovery strategy in detail.

3.2 Semantic Graph Formation

To link the dispersed knowledge bases to form a connected semantic graph, we propose a semantics-based topology adaptation scheme to connect the knowledge bases containing similar semantic properties and facilitate the establishment of semantic mappings or links. The foundation of this scheme is a metric that measures peers' semantic similarities. In one of our previous studies [19], we extended the previously reported distance-based approaches (e.g., [18], [16], and [25]) to accurately measure the semantic similarities between ontologies. Our proposed approach extends the previous approaches by supporting multiple ontologies and improves the accuracy by integrating factors, such as the depth of a node in the ontology hierarchy and the type of links. After semantically related knowledge bases have been located, mapping or linking can be established between these knowledge bases or in some scenarios related knowledge bases can be merged.

The topology adaptation is a process of finding semantically related neighbors. This process can be performed at different states of the life time of a node. That is: 1) on joining the network, a node may choose neighbors based on their semantic similarities, or 2) after updating its knowledge base, a node will reevaluate its neighbor relationships to connect to most related peers. This methodology ensures that the underlying node topology will always reflect the changing interests and data contents. The aforementioned process can be achieved by peer-to-peer-based neighbor discovery and overlay formation. We encourage the readers to find more details about our proposed topology adaptation scheme by referring to our previous studies, such as [18] and [20].

3.3 Semantic Entity Location

Another task for semantic path finding is to efficiently locate specific semantic entities (e.g., the source and destination entity) in the entity-level semantic graph. To fulfill this task, we construct indexes on entities. As mentioned above, entities are subjects and objects in the RDF triples. Triples (in distributed knowledge base) that share common entities (i.e., the same subjects and/or objects) should be indexed together in one of the distributed peers so that they can be located at a later stage. The challenge in this scenario lies in assigning "index rendezvous points" for entities. To avoid bottlenecks (courtesy of centralization), we use a DHT overlay to provide a decentralized and scalable rendezvous for the RDF triple entities. Each triple is sent to two rendezvous peers based on the subject and object, respectively. This ensures that the triples with common subjects and/or objects will always be colocated. Unlike the data indexing of RDFPeer [8], we do not index predicates (i.e., edges of the semantic graph). This is due to the fact that normally, we only need to locate entities of the semantic graph and not the edges. We store each triple twice by applying a hash function to its subject and object. The insertion operation of a triple t is performed as follows:

$Insert(t) \equiv Insert(SHA1Hash(t.subject),t),\ Insert(SHA1Hash(t.predicate),t),\ Insert(SHA1Hash(t.object),t).$

For example, the statement $t:\{<Billy>,<teaches>,<cs213>\}$ is first indexed by *subject*, and then sends the following message to the overlay:

$Insert\ \{key,\{(\text{“}subject\text{”},<Billy>),$
$(\text{“}predicate\text{”},<teaches>),$
$(\text{“}object\text{”},<cs213>)\}\},$
$where\ key = SHA1Hash(\text{“}<Billy>\text{”}).$

In the above message, the first attribute-value pair *("subject," <Billy>)* is the routing key pair, and the key is SHA1 hash value of the subject value. Similarly, the triple is indexed by object as well. The DHT indexing guarantees that the entities can be located within $log(N)$ hops, where N is number of peers in the Semantic Web.

4 Semantic Path Discovery

With the semantic graph created and both source and goal entities located, the next step is to locate paths between the source and goal entities. This is the topic of the subsequent sections.

4.1 Path Cost at the Peer Level

To discover path at the peer level, the optimal distances for crossing each peer must be precomputed and cached. The optimal distances are computed based on the entity-level graph. For simplicity, we assume that each semantic edge connecting two semantic entities has a unit cost. In reality, we must distinguish edges and consider factors, such as the context of the query, the relevance between the neighboring entities, and the trust between knowledge base owners. The above-mentioned factors can be handled by adding weights to semantic edges without further modifying our proposed algorithm. Each peer records a set of distances (in terms of number of semantic edges in the entity-level semantic graph) before they can be treated as a "black box." The distance that matters the most is the shortest distance between knowledge bases. As depicted in Fig. 3, knowledge base A is linked to knowledge bases B, C, and D though common URIs or though the ontology mappings (e.g., *equivalentTo* or *seeAlso*). The entities in A that are mapped to other knowledge bases are called the *gateway* nodes of A. For example, G_{AB}, G_{AC}, and G_{AD} are knowledge base A's gateways to the neighboring knowledge bases. Knowledge base A will only record the shortest distances between the gateways to the adjacent knowledge bases. As it will be further explained in Section 4.2.4, the shortest distance between gateways will be used as an upper bound to prune semantic paths beyond a predefined limit of k at the knowledge base level. In Fig. 3, one can see that from A's local knowledge base $min\_dist\ (G_{AB}, G_{AC)} = 2$ (i.e., the minimum cost of the path from B to C via A is 2), $min\_dist\ (G_{AB}, G_{AD}) = 3, min\_dist(G_{AC}, G_{AD}) = 4$. With the minimum costs between gateways defined, we can ignore the details inside each peer's knowledge base and treat knowledge bases as nodes and the gateway costs as the cost of edges between nodes.

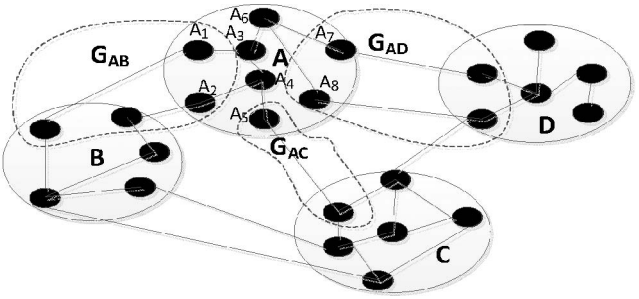


Fig. 3. Connection of knowledge bases and their gateways.

Given two entities, there may be multiple paths between the entities representing different semantic relations. As shown by the small-world theory [3], any two persons in the world can be connected by a link of six hops. The above also applies to other relationships as well, such as the relationships between two diseases, two genes, or a gene and a disease. Therefore, it is not necessary to find a path of very long length. Indeed, the relationship between two entities connected by a long path is vague and difficult to explain and understand [49]. Therefore, in our study, we set a length limit k, and only consider finding semantic paths with less than k semantic edges. In our experiment, we set the default value for k to be equal to 6.

4.2 Semantic Zone Discovery Protocol

Using abstraction at the peer or knowledge-base level reduces the search space. However, path discovery at this level is still very challenging. Unlike the route planning problems in which only one shortest path between a pair of addresses is returned, our path discovery problem must return multiple paths. Moreover, the network is dynamic not only because peers may join or leave at all times but also because peers may add or update their knowledge base, which consequently may change the underlying network topology. To address these challenges, we propose

a novel decentralized and scalable discovery protocol, termed the semantic zone discovery protocol (SZDP).

4.2.1 Overview

To find paths between two nodes, the most straightforward way is to flood: 1) the query from the source to the network until the query reaches the destination or 2) from both the source and the destination until their boundaries meet. However, flooding creates an enormous amount of network traffic, and the latency experienced by the end users is unacceptable. In fact, all on-demand discovery (i.e., discovery when needed) approaches adopt some forms of flooding. Although on-demand discoveries may utilize some forms of optimizations methodologies, such as topology adaptation, adaptive time to live (TTL), and flow control [7], [6], [21], such methods still suffer from the long latency and the associated large overhead. To reduce latency and query traffic, routes must be planed beforehand. We call such kind of discovery as *proactive discovery*.

Two traditional routing schemes, distance vector routing and link-state routing, belong to the proactive discovery scheme. In distance vector protocols, a node exchanges with its neighbors a vector containing the current distance information to all of the known destinations. This approach works well for locating shortest paths between two nodes. However, maintaining multiple paths between nodes require huge discovery tables, which are impossible to scale given large-scale networks. Therefore, we cannot use distance vector-based discovery scheme to locate multiple paths. On the other hand, in the link-state protocols, each node disseminates the status of each of its links throughout the network via flooding in the form of link-state updates. Each node locally computes routes in a decentralized manner using the complete topology information. Link-state protocols do not record paths between each node pair. Therefore, such protocols do not need a large discovery table as required by distance-vector protocols. However, due to flooding, link-state protocols cannot scale well given a large-scale network. To solve the above stated problem, we divide the network into zones based on the peer hops and use link-state protocol inside of the zones. The semantic paths are likely to be located inside the zone of the querying node. This is because of the semantic locality property of the semantic topology [43].

Our proposed discovery protocol, SZDP, is inspired from the zone routing protocol (ZRP) [12] for mobile ad hoc networks. The SZDP protocol must be considered different from the ZRP protocol because ZRP only needs to locate one shortest path from source to destination, while SZDP must locate multiple paths to discover more than one relationship between the entities. The cost computation of SZDP also is different from ZRP. Unlike ZRP, SZDP does not use the hop count as the cost but considers the cost of the paths between gateways.

4.2.2 Architecture

The SZDP, as the name implies, is based on the concept of semantic zones. Each node defines its own semantic zone, which is the neighborhood area in terms of number of peer hops, termed *radius* (r). The zones of the neighboring nodes may overlap. An example of a semantic zone is depicted in Fig. 4, where the discovery zone of A includes all of the nodes within $r = 2$. We must note that the nodes in the graph are in fact gateways to the peers and the costs on the edges are the gateway distances. For simplicity, readers can interpret them as distances between peers. (Note that the value of r is predefined by the application.) The setup value of r is intriguing: It should not be set too large, which will cause large amount of maintenance overhead; while it should not be too small, because the zone may not contain enough discovery information to find all of the paths. Ideally, each node must maintain a zone that is manageable by the node, and at the same time, it enables discovering most of the important semantic relations within the zone. For example, in Fig. 4, if the query's semantic path limit k is set to be not more than 6, then the query can be resolved within A's local zone. This is due to the fact that all of the nodes whose shortest distance to A is less than k are within A's zone, i.e., a radius of 2 can cover all of the paths within the set limit of 6. It is noteworthy to mention that the radius r and semantic path limit k are two different concepts: r is the number of peer hops (at the peer or knowledge base-level), while k is the number of semantic edges (at the semantic entity level).

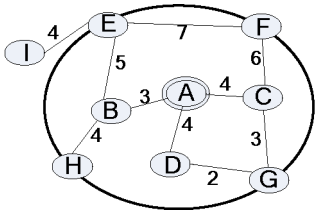


Fig. 4. Example discovery zone with radius $r = 2$ (nodes: gateway of peers; edge cost: gateway distance).

4.2.3 Semantic Zone Discovery Protocol

The SZDP is a limited-depth link-state discovery protocol. In SZDP, each node maintains link information for peers that are within its discovery zone. Therefore, queries can be solved solely within the discovery zone. The link-state information is used to build and calculate the paths to all of the known destinations. The following is a simplified way of looking at the various steps of the protocol:

1. Each node generates link-state advertisements (LSA) for the entire link set. The LSA includes a:
   a. source identification sequence number S,
   b. link-state age T,
   c. list of neighbors L_N,
   d. cost of link C, and
   e. level number l representing the number of hops the LSA has passed (l starts from 0).

   A node generates a LSA periodically, as well as in response to the discovery of a new neighbor, a semantic link, or when a neighbor changes state from up to down or conversely. The LSA advertisement will represent the collection of all link states on that node.
2. All of the nodes will exchange link states by means of limited broadcasting. Each node that receives a link-state update must store a copy on the local link-state

TABLE 1
Node A's LSA

From	To	Cost	LSA seq. num	level
A	B	3	23	0
A	C	4	23	0
A	D	4	23	0

database and add one level to the level number l, i.e., $l = l + 1$. If l is less than the zone radius r, then the node will propagate the update to the other nodes. Each of the participating nodes constructs a graph, termed as the *link-state graph* that is based on the link-state database. The link-state graph is used to find paths between nodes.

3. If not a sufficient number of changes occur in the zone, such as many links being added or deleted, then the traffic caused by transmitting LSA must be considered to be trivial. Any changes that occur are communicated via LSA, and the link-state graph is reconstructed to reflect the update.

Algorithm 1 details the process of the LSA propagation.

Algorithm1. foward_LSA (LSA).

/ A node call this method when a LSA packet is received*/*

1. *if the LSA packet has been received before*
2. *discard the LSA packet*
3. *Return*
4. *Update local link-state database*
5. *Increase LSA.l by 1*
6. *If LSA.l less than r*
7. *Insert current_node_id into LSA. L_N*
8. *For each neighbor N except the node from which the LSA is received*
9. *Send LSA to N*

Let us assume that we have a network diagram with the indicated semantic costs as reported in Fig. 4. Every node will broadcast the corresponding LSA within the zone. For example, A sends its LSA as shown in Table 1 to its neighbors. A's neighbors, for example B, will forward the LSA to its neighbors after adding 1 to the level number l, as shown in Table 2. Eventually, this LSA will arrive at every node within the zone of A. Based on its LSA database, each node can construct a zone graph. The LSA database of node A is presented in Table 3. (For simplicity, we assume that the cost is the same in both directions.) All of the nodes within A's zone are included in A's LSA database. All of the LSAs are flooded (with time to live—TTL r) through the network. Flooding means that every node sends the received LSA to all of its neighbors except to the one from where it has been received. To avoid flooding of older LSAs, each LSA is designed to hold a sequencing number. By observing the sequence number, a node can determine if the newly arrived LSA carries more recent information than the one that is already in the database. Modulo N numbering is used to avoid large sequencing. According to its link-state database, a node can construct the link-state zone graph. For instance, the graph shown in Fig. 4 is A's link-state graph.

TABLE 2
Node A's LSA Forwarded by A's Neighbor

From	To	Cost	LSA seq. num	level
A	B	3	23	1
A	C	4	23	1
A	D	4	23	1

TABLE 3
Node A's Link-State Database Table

From	To	Cost	LSA seq. num
A	B	3	23
A	C	4	23
A	D	4	23
B	A	3	10
B	E	5	10
B	H	4	10
C	A	4	14
C	F	6	14
C	G	3	14
D	A	4	15
D	G	2	15
E	B	5	5
E	F	7	5
E	I	4	5
F	C	6	9
F	E	7	9
G	C	3	11
G	D	2	11
H	B	4	3

4.2.4 Border Casting

Based on its semantic zone graph, nodes can solve the path discovery query very efficiently. As previously mentioned in Section 4.2.2, the value of the zone radius r is normally set to be large enough to cover a query's semantic path length limit k. Therefore, in most of the cases, the path discovery query can be resolved within the querying node's local zone. A node receiving a request to discover the paths to a destination first checks whether the destination is within its local zone using its local semantic zone graph. If the destination is within its semantic zone, then the node can retrieve the paths based on local zone graph. Otherwise (rarely), the route discovery query will be sent to the border nodes that are r hops away. If the receiver of the route request knows the destination, then it responds by sending a route reply. Otherwise, it continues the process by forwarding the request to its borders. In this way, the discovery request spreads throughout the network.

Forwarding through the borders, termed border casting, can be more efficient than flooding. This is due to the fact that the route requests are only sent to the peripheral nodes. Because the zones of borders may overlap with each other as well as with that of the original query node, border casting may create redundant query messages. To solve the above-stated problem, we can use some query-control mechanisms, such as caching or computing the overlaps

to direct queries away from covered zones [44]. As mentioned previously, due to the semantic locality property [43], most of the queries can be solved within the local zones and do not need to be forwarded to the borders.

4.3 Semantic Path Retrieval

Given a query that discovers the relationships between entities A and B, the system first locates the two peers, say P_A, P_B, in charge of these two entities. There may be multiple such peers, as most probably they must have been linked (as gateways) at the topology adaptation stage as mentioned in Section 3.2.

To efficiently locate all of the paths between the source and destination nodes whose length is less than k, we use the depth-limited search algorithm [30], which is depth-first search with a predetermined depth limit l. That is to say, nodes at depth l are treated as if they have no successors. The value of l is set to $k - t$, where t is the sum of distances between the starting and ending entities to their respective gateways. Path discovery is first performed at the peer level (knowledge base-level). Using the depth-limit search, the system can efficiently find all of the paths connecting P_A and P_B at the peer or knowledge base level. The cost in a node's link-state database table (i.e., the minimum distance between knowledge bases) is used as an upper bound to prune paths beyond the hop limit at the peer level. That is, if we find that the minimum cost of a particular path passing through knowledge bases exceeds the predefined hop limit k, then we do not need to retrieve this path at the semantic entity level. Therefore, we can avoid unnecessary computing and communication cost.

After the peer-level paths have been located, we can further retrieve paths at the semantic entity level. First, we need to find the semantic path from entity A to a gateway node in P_A, which lies on the path to P_B. Similarly, we find the semantic path from entity B to a gateway node in P_B, which lies on the path to P_A. Thereafter, we will retrieve the semantic entity paths following the peer-level paths. During the entity-path retrieval stage, we also look into the "black boxes," i.e., the knowledge bases to enumerate all of the possible path combinations connecting different gateways. Using Fig. 3 as an example, from knowledge bases B to C, which passes through the knowledge base A, we may retrieve two possible paths: Path1: $A_1A_3A_4A_5$, Path2: $A_2A_4A_5$, with length 3 and 2, respectively. While from knowledge bases B to D, which passes through knowledge base A, we may retrieve four possible paths: Path1: $A_1A_3A_6A_7$, Path2: $A_1A_3A_6A_8$, Path3: $A_2A_4A_3A_6A_7$, and Path4: $A_2A_4A_3A_6A_8$.

4.4 Analysis of the Proposed Algorithms

In this section, we analyze some of the important properties of the proposed algorithms. The time and communication complexities of the topology construction algorithm (as shown in Algorithm 1) and the path discovery algorithm (as presented in Section 4.3) are listed in Table 4. To construct the link-state database table that is needed for routing queries, each node within the network periodically (with update interval I) broadcasts the LSA within its zone. For each node, the table update complexity is deemed to be constant. For path discovery, if the path limit k is within the range of a node's zone radius r, then the path can be retrieved instantly by peer-level depth-limit search of the zone's topology and the entity-level retrieval of the path combinations between different gateways. On the other hand, if path limit k is beyond the range of a node's zone radius r, then an on-demand border casting will be performed to get the peer-level path that is beyond the zone radius r but within k semantic hops.

TABLE 4
Complexity of the Proposed Algorithms

Algorithm		Time Complexity	Communication Complexity
Topology Construction		$O(I \times D)$	$O(Z \times V)$
Path Discovery	$k \leq$ zone limit	$O(N^L)$	$O(P)$
	$k >$ zone limit	$O(L/r+N^L)$	$O(B^{L/r}+P)$

Acronyms—D: Diameter of the network; I: Periodical update interval; Z: Maximum number of nodes in a zone; V: Number of nodes in the network; N: Maximum number of neighbors per node; L: Number of nodes on the longest reply path; P: Total number of nodes on all of the reply paths; B: Number of border nodes; r: radius of a zone; k: length limit of semantic paths.

We shall say that the proposed search algorithm is correct if an execution returns paths from the source semantic entity to the destination semantic entity. Using the depth-first-based search, the semantic path discovery algorithm returns entity-level paths connect the source entity with the destination entity. That is, the proposed algorithm can correctly discover the semantic relationships between two semantic entities. Moreover, the path discovery algorithm must also be considered complete in theory. Our claim is based on that: 1) peer-level search is based on depth-limited search strategy, which is a complete search algorithm [33] with proven theoretical correctness, and 2) exhaustive enumeration of all of the possible candidate paths connecting different gateways and the path-length limit of l at the semantic entity-level guarantees the completeness of the proposed algorithm at the entity level. However, large-scale distributed systems are dynamic in nature as nodes may join, leave, or fail, which entails that knowledge bases must be updated very frequently. Therefore, in practice, the paths returned may not (in some cases) be 100 percent complete. The simulated experimental results in Section 5.2 further justify the correctness and completeness of the proposed algorithm.

5 Experiments and Discussion of Results

5.1 Experimental Setup

We tested the performance of the proposed mechanism with both real and synthetic data sets. We used an open knowledge base, DBpedia [37], as our knowledge source. DBpedia extracts structured information from Wikipedia and uses the RDF as a data model for representing extracted information. The DBpedia knowledge base currently describes more than 3.4 million entities, out of which 1.5 million are classified in a consistent Ontology, including 312,000 persons, 413,000 places, 94,000 music albums, 49,000 films, 15,000 video games, 140,000 organizations, 146,000 species, and 4,600 diseases [37]. In particular, we used the RDF dumps of triples from the infobox [37].

To simulate a distributed environment in which dispersed DBpedia knowledge can be shared, we created a network simulator with 1,024 computers (nodes). Thereafter,

we divided the dumped infobox triples into smaller parts (subknowledge bases). In particular, we adopted the RelFinder's decomposition algorithm [55] to decompose the DBpedia infobox graph into maximally connected subgraphs. We only used the largest connected subgraph in our experimental analysis. Thereafter, we applied a well-known clustering algorithm to cluster the graph obtained from the decomposition process into "communities" based on the idea of edge betweenness [56]. A community is defined as a group, where the nodes are densely connected, but there are few connections between disparate groups. Thereafter, we deployed each of these communities on one of the nodes within the simulated network. To model the interknowledge base ontology mappings or links, we used the preexisting links connecting entities that were located in different subknowledge bases after knowledge base decomposition. Therefore, we converted the centralized DBpedia knowledge base to a set of smaller knowledge bases and distributed them in the network.

To better control the semantic data and imitate the natural semantic mapping between autonomous organizations in a large distributed network, we used synthetic data sets to test our proposed approach. Because the ontology data can be characterized by many factors, such as the number of classes, properties, and individuals, we generated the test data in multiple steps that are detailed in the subsequent text. The data generation started with the ontology schema. Each schema included the definition of a number of classes and properties. Because our work primarily focuses on discovering semantic relations, we created only object properties without any data type properties. Each object property linked two classes, one as the domain and the other as the range. The classes were instantiated by creating a number of individuals of the classes. To generate an RDF instance triple t, we first randomly chose an instance of a class C among the classes to be the subject *sub(t)*. A property p of C is chosen as the predicate *pre(t)*, and a value from the range of p to be the object *obj(t)*. The range of the selected property p is instances of a class C', i.e., *obj(t)* is a resource.

We used BRITE [22], to generate network topologies. Our simulator used a parser that can parse the output file exported by BRITE to create the targeted topology. Power-law distributions (also known as heavy-tailed distributions) have been observed in the Internet and also in the case of Semantic Web [46], [47]. Therefore, we incorporate power law in the topology generation by using Waxman and BarábasiAlbert models. Moreover, the knowledge base distribution also follows the power-law distribution. In particular, we use a Zipf distribution [57] to model the distribution of the knowledge base.

Once the network topology is established, all of the nodes in the network generated their semantic instance data using the above-mentioned procedure. For simplicity, we constructed instance mappings between neighboring nodes, i.e., we assume semantic mappings have been created. Queries were generated by providing two semantic entities, the source and the destination. The source was picked randomly from a peer's knowledge base. We also randomly picked a semantic path starting from the source entity with a path of length limit (1-10), which led to a semantic entity that is labeled as the destination. The path may also cross multiple peers (knowledge bases). Therefore, the query was to find all of the paths (of length limit k) between these two semantic entities. Each experiment was repeated 10 times with different random seeds, and the results (reported in this paper) are the average of the obtained results.

TABLE 5
Important Parameters Used in the Simulations

Parameter	range (default)
network size	512-4096 (4096)
network topology	Waxman random (α=0.15, β=0.2), and Barabasi -Albert
# of semantic entities per node	3-25
# of semantic links per entity	1-10
length limit between source and destination entity (used when artificially generating the query)	1-25
zone radius	1-6
semantic path length limit	1-25
knowledge update probability per node per time slice	0.001%
query probability per node per time slice	0.1%

The various simulation parameters and their default values are listed in Table 5. To simulate an open large-scale distributed environment, the network size was varied within the range of [512, 4,096]. As mentioned previously, we adopt Waxman and BarábasiAlbert models to simulate the power-law network topology. According to existing research on DBpedia, the average number of outgoing connections of an object is 5.67, and the distance between any two objects is normally within 5-9 steps [55]. Considering that the DBpedia is a very large complex knowledge base, our parameters set for the number of semantic links per entity and the path length limit would be large enough to model the distributed knowledge bases. The values of the zone radius were set based on our prior studies [20], [59] and experimental results reverified in this study. To simulate a dynamic environment, the knowledge bases continuously update. The query frequency and the knowledge base update frequencies were modeled as a probability in each time slice.

5.2 Experimental Results

5.2.1 Experiments Based on Real Knowledge Base

We began to evaluate the correctness of the proposed discovery scheme using the DBpedia knowledge base. We first discovered the relationships between two entities in the parsed centralized DBpedia knowledge base and record the results. Thereafter, we divided the knowledge base into subknowledge bases and deployed them on the simulated nodes within the network. We applied our proposed approach to perform resource discovery and compared the results with a centralized knowledge base.

Fig. 5 depicts the example results of relationships between two entities "Bill Clinton" and "Barack Obama" returned by our proposed distributed approaches. It is exactly the same as the expected result returned through querying the centralized knowledge base. The length limit k was set to 2 in this experiment. (Note that the entities in the

Bill Clinton (occupation) Lawyer (occupation) Barack Obama
Bill Clinton (title) President of the United States (incumbent) Barack Obama
Bill Clinton (party) Democratic Party (party) Barack Obama
Bill Clinton (spouse) Hillary Rodham Clinton (president) Barack Obama
Bill Clinton (successor) George W. Bush (successor) Barack Obama

Fig. 5. Discovered relationships between two entities in DBpedia using the proposed discovery approach.

Length limit	1	2	3	4
Recall	100%	100%	100%	100%

Fig. 6. Recall of the proposed discovery approach versus semantic relation length limit.

Fig. 7. Average bandwidth consumption.

brackets are properties ("dbppro" in DBpedia dump)). We did not consider the direction of the links in this set of experiments. Fig. 6 illustrates the *recall* rate of the proposed distributed discovery scheme, which refers to the fraction of the semantic relations that are relevant to the query that are successfully retrieved. Relations obtained from the centralized knowledge base were used as the standard answers. The results illustrate that the proposed approach can find all of the related relationships. It is difficult to determine which relationships are of relevance in a certain situation and which irrelevant [49]. Therefore, our work focuses on locating as many relationships as possible without considering the precision.

To evaluate the scalability and efficiency of the proposed discovery protocol SZDP, we compared its performance with breadth-first search (BFS)-based discovery scheme [1], [42] in terms of average bandwidth and the number of semantic entities traversed. BFS-based discovery [1], [42] was used as the benchmark scheme because of its simplicity and popularity [45]. In the simulations, a random set of nodes periodically issue discovery queries. The queries were to find the relationships between a local entity of the querying node and another entity randomly selected from a node in the network. In each simulation time slice (1 second), the query probability for a given node was set to 0.1 percent. The length limit of the relations k was set to 4. Zone radius was also set to 4. As can be seen from Figs. 7 and 8, our proposed discovery protocol SZDP significantly outperforms BFS-based discovery in terms of the bandwidth consumed and the number of semantic entities traversed. (Note that Fig. 8 was plotted on a logarithmic scale to better illustrate the significance of the proposed methodology.) This justifies the efficiency of the proposed discovery scheme.

5.2.2 Experiments Based on Synthetic Data

In the previous experiments, we used the DBpedia data dump as our knowledge base. Figs. 5, 6, 7, and 8 have justified the correctness and efficiency of the proposed

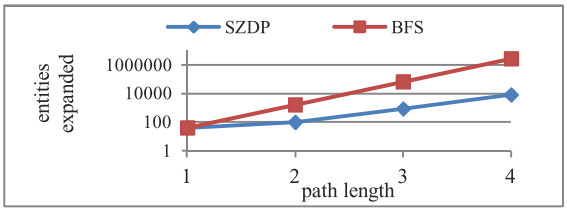


Fig. 8. Number of entities expanded versus relation path length.

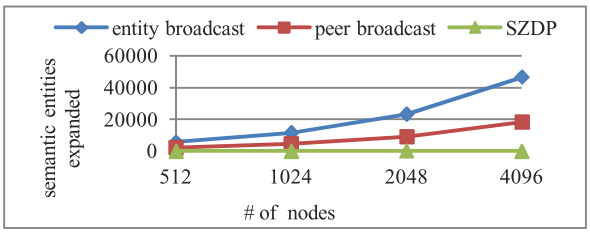


Fig. 9. Performance of discovery at different levels of abstraction.

approach. However, DBpedia is a centralized stand-alone database. Therefore, to evaluate our discovery approach, we must divide DPpedia into multiple subknowledge bases and use the original properties or links as mappings between knowledge bases. This does not reflect the real working scenario of our approach, within which knowledge bases are created autonomously and independently, and mappings are constructed between related nodes. To solve the aforementioned problem, we turn to synthetic data sets to test our proposed methodologies. The experiments in the rest of this section focus on evaluating important properties of the proposed approach and are performed on the synthetic data that we had created.

First, we studied the properties of the hierarchical discovery scheme. Our proposed discovery algorithm uses a hierarchical and multiresolution strategy to search for semantic paths. We believe this strategy improves the scalability of the system. To test this hypothesis, we first evaluated the effectiveness of the hierarchy (or say abstraction) strategy. Fig. 9 compares the number of semantic entities traversed when searching at different levels of abstractions. That is to say, from the low-level semantic entity search to the knowledge base-level peer search, and finally to the more abstract zone-based search. We assume that the source and destination entities have been located for all of these three cases. As can be observed from Fig. 9, by using levels of abstraction and applying routing technologies, our discovery protocol SZDP significantly reduces the search space and improves scalability. It is noteworthy to mention that the number of entities traversed by SZDP is always less than 100 for the four networks, which is considerably less than both of the entity broadcast and peer broadcast approaches. We must note that the results of the SZDP are obtained after the routing structure has been constructed. That is, we did not consider the preprocessing overhead. We can ignore the preprocessing overhead because: 1) The preprocessing overhead is not very high (as reported in Fig. 11), and 2) once the initial routing structure is constructed, the system will not produce that overhead anymore.

Second, we evaluated the performance of our proposed end-entity location mechanism. Locating the two end

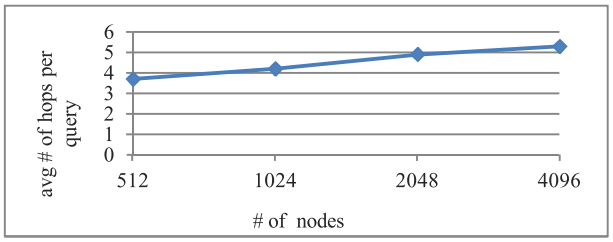


Fig. 10. Performance of end entity lookup.

entities of a semantic path is another important issue affecting the efficiency of the system. In this experiment, we measured the performance of locating source and destination entities by using a DHT indexing overlay. We adopted Pastry [28] to realize the DHT index overlay. Each peer was assigned a 160-bit identifier, representing 80 digits (each digit uses 2 bits) with base $b = 2$. We varied the number of Pastry nodes in the network between 512 and 4,096. Fig. 10 reports the average number of routing hops taken as a function of the network size. The results show that the number of route hops scales with the size of the network as hypothesized. With DHT indexing, the network can quickly locate the end entities and more importantly, nodes sharing the same instances can meet each other to merge or construct mappings or linkages.

Next, we measure the impact of the zone radius to the performance of the system. Intuitively, the zone radius r determines how far resource information can be propagated and how much a node can learn about the network. We can manipulate the tradeoffs between the routing maintenance overhead and query overhead by adjusting the *radius*.

First, Fig. 11 shows the total message overhead of the whole network to construct the link-state table. The network size in this experiment is 4,096. When radius $r = 1$, nodes do not need to exchange routing information and do not maintain the link-state table. However, nodes cannot know their neighborhood information. With the increase in r, the coverage grows exponentially and nodes must manage larger semantic zone graphs. Fortunately, this link-state initiation process is only performed only at the beginning of the network construction. Once each node has constructed its link-state database, the update can be done periodically and incrementally. The overhead is negligible if the network is relatively stable.

Second, Fig. 12 illustrates the relationship between the zone radius r and the average number of query messages propagated for a query with path length limit $k = 25$. The network size in this experiment is 4,096. We used caching and overlapping computation techniques [44] to control

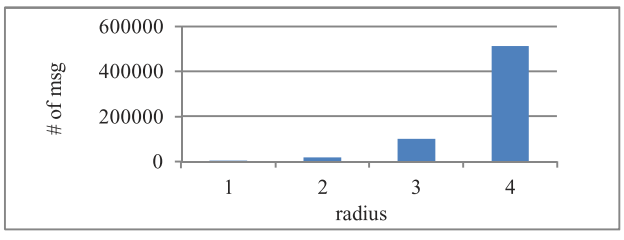


Fig. 11. Message overhead of constructing the link-state database table for various zone radius.

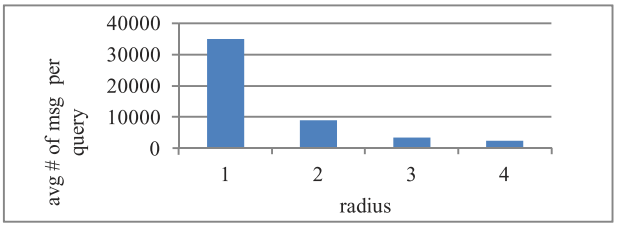


Fig. 12. Query overhead to discover in zones of various radius.

the duplicated messages of border casting. When the radius is set as 1, the SZDP algorithm degrades to a flooding algorithm. With the increase in the radius, the query overhead decreases dramatically. When the radius increases to 3, most of the queries can be answered in the querying nodes' local zone. Therefore, further increase in r is not necessary. In this experiment, we set path length limit $k = 25$ and increase the radius to 4. This is only for demonstration purposes. As we have explained in Section 4.2.2, in reality, a reasonable number for k is normally less than 10. In that case, a small radius r would cover all the queried paths. For example, in Fig. 4, r is set to 2 and all of the path queries with length limit $k \leq 6$ can be answer within the zone. In this case, further increasing r is a waste because it only causes more system overhead without bringing any benefit.

The simulation results demonstrate the unique properties and superior advantages of the proposed discovery scheme, SZDP. The experiments performed on real DBpedia data illustrate the practicality, correctness, and efficiency of SZDP (Figs. 5, 6, 7, and 8). The simulations performed on synthetic data justify the scalability and efficiency of SZDP in a large-scale distributed environment. In particular, we analyze the performance of the major components of SZDP in terms of scalability and efficiency of:

1. knowledge abstraction (Fig. 9),
2. DHT-based entity lookup (Fig. 10),
3. query (Fig. 11), and
4. maintenance (Fig. 12).

We must also report that we were not able to compare SZDP with other related systems. To the best of our knowledge, there is only one previously reported work [15] on discovering semantic relationships in distributed environments. However, the approach presented in [15] has not been implemented or experimentally evaluated or benchmarked using simulations. Moreover, many technical details (as mentioned in Section 1) are not addressed in the work reported in [15]. Therefore, we could not implement the approach by following the description outlines in [15]. However, we have made every possible effort to make our simulation study as thorough as possible so that all aspects of the proposed system can be highlighted.

6 Related Work

Most of the current research on searching or querying Semantic Web uses an information retrieval (IR)-based search engine [10], [11], [14], [32]. The IR-based systems, such as Swoogle [10] and SWSE [14], index the Semantic Web by crawling and indexing the Semantic Web RDF documents that are found online and then offer a search interface over

these documents. However, the IR-based searching techniques do not provide structured query capability.

Several groups [5], [9], [13], [17] have developed technologies to store RDF nodes, edges, and labels into relational database systems, such as MySQL, Oracle, and DB2 so that Semantic Web data can be efficiently indexed and retrieved. The above-mentioned works translate a SPARQL query into SQL statements that are evaluated on the triplet stored in the relational databases.

To address scalability issues, researchers have adapted P2P technologies to Semantic Web. Systems, such as Edutella [23] and InfoQuilt [2], use broadcast or flooding to search the RDF data, while many other projects, such as RDFPeer [8] and OntoGrid [24], apply DHT-based techniques [28], [31] for the retrieval of ontology-encoded knowledge. The queries that we address (in this paper) are fundamentally different from those described in any of the above-mentioned approaches. Queries in the aforementioned approaches are mostly concerned with locating specific resources that satisfy specific constraints, while we focus on locating relations between resources.

There have been many works on semantic relation discovery using data mining approaches, such as association rules and clustering [38], [39], [40], [41]. However, their data sets are quite different from our semantic link-based data, and their discovery mechanisms are normally based on the co-occurrence of the entities in documents, which is significantly different from our work.

The query supporting RDF semantic relationships were first proposed by Anyanwu and Sheth [1]. In particular, they define a semantic association as a complex relationship between two resources and introduce a set of operators for querying semantic associations. Based on their work, several applications have appeared that use semantic relations [33], [34], [42], [49], [55]. However, most of these applications assume a centralized data set. Researchers in [15] propose a method for computing semantic associations over a P2P network. The authors use a superpeer-based query planning algorithm for ρ-path quires. In their proposed system, knowledge bases are stored at the peer level, while indexes are stored at the superpeer level. Each superpeer is responsible for a group of peers. A superpeer knows about all of the other superpeers in the network and can query them to determine the semantic paths. This is an effective approach, but the scalability is still an unsolved issue. That is, 1) how to organize the peer group to reflect the semantic closeness, and 2) how superpeers efficiently communicate are unaddressed.

Path finding using hierarchy is described in [4], which abstracts a map into linked local clusters. At the local level, the optimal distances for crossing each cluster are precomputed and cached. At the global level, clusters are traversed in a single "big" step. Such a hierarchy can be extended to more than two levels, and small clusters are grouped together to form larger clusters. Computing the crossing distances for a large cluster uses distances computed for the smaller contained clusters. Related to the aforementioned work, there also are other works [26], [27] that use hierarchical approach for path finding. Our hierarchical routing was inspired by the path-finding problems. However, unlike the path-finding problems reported in [4], [26], [27], we do not have a complete view of the "map" rather we only have pieces of "maps" distributed all over the network. Therefore, we merge the path-finding problem with the network routing problem to identify novel solutions for web semantics.

7 Conclusions

In this paper, we presented a scalable and efficient approach to discover complex semantic relationships from distributed knowledge bases. This approach allows users to share their local knowledge to collectively make new discoveries. By correlating isolated islands of knowledge base, we constructed a large-scale semantic graph. The relation discovery problem was then converted into a path discovery problem over the semantic graph. Inspired by the route planning problem, we adopted abstraction to reduce the huge discovery space. The proposed technique defined the semantic graph on three levels: entity level, knowledge base level, and zone level. At the knowledge base level, the optimal distances for crossing each knowledge base were precomputed. A zone-based discovery protocol was proposed to efficiently search at the knowledge base level. This method also supported path discovery in dynamic environments. The experimental results revealed the scalability and efficiency of the discovery framework.

As future work, we plan to incorporate richer semantics as well as security or privacy enforcement into the system. In particular, we will add weights to the semantic links based on trust, property of the relation relevance, and so on. We also plan to study how to efficiently rank the discovery results in a large-scale Semantic Web. Moreover, we also aim to evaluate the proposed approach on real knowledge bases over the Internet.

References

[1] K. Anyanwu and A. Sheth, "ρ-Queries: Enabling Querying for Semantic Associations on the Semantic Web," *Proc. 12th Int'l Conf. World Wide Web (WWW '03),* 2003.

[2] M. Arumugam, A. Sheth, and I.B. Arpinar, "Towards Peer-to-Peer Semantic Web: A Distributed Environment for Sharing Semantic Knowledge on the Web," *Proc. Int'l Conf. World Wide Web (WWW '02),* 2002.

[3] A. Barabási, *Linked: How Everything Is Connected to Everything Else and What It Means for Business, Science, and Everyday Life.* Plume, 2003.

[4] A. Botea, M. Müller, and J. Schaeffer, "Near Optimal Hierarchical Path-Finding," *J. Game Development,* vol. 1, pp. 7-28, 2004.

[5] J. Broekstra, A. Kampman, and F. van Harmelen, "Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema," *Proc. First Int'l Semantic Web Conf.,* pp. 54-68, 2002.

[6] F. Banaei-Kashani and C. Shahabi, "Criticality-Based Analysis and Design of Unstructured Peer-to-Peer Networks as Complex Systems," *Proc. Third IEEE/ACM Int'l Symp. Cluster Computing and the Grid,* pp. 351-358, 2003.

[7] Y. Chawathe, S. Ratnasam, L. Breslau, N. Lanhan, and S. Shenker, "Making Gnutella-like P2P Systems Scalable," *Proc. ACM SIGCOMM,* 2003.

[8] M. Cai and M. Frank, "RDFPeers: A Scalable Distributed RDF Repository Based on a Structured Peer-to-Peer Network," *Proc. 13th Int'l Conf. World Wide Web (WWW '04),* May 2004.

[9] E.I. Chong, S. Das, G. Eadon, and J. Srinivasan, "An Efficient SQl-Based RDF Querying Scheme," *Proc. 31st Int'l Conf. Very Large Data Bases,* 2005.

[10] L. Ding, T. Finin, A. Joshi, R. Pan, R.S. Cost, Y. Peng, P. Reddivari, V. Doshi, and J. Sachs, "Swoogle: A Search and Metadata Engine for the Semantic Web," *Proc. 13th ACM Int'l Conf. Information and Knowledge Management,* 2004.

[11] R. Guha, R. McCool, and E. Miller, "Semantic Search," *Proc. 12th Int'l Conf. World Wide Web (WWW '03),* 2003.

[12] Z.J. Haas, M.R. Pearlman, and P. Samar, "Zone Routing Protocol (ZRP)," IETF MANET Working Group, 1999.

[13] S. Harris, "SPARQL Query Processing with Conventional Relational Database Systems," *Proc. Int'l Workshop Scalable Semantic Web Knowledge Base System,* 2005.

[14] A. Hogan, A. Harth, J. Umbrich, and S. Decker, "Towards a Scalable Search and Query Engine for the Web," *Proc. 16th Int'l Conf. World Wide Web (WWW '07),* 2007.

[15] M. Perry, M. Janik, C. Ramakrishnan, B. Arpinar, and A. Sheth, "Peer-to-Peer Discovery of Semantic Associations," *Proc. Workshop P2P Knowledge Management,* pp. 1-12, 2005.

[16] T. Pedersen, S. Patwardhan, and J. Michelizzi, "WordNet: Similarity-Measuring the Relatedness of Concepts," *Proc. 19th Nat'l Conf. Artificial Intelligence (AAAI '04),* 2004.

[17] M. Li, W. Chen, L. Jing, C. Feng, P. Yue, and Y. Yong, "Effective and Efficient Semantic Web Data Management over DB2," *Proc. ACM Int'l Conf. Management of Data (Sigmod '08),* June 2008.

[18] J. Li and S. Vuong, "SOON: A Scalable Self-Organized Overlay Network for Distributed Information Retrieval," *Proc. 19th IFIP/IEEE Int'l Workshop Distributed Systems: Operations and Management: Managing Large-Scale Service Deployment (DSOM '08),* pp. 1-13, 2008.

[19] J. Li and S.U. Khan, "MobiSN: Semantics-Based Mobile Ad Hoc Social Network Framework," *Proc. 28th IEEE GlobeCom,* 2009.

[20] J. Li, "Grid Resource Discovery Based on Semantically Linked Virtual Organizations," *J. Future Generation Computer Systems,* vol. 26, no. 3, pp. 361-373, 2010.

[21] C. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and Replication in Unstructured Peer-to-Peer Networks," *Proc. ACM SIGMETRICS Int'l Conf. Measurement and Modeling of Computer Systems,* 2002.

[22] A. Medina, A. Lakhina, I. Matta, and J. Byers, "BRITE: An Approach to Universal Topology Generation," *Proc. Ninth Int'l Symp. Modeling, Analysis and Simulation of Computer and Telecomm. Systems (MASCOTS '01),* p. 346, 2001.

[23] W. Nejdl et al., "EDUTELLA: A P2P Networking Infrastructure Based on RDF," *Proc. Int'l Conf. World Wide Web (WWW '02),* 2002.

[24] OntoGrid Project, http://www.ontogrid.net/, 2013.

[25] R. Rada, H. Mili, E. Bicknell, and M. Blettner, "Development and Application of a Metric on Semantic Nets," *IEEE Trans. Systems, Man and Cybernetics,* vol. 19, no. 1, pp. 17-30, Jan./Feb. 1989.

[26] S. Rabin, "A* Aesthetic Optimizations," *Game Programming Gems,* M. Deloura, ed., pp. 264-271, Charles River Media, 2000.

[27] S. Rabin, "A* Speed Optimizations," *Game Programming Gems,* M. Deloura, ed., pp. 272-287, Charles River Media, 2000.

[28] A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems," *Proc. IFIP/ACM Int'l Conf. Distributed Systems Platforms,* 2001.

[29] C. Rocha, D. Schwabe, and M.P. Aragao, "A Hybrid Approach for Searching in the Semantic Web," *Proc. 13th Int'l Conf. World Wide Web (WWW '04),* 2004.

[30] J. Russell Stuart and P. Norvig, *Artificial Intelligence: A Modern Approach,* second ed. Prentice-Hall, 2003.

[31] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," *Proc. ACM SIGCOMM,* pp. 149-160, Aug. 2001.

[32] L. Zhang, Q. Liu, J. Zhang, H. Wang, Y. Pan, and Y. Yu, "Semplore: An IR Approach to Scalable Hybrid Query of Semantic Web Data," *Proc. Sixth Int'l Semantic Web Conf.,* 2007.

[33] B.A. Meza, M. Nagarajan, C. Ramakrishnan, L. Ding, P. Kolari, A.P. Sheth, I.B. Arpinar, A. Joshi, and T. Finin, "Semantic Analytics on Social Networks: Experiences in Addressing the Problem of Conflict of Interest Detection," *Proc. 15th Int'l Conf. World Wide Web (WWW '06),* May 2006.

[34] A. Sheth, B. Aleman-Meza, I.B. Arpinar, C. Halaschek, C. Ramakrishnan, C. Bertram, Y. Warke, D. Avant, F.S. Arpinar, K. Anyanwu, and K. Kochut, "Semantic Association Identification and Knowledge Discovery for National Security Applications," *J. Database Management,* vol. 16, no. 1, pp. 33-53, Jan.-Mar. 2005.

[35] J. Lehmann, J. Schuppel, and S. Auer, "Discovering Unknown Connections: The DBpedia Relationship Finder," *Proc. First SABRE Conf. Social Semantic Web (CSSW '07),* 2007.

[36] S. Auer and J. Lehmann, "What Have Innsbruck and Leipzig in Common?" *Proc. Fourth European Semantic Web Conf. (ESWC '07),* pp. 503-517, 2007.

[37] DBpedia, http://wiki.dbpedia.org/OnlineAccess, 2013.

[38] S. Nagano, M. Inaba, and T. Kawamura, "Extracting Semantic Relations for Mining of Social Data," *Proc. Third Int'l Workshop Social Data on the Web,* 2010.

[39] M. Ruiz-Casado, E. Alfonseca, and P. Castells, "Automatic Extraction of Semantic Relationships for WordNet by Means of Pattern Learning from Wikipedia," *Proc. 10th Int'l Conf. Applications of Natural Language to Information Systems, (NLDB '05),* pp. 67-79, June 2005.

[40] Z. Syed, E. Viegas, and S. Parastatidis, "Automatic Discovery of Semantic Relations Using MindNet," *Proc. Seventh Int'l Conf. Language Resources and Evaluation,* 2010.

[41] T. Jiang, A.H. Tan, and K. Wang, "Mining Generalized Associations of Semantic Relations from Textual Web Content," *IEEE Trans. Knowledge and Data Eng.,* vol. 19, no. 2, pp. 164-179, Feb. 2007.

[42] P. Heim, S. Lohmann, and T. Stegemann, "Interactive Relationship Discovery via the Semantic Web," *Proc. Seventh Int'l Conf. Semantic Web: Research and Applications (ESWC '10),* pp. 303-317, 2010.

[43] R. Akavipat, L.-S. Wu, F. Menczer, and A. Maguitman, "Emerging Semantic Communities in Peer Web Search," *Proc. Int'l Workshop Information Retrieval in Peer-to-Peer Networks (P2PIR '06),* pp. 1-8, 2006.

[44] Z. Haas and M. Pearlman, "The Performance of Query Control Schemes for the Zone Routing Protocol," *Proc. ACM SIGCOMM,* 1998.

[45] V. Krishna, N.N.R. Ranga Suri, and G. Athithan, "A Comparative Survey of Algorithms for Frequent Subgraph Discovery," *Current Science,* vol. 100, no. 2, pp. 190-198, 2011.

[46] L. Ding and T. Finin, "Characterizing the Semantic Web on the Web," *Proc. Fifth Int'l Semantic Web Conf. (ISWC '06),* 2006.

[47] Y. Theoharis et al., "On Graph Features of Semantic Web Schemas," *IEEE Trans. Knowledge Data Eng.,* vol. 20, no. 5, pp. 692-702, May 2008.

[48] N.A. Lynch, *Distributed Algorithms.* Morgan Kaufmann, 1996.

[49] P. Heim, S. Lohmann, and T. Stegemann, "Interactive Relationship Discovery via the Semantic Web," *Proc. Seventh Extended Semantic Web Conf. (ESWC '10),* May/June 2010.

[50] Z. Bellahsene, S. Benbernou, H. Jaudoin, F. Pinet, O. Pivert, F. Toumani, S. Bernard, P. Colomb, R. Coletta, E. Coquery, F. De Marchi, F. Duchateau, M. Hacid, A. HadjAli, and M. Roche, "FORUM: A Flexible Data Integration System Based on Data Semantics," *ACM Sigmod Record,* vol. 39, no. 2, pp. 11-18, 2010.

[51] P. Adjiman, P. Chatalic, F. Goasdoué, M. Rousset, and L. Simon, "Distributed Reasoning in a Peer-to-Peer Setting: Application to the Semantic Web," *J. Artificial Intelligence Research,* vol. 25, pp. 269-314, 2006.

[52] G. Qi, Q. Ji, and P. Haase, "A Conflict-Based Operator for Mapping Revision," *Proc. Eighth Int'l Semantic Web Conf.,* pp. 521-536, 2009.

[53] P. Haase, R. Siebes, and F. Harmelen, "Peer Selection in Peer-to-Peer Networks with Semantic Topologies," *Proc. Int'l Conf. Semantics in a Networked World (ICNSW '04),* pp. 108-125, 2004.

[54] P. Adjiman, F. Goasdoué, and M. Rousset, "SomeRDFS in the Semantic Web," *J. Data Semantics,* vol. 8, pp. 158-181, 2007.

[55] J. Lehmann, J. Schuppel, and S. Auer, "Discovering Unknown Connections—The DBpedia Relationship Finder," *Proc. First Conf. Social Semantic Web (CSSW '07),* 2007.

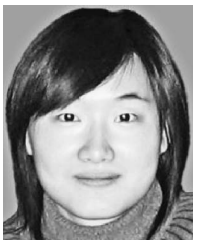
[56] J. Peng, D.R. Heisterkamp, and H.K. Dai, "LDA/SVM Driven Nearest Neighbor Classification," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR '01),* p. 58, 2011.

[57] G.K. Zipf, *Human Behaviour and the Principle of Least-Effort.* Addison-Wesley, 1949.

[58] W3C, "SPARQL Query Language for RDF," http://www.w3.org/TR/ rdf-sparql-query/, 2013.

[59] J. Li, "Exploiting Social Property for Improving Distributed Semantic Search," *Int'l J. Computer and Information Science,* vol. 10, no. 2, 2009.

Juan Li received the BS degree from Beijing Jiaotong University, China, in July 1997 and the PhD degree from the University of British Columbia, Vancouver, Canada, in May 2008. She is currently an assistant professor in the Computer Science Department, North Dakota State University, Fargo. Her major research interest lies in distributed systems, including P2P networks, grid and cloud computing, mobile ad hoc networks, social networking, and Semantic Web technologies. She is a member of the IEEE.



Hui Wang received the BS degree in computer science from Wuhan University in 1998 and the MS and PhD degrees in computer science from the University of British Columbia in 2002 and 2007. She has been an assistant professor in the Computer Science Department, Stevens Institute of Technology, since 2008. Her research interests include data management, database security, and data privacy. She is a member of the IEEE.

Samee Ullah Khan received the BS degree from the Ghulam Ishaq Khan Institute of Engineering Sciences and Technology, Topi, Pakistan, and the PhD from the University of Texas, Arlington. Currently, he is assistant professor of electrical and computer engineering at the North Dakota State University, Fargo. His research interests include optimization, robustness, and security of cloud, grid, cluster and big data computing, social networks, wired and wireless networks, power systems, smart grids, and optical networks. His work has appeared in over 200 publications. He is a fellow of both the Institution of Engineering and Technology and the British Computer Society.

Qingrui Li received the BS degree from Zhejiang University, Hangzhou, China, in 2002, and the MS degree from the National University of Singapore in 2004. He is currently working toward the PhD degree in the Computer Science Department, North Dakota State University, Fargo. His current research focuses on intelligent systems. He is a student member of the IEEE.

Albert Y. Zomaya is the chair professor of High Performance Computing and Networking and Australian Research Council Professorial Fellow at the School of Information Technologies, University of Sydney. He is also the director at the Centre for Distributed and High Performance Computing, which was established in late 2009. He is the author/coauthor of seven book and more than 400 papers, and he is the editor of nine books and 11 conference proceedings. He is the editor in chief of the *IEEE Transactions on Computers* and serves as an associate editor for 19 leading journals. In 2000, he received the Meritorious Service Award, and in 2006, he received the Golden Core Recognition, both from the IEEE Computer Society. He also received the IEEE TCPP Outstanding Service Award and the IEEE TCSC Medal for Excellence in Scalable Computing, both in 2011. He is a chartered engineer, a fellow of the AAAS, IEEE and IET (United Kingdom), and a distinguished engineer of the ACM.