2012 IEEE 28th International Conference on Data Engineering

R^{2}DB: A System for Querying and Visualizing Weighted RDF Graphs

Songling Liu #1, Juan P. Cedeño #1, K. Selçuk Candan #1, Maria Luisa Sapino #2,
Shengyu Huang #1, and Xinsheng Li #1

[1]*Computer Science and Engineering*
Arizona State University,
Tempe, AZ, 85287, USA.
{songling.liu,jcedeno,candan,shengyu.huang,lxinshen}@asu.edu

[2]*Computer Science,*
University of Torino,
Torino, Italy.
mlsapino@di.unito.it

***Abstract*—[1] Existing RDF query languages and RDF stores fail to support a large class of knowledge applications which associate *utilities* or *costs* on the available knowledge statements. A recent proposal includes (a) a *ranked RDF* (R2DF) specification to enhance RDF triples with an application specific weights and (b) a SPARankQL query language specification, which provides novel primitives on top of the SPARQL language to express top-k queries using traditional query patterns as well as novel flexible *path predicates*. Here, we introduce and demonstrate R^{2}DB, a database system for querying weighted RDF graphs. R^{2}DB relies on the AR2Q query processing engine, which leverages novel index structures to support efficient ranked path search and includes query optimization strategies based on proximity and sub-result inter-arrival times. In addition to being the first data management system for the R2DF data model, R^{2}DB also provides an innovative features-of-interest (FoI) based method for visualizing large sets of query results (i.e., subgraphs of the weighted RDF data graph).**

I. Introduction

Resource Description Framework (RDF) [15] represents data or metadata in the form of a directed graph, where nodes represent entities and the labeled edges represent relationships between pairs of entities. Applications access RDF knowledge bases, residing in RDF stores such as Jena TDB [23], through declarative query languages, such as SPARQL [19] and SeRQL [18]. In [6], we noted that existing RDF query languages and RDF stores fail to support key primitives needed in a large class of applications which associate *utilities* or *costs* on the available knowledge statements. The ability to annotate statements in the database is especially important for applications which need to impose additional parameters, like validity, trust, or preference to the data [4]. Figure 1, for example, shows a weighted graph representing the integration of two conflicting taxonomies: each edge value indicates the amount of *source disagreement* on the corresponding assertion [20].

When the knowledge statements have *utilities* or *costs* associated to them, the utility or cost of query results also varies and users are interested not in all the results to their queries, but in the ones that are best suited for the given task. These necessitate new language primitives and a query processing system extended in order to take advantage of the extra utility or cost information associated with the RDF statements. In [6], we proposed the *R2DF* model, an extension of the RDF that allows application specific weights attached to the triples. In addition, we defined a new *SPARankQL* language, an extension of SPARQL, which enables users to express top-k queries using traditional patterns and operators, novel path predicates, and top-k queries over weighted RDF graphs using both the traditional subject-predicate-object query patterns as well as advanced operators to support reachability predicates.

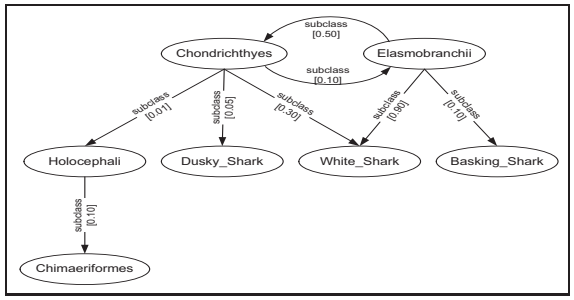


Figure 1. A sample knowledge base where the available information have varying utilities (adapted from [20], [21])

Here, we propose to demonstrate the R^{2}DB *database system for querying weighted RDF graphs*. Unlike many of existing graph DBMSs, such as [10], [11], [14], that focus on limited pattern types, R^{2}DB aims to execute arbitrary structured (SPARanQL) queries over (weighted) RDF graphs. For this purpose, R^{2}DB provides a query processor engine, AR2Q, which supports the novel features of the SPARankQL language, including path predicates and ranked joins. AR2Q leverages novel index structures and optimization strategies for efficient query execution over weighted RDF graphs [6].

Given a large R2DF graph and a complex SPARanQL query, which may return 100s or 1000s of subgraphs, visualizing the results to the user is not trivial. While it is possible to plot the original data graphically and to highlight the individual subresults on it, when the data graphs are large and queries are complex, the user cannot easily observe the key features of the results. Therefore, in addition to being the first data management system for the weighted RDF model, R^{2}DB *also includes an innovative features-of-interest (FoI) based method for visualizing large sets of query results, each of which is a subgraph of a large R2DF data graph*.

[1]This work is partially supported by NSF Grant NSF-III1016921. "One Size Does Not Fit All: Empowering the User with User-Driven Integration."

1084-4627/12 $26.00 © 2012 IEEE
DOI 10.1109/ICDE.2012.134

```
SELECT ?o ?w WHERE {
 :Chondrichthyes ?p ?o ?w .
 FILTER (?w > 0.01)
}
```
(a) Weight filtering

```
SELECT @ WHERE {
  ?s <nEdges> :Holocephali .
  ?s <nEdges> :Dusky_Shark .
  ?s ?p :White_Shark .
 }
```
(b) "SELECT @" query

```
SELECT ?s ?w WHERE {
 {?s <nEdges> :Holocephali .
  ?s <nEdges> :Dusky_Shark .
  ?s ?p :White_Shark .
 } RANK 10 DECREASING
}
```
(c) top-10 query

```
SELECT ?s ?w WHERE {
 {?s <nEdges> :Holocephali .
  ?s <nEdges> :Dusky_Shark .
  ?s ?p :White_Shark .
 } SCORE IN ?w .
 FILTER (?w > 0.01)
}
```
(d) threshold query

Figure 2. Example SPARankQL queries

II. R2DB FOR MANAGING WEIGHTED RDF DATA

In RDF [15], data are modeled as a graph, where each edge (or *triple*) expresses an assertion in the form of a 3-tuple and contains a "subject," a "predicate," and an "object."

SPARQL [19] enables users to express queries in the form of graph patterns that match and retrieve subgraphs from an RDF graph. A SPARQL query asks for subgraphs of the underlying RDF graph that *match* the corresponding *basic graph pattern (BGP)*: a subgraph *s* is a valid match for a BGP *b* if after substituting the variables of *b* with RDF nodes (or edges) of *s*, *b* and *s* are equal.

A. R2DF: Weighted RDF Model

In R2DF [6] model, a weighted triple t_{w} is a 4-tuple (or quadruple) composed of subject, predicate, object, and a real weight *w* between 0 and 1 representing the application specific utility/cost of the assertion. A weighted RDF graph G_{w} is a set of weighted triples.

B. SPARankQL Query Language

SPARankQL extends SPARQL with clauses for weight aware processing and general path matching (Figure 2):

1) Weight Extended Triple Patterns: This is an *optional* element, variable or literal, added to the triple patterns to reflect the weight associated to the matching results of the given pattern. As a variable, it can be used freely by other patterns or operators, such as filter or select (Figure 2(a)).

2) Path Query Predicates: The novel query predicate *<nEdges>* represents multi-edge relationships between two nodes in an RDF graph; the predicate matches one or more edges irrespective of the labels of the edges or vertices on the path (Figure 2(b)). The predicate can also be extended to impose optional constraints, such as label and upper and lower bounds to the weight of the matches.

3) "SELECT @" Clause: One difficulty with the standard "SELECT" and "SELECT *" clauses is that they allow the query to return only named variables (nodes or edges). Therefore, SPARankQL also introduces a new "SELECT @" clause, which outputs the entire matching result subgraph in the form of a (serialized) RDF graph (Figure 2(b)). Thus, the user can explore the resulting graph, including any edges or nodes that have not been explicitly enumerated in the query specification.

4) "RANK" Clause: The `RANK` clause takes as a parameter an integer, *k*, indicating the desired number of top results based on the results weights (Figure 2(c)).

5) "SCORE IN" Clause: As shown in Figure 2(d), the `SCORE IN` clause takes as a parameter a variable name which will be included in the set of query variables and bound to the weight of each resulting match for the BGP to which the clause belongs. The aggregation function, f, applied to the weights of the triples of the underlying subgraph may be `sum`, `min`, `max`, `average`, or `product`. The score variable can be used to associate a filter condition on the weight of the entire BGP.

C. AR2Q Engine for SPARankQL Query Processing

Extensions provided by the SPAR2QL language, including the extended triple patterns, path predicates, and the new clauses described in the previous section, necessitate (a) new algebraic operators, (b) new index structures, and (c) reconsideration of the query processing strategies [6].

1) Path Enumerator: AR2Q query engine provides a *path enumerator* component which returns solution mappings that match user specified path patterns. For this purpose, we use a version of Yen's k-shortest paths algorithm, which returns k shortest simple paths between a given pair of nodes [8].

2) Quadruple Index: AR2Q includes *weighted triple (quadruple) indices* that enable weighted triples (or quadruples) to be returned in non-increasing order of their weight no matter what literals are available in the query pattern.

3) Reachability Index: AR2Q creates an all-pairs reachability index to recover the reachability information quickly in run-time. The *reachability index* allows recovering, for each node, n, the list of nodes for which there is a path from n.

4) Proximity Index: Intuitively, a node is said to have a *high proximity* to another, if the paths connecting the first node to the second in the graph are large in number, short in length, and have strong weights [3], [24], [25]. We argue (and experimentally validate in [6]) that a query node which has high proximity to the other query nodes has a higher likelihood of producing many desirable matches with the remaining query nodes. The *proximity index* allows recovering, for each node n, the list of nodes that are close to it in terms of the number of paths between them and the length and costs of these paths.

5) Ranked Join Operators: In the literature, there are various ranked join algorithms, such as [17], [9], [7]. These commonly assume that the aggregation function being used to combine the ranked inputs is monotonic. Unfortunately, when combining subgraphs, whether the function is monotonic or not depends on whether *set* or *multi-set* semantics is used to combine the mappings. On the other hand, it can be shown that a more lax monotonicity requirement, referred to as *sum-max monotonicity* [21], holds in both cases. Therefore, AR2Q also provides a modified version of the *horizon-based rank join* (HR-Join) algorithm presented in [21] designed to handle ranked joined joins when sum-max monotonicity holds.

6) Join Ordering: Existing SPARQL optimizers, such as [23], [22], collect and use statistics about the selectivities of the triple patterns. The optimization strategies used in AR2Q, on the other hand, rely on two characteristics inherent to each pattern of a BGP: the result inter-arrival times for each sub-query pattern and the *proximity* of a pattern to the

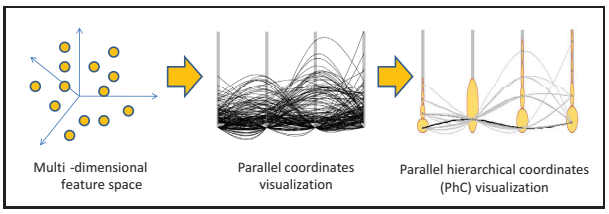


Figure 3. PC and PhC visualizations of a 4-dimensional feature space

other patterns in the query. AR2Q uses hybrid strategies that combine the join ordering recommendations provided by these two criteria. A key intuition is that a query node which has high proximity to the other query nodes has a higher likelihood of producing many (due to the multitude of the paths) short (due to the low weights) matches with the remaining query nodes and therefore should be placed as left in the query plan tree as possible. In particular, experiments showed that when the proximity score variance is high, proximity-score based decisions become highly effective in reducing the cost [6].

III. Features-of-Interest driven Result (SubGraph) Visualization in R^{2}DB

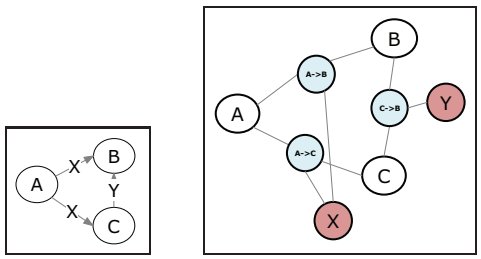
Results to a SPARankQL query are subgraphs of the underlying weighted RDF (R2DF) data graph. Visualizing these result sets is hard: even if the original data is plotted graphically and the few best results are highlighted on it, the user may not easily observe complex structural relationships on these graphs with respect to *relevant* nodes and edges.

To address this shortcoming, R^{2}DB also includes an innovative *features-of-interest* (FoI) based visualizing mechanism for helping explore large sets of (weighted) subgraphs relative to features relevant to the user.

A. Parallel Coordinates for Multidimensional Spaces

The basic parallel coordinates (PC) technique [12] for visualizing multidimensional spaces is based on the following data mapping strategy: the dimensions of the data (e.g., the attributes of a data relation) are represented as (often equi-spaced) vertical parallel lines or *parallel coordinates*. Given a dataset, each multi-dimensional data element (e.g., a tuple in a relation) is converted into a poly-line or a continuous curve which passes through the corresponding value points on the vertical coordinate lines (Figure 3). By mapping data elements onto the 2D space in such a way that similar elements are represented as similar poly-lines or curves on this space, PC aims to help users discern dominant patterns in the multi-dimensional data. For instance, a cluster of objects in the multi-dimensional space will appear as a dense cluster of lines that have similar flows in the 2D space. Similarly, outliers are also easily discernible.

A more recent proposal, *parallel hierarchical coordinates* (PhC [2]), allows the users to further reduce the visual clutter by leveraging value clustering hierarchies to decide how much detail to maintain for each of the data coordinates (Figure 3). The user can navigate over the resulting multi-resolution parallel coordinate space to understand distribution of the available data within the attribute space. To support this,



(a) original S (b) feature extension, S^e, of S

Figure 4. Feature-extension of graph S (weights are omited for clarity

PhC provides support for OLAP-like navigational operators, such as *drill-down*, where the user can navigate from a more general view to a more detailed view, by stepping down on a given hierarchy, and *roll-up*, which lets the user increase the amount of aggregation and clustering by climbing up a hierarchy. The user can also apply other data reduction operations, such as skylines, where only those tuples that are not *dominated* by any others in the dataset (commonly known as the *skyline* set) are included in the PhC visualization.

B. Feature-of Interest based Result Visualization

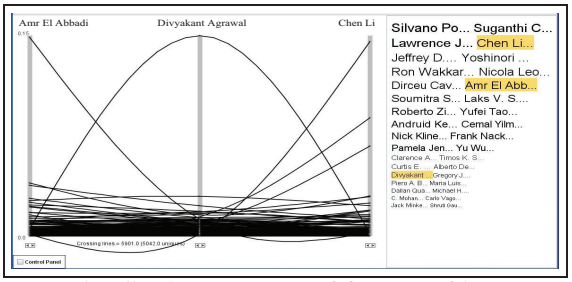
Let $G(V, E, l_V, l_E, w_E)$ be a node- and edge-labeled and edge-weighted (R2DF) graph. Any node $v \in V$, edge $e \in E$, node label $lv \in l_V$, or edge label $le \in L_E$ can be declared by the user as a feature of interest (FoI).

1) Basic FoI-Score: Let G be an R2DF graph, S be a subgraph of G, and f be an FoI. The *basic FoI-Score* of f in S ($fois_{basic}(f, S, \lambda())$) is computed by aggregating the corresponding edge weights on the subgraph S using an aggregation function, $\lambda()$ (e.g. min, max, or avg). For a node label feature, the weights of all incoming and outgoing edges and for an edge label the weights of all edges with the given label are aggregated.

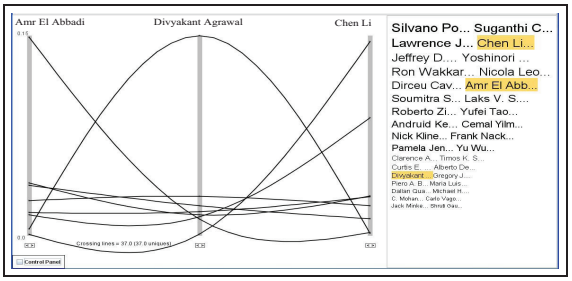
2) Structure-sensitive FoI-Score: Let G be an R2DF graph and S be a subgraph of G. The feature-extended *undirected* graph S^e is obtained by introducing (a) a virtual "edge" node for each edge feature and (b) a virtual "edge-label" node for each edge-label (Figure 4).

As we mentioned earlier, a node is said to have a *high proximity* to another, if the paths connecting the first node to the second in the graph are large in number, short in length, and have strong weights [3], [24], [25]. Thus, we define the *structure sensitive FoI-Score* of f_i relative to f_j on S ($fois_{ss}(f_i, f_j, S)$) as the personalized pagerank (PPR) [13] based proximity of f_i relative to f_j on S^e. Similarly, the *structure sensitive FoI-Score* of the graph S for the feature of interest f_i ($fois_{ss}(f_i, S)$) is computed as the pagerank [1] of f_i on S^e. Note that in both $fois_{ss}(f_i, f_j, S)$ and $fois_{ss}(f_i, S)$ computations, the edge weights (w_E) of the input R2DF graph are used for regulating the underlying random walk process.

3) FoI-based Visualization of Graphs: Given a graph, G, a set of subgraphs, $\mathcal{S}$, to be visualized, and a set of user provided features of interest $\mathcal{F}$, R^{2}DB first computes the (basic or PPR-based) FoI-scores for each subgraph-FoI pair $\langle S_i, f_j \rangle$, where $S_i \in \mathcal{S}$ and $f_j \in \mathcal{F}$.



(a) visualization wrt. a set of features of interest



(b) skyline visualization wrt. set of features of interest

Figure 5. (a) Features-of-Interest based visualization of a subgraph (result of a query over the DBLP RDF data set) and (b) the corresponding skyline visualization. In this example, the right hand side of the interface presents author node labels (that are structurally significant for the given subgraph) to help the user select the FoI set.

The FoI-visualization of $\mathcal{S}$ relative to $\mathcal{F}$ is a PhC visualization, where (a) each FoI feature $f_j \in \mathcal{F}$ is a vertical *parallel coordinate* and (b) each subgraph $S_i \in \mathcal{S}$ is a continuous curve which passes through the corresponding FoI-score points on the vertical coordinate.

If a user selects subgraph S_i for detailed visualization, then a second PhC window helps the user observe the FoI-score distribution of the features of the subgraph S_i relative to the selected features of interest (Figure 5(a)): here, each FoI feature $f_j \in \mathcal{F}$ is shown as a vertical coordinate and each result feature $f_h \in S_i$ is represented as a curve. The user can interact to obtain detailed or reduced (e.g. skylined, as in Figure 5(b), or clustered) views of the result with respect to the FoI set.

IV. Demonstration Scenarios and Conclusions

The demonstration highlights the key and distinguishing features of the R^{2}DB system for managing weighted RDF data and knowledge bases. As mentioned earlier, R^{2}DB is built based on the R2DF model that extends RDF triples with weights. We demonstrate SPARankQL, an extension of the SPARQL specification to support, among other features, (a) new path predicates to express advanced relationships (reachability) between nodes and (b) the ability to express ranked queries over weighted data. We also R^{2}DB's novel query processing and feature-of-interest based result visualization.

We demonstrate the various functionalities of R^{2}DB using different weighted knowledge bases. The first knowledge base we consider is the CoSeNa [5]; a weighted graph based on the bag of words from articles of the New York Times. The nodes of this graph represent words and the weighted edges represent the co-occurrence of two words in the same article. Secondly, we use a knowledge base containing author relationships based on a subset of the online DBLP database (`http://www.informatik.uni-trier.de/~ley/db/`) [16]. The nodes in this graph represent authors and the edges indicate the participation of two authors in the same publication. The weights of the edges in the knowledge base we constructed indicate the number of publications in which two authors co-participated.

Acknowledgements

We thank Yan Qi and Huiping Cao for the HR-Join operator and Luigi Di Caro for the initial PhC code.

References

[1] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. Proc. *WWW*, pp. 107-117, 1998.
[2] K. S. Candan, L. D. Caro, and M. L. Sapino. Phc: Multi-resolution visualization and exploration of text corpora with parallel hierarchical coordinates. *accepted to ACM TIST* , 2011.
[3] K. S. Candan and W.-S. Li. Using *Random walks* for mining web document associations. In *PAKDD*, pages 294–305, 2000.
[4] I. Cantador, M. Fernndez, D. Vallet, P. Castells, J. Picault, and M. Ribire. A multi-purpose ontology-based approach for personalised content filtering and retrieval. *Studies in Computational Intelligence* v.9,, 2008.
[5] M. Cataldi, C. Schifanella, K. S. Candan, M. L. Sapino, and L. Di Caro. Cosena: a context-based search and navigation system. Proc. *MEDES'09*, pp. 218–225, 2009.
[6] J. P. Cedeño and K. S. Candan. R^{2}DF framework for ranked path queries over weighted rdf graphs. *WIMS 11*, 2011.
[7] S. Chaudhuri, L. Gravano, and A. Marian. Optimizing top-k selection queries over multimedia repositories. *IEEE TKDE*, 16(8), 2004.
[8] E.Q.V. Martins, and M. M. B. Pascoal. A new implementation of yen's ranking loopless paths algorithm. *4OR*, 1(2), 2003.
[9] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. *PODS'01*, 2001.
[10] H. He and A. Singh. Graphs-at-a-time: query language and access methods for graph databases. Proc. *SIGMOD'08*, pp. 405-418, 2008.
[11] http://g-store.sourceforge.net/
[12] A. Inselberg and B. Dimsdale. Parallel coordinates: a tool for visualizing multi-dimensional geometry. Proc. *VIS'90*, pp. 361–378, 1990.
[13] G. Jeh and J. Widom. Scaling personalized web search. Proc. *WWW'03*, pp 271–279, 2003.
[14] C. Jin, S.S. Bhowmick, X. Xiao, J. Cheng, and B. Choi. GBLENDER: towards blending visual query formulation and query processing in graph databases. Proc. *SIGMOD'10*,pp. 111-122, 2010.
[15] G. Klyne and J. J. Carroll. Resource description framework (RDF): Concepts and abstract syntax. World Wide Web Consortium, Recommendation REC-rdf-concepts-20040210, February 2004.
[16] M. Ley. The dblp computer science bibliography: Evolution, research issues, perspectives. In *SPIRE*, *LNCS 2476*, pp. 1–10, 2002.
[17] C. Li, K. C.-C. Chang, I. F. Ilyas, and S. Song. Ranksql: Query algebra and optimization for relational top-k queries. Proc. *SIGMOD'05*, pp 131–142, 2005.
[18] openRDF.org. Sesame, 2007.
[19] E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF. W3C Recommendation, 2008.
[20] Y. Qi, K. S. Candan, and M. L. Sapino. Ficsr: feedback-based inconsistency resolution and query processing on misaligned data sources. Proc. *SIGMOD'07*, pp. 151–162, 2007.
[21] Y. Qi, K. S. Candan, and M. L. Sapino. Sum-max monotonic ranked joins for evaluating top-k twig queries on weighted data graphs. Proc. *VLDB'07*, pp. 507–518, 2007.
[22] M. Stocker, A. Seaborne, A. Bernstein, C. Kiefer, and D. Reynolds. Sparql basic graph pattern optimization using selectivity estimation. Proc. *WWW'08*, pp. 595–604, 2008.
[23] Tdb - jena semantic framework http://openjena.org/wiki/TDB.
[24] H. Tong and C. Faloutsos. Center-piece subgraphs: problem definition and fast solutions. Proc.*KDD'06*, pp. 404–413,2006.
[25] H. Tong, C. Faloutsos, and Y. Koren. Fast direction-aware proximity for graph mining. Proc. *KDD'07*, pp. 747–756, 2007.