FedS: Towards Traversing Federated RDF Graphs

Qaiser Mehmood, Alokkumar Jha, Dietrich Rebholz-Schuhmann, and Ratnesh Sahay

Insight Centre for Data Analytics,
National University of Ireland, Galway
{`qaiser.mehmood, alok.kumar, rebholz, ratnesh.sahay`}`@insight-centre.org`

Abstract. Traversing paths within a graph is a well-studied problem and highly intractable especially with large-scale graphs. In case of multiple graphs, the standard practice is to merge distinct graphs in a centralised way to evaluate the existence of paths between given entities (or nodes). In the biomedical domain counting and retrieving the number of paths (or edges) that connect two biological entities is a highly desirable feature expected from graph databases. Therefore, non-standard solutions exist that count and retrieve paths from a single graph database. From the standard perspective, SPARQL 1.1 provides the navigational feature called Property Paths (PP) which is limited only to a single RDF graph where path existence can be evaluated between pair of nodes. In this paper, we propose a federated approach – called FedS – that retrieves paths from multiple RDF triple stores. Our key idea is to partially delegate computational load to a set of federated RDF triple stores in a peer-to-peer manner thus reducing the computational burden on a centralised query processing server. In our preliminary investigation, we evaluate FedS against the state-of-the-art approaches that provide the path counting feature over single RDF graph. We compare FedS against these approaches in terms of performance (overall path retrieval time) and result completeness, i.e., number of paths retrieved.

1 Introduction

It is very common in the biomedical domain that two biological entities (gene, protein, pathway, drug, etc.) are associated via several properties and therefore, discovery of such connecting properties (or paths) between two given biological entities is often a fundamental requirement behind novel biological innovations [12, 5]. The recent expansion of "Web of Data", particularly the life sciences portion of the Link Open Data (LOD) Cloud[1], has thrown entirely new dimensions of challenges for the graph-based knowledge and data integration communities. The life-sciences community has been very active within the Linked Open Data (LOD) movement: 41 datasets on the LOD cloud are classified as specialising in the "Life Sciences" domain and 70 SPARQL endpoints have been made available by these publishers, most prominently by the Bio2RDF[2] [2, 1] and Linked Life Data[3] initiatives. Other general-knowledge datasets, such as DBpedia and corresponding triple stores, also contain rich information about

[1] http://lod-cloud.net/
[2] http://bio2rdf.org/
[3] http://linkedlifedata.com/

the life sciences. Although, the underlying SPARQL endpoints behind the LOD Cloud has created a mashup of linked data connected through various linking properties (e.g., *xRef, owl:sameAs, x-relation*) [9]. Unfortunately, there is no mechanism available – at the time of writing this article – that can traverse across multiple RDF triple stores in a federated manner to retrieve properties (or paths) that connect two given nodes. Manually identifying which of the LOD datasets contain the connecting properties is impractical, cumbersome, and a time-consuming process. For instance, in the current scenario, a SPARQL 1.1 property path query (shown in the Listing 1.1)

Listing 1.1: Property Path query in SPARQL1.1

```
Prefix : <urn:exe:>
SELECT ?s ?p ?o
{ :cnv (:|!:)* ?s. ?s ?p ?o. ?o (:|!:)*  :gene }
Group By ?s ?p ?o
```

would return all the *genes* which *:cnv (copy number variation)* connects to via any property in a single RDF graph. Although the SPARQL 1.1 version supports federation (via the SERVICE operator) and property path features, it is surprising that existing approaches and open source triple stores like Virtuoso and Jena does not provide any solution to federate path queries over distributed graphs. Similarly, there is numerous research already carried out in traversing and/or finding shortest paths in a graph [13, 6]. In the context of Linked Data, the European Semantic Web Conference (ESWC) in 2016 hosted a challenge [15] to find "Top-K Shortest Path in Large Typed RDF Graphs" where the main focus of all the competing participants has been on the extending or optimising the state-of-the-arts graph traversal algorithm over single RDF graph; none were aimed to navigate across federated graphs. Therefore, to find the path between source and target, the centralised approaches adopted by current systems pose some challenges such as; (i) user must know the priori knowledge of underlying schema of data to query on, (ii) requires to merge whole data into a single graph, which is a cumbersome task, (iii) copied data needs to be synchronized, and (iv) it lacks the opportunity to query the up-to-date and fresh data. while on the other-side federation provides the opportunity to tackle these challenges. Hence, to address these challenges we propose FedS that takes as input source (subject), target (object) nodes, and a list of triple stores to federate upon; and returns a list of paths that connect the source and target nodes across the federated triple stores.

In our previous work [17], we proposed an extension of SPARQL which allows finding the top-k shortest paths on a single compressed RDF graph – in the HDT [3] format – using the property path expression. In this paper, we propose FedS that federates across multiple triple stores by (i) selecting prospective triple stores where source and target nodes are available; (ii) retrieving paths from different triple stores; and finally (iii) merging all the retrieved paths that connect source and target nodes hosted in different triple stores. We start the paper by presenting a motivation scenario that aims to retrieve paths between source and target nodes hosted in different triple stores. We then discuss the related works of traversing and retrieving paths in graph databases. We then present the

FedS architecture and description of its four core components. We provide an evaluation of FedS compared to the state-of-art approaches. Finally, we present our conclusion and various routes to optimise the navigation across federated RDF graphs.

2 Motivation Scenario - Cancer Genomics

In order to understand the cancer progression, it is often the case that several genetic features, diseases, medical history, etc. are studied together, therefore, one of the key challenge in cancer genomics – a cornerstone of precision medicine – is to discover gene-disease-drug associations. Such novel associations provide insight into the drug development process tailored specifically for an individual patient (or a group of patients) targeting prevention, diagnosis and treatment of the diseases [18].

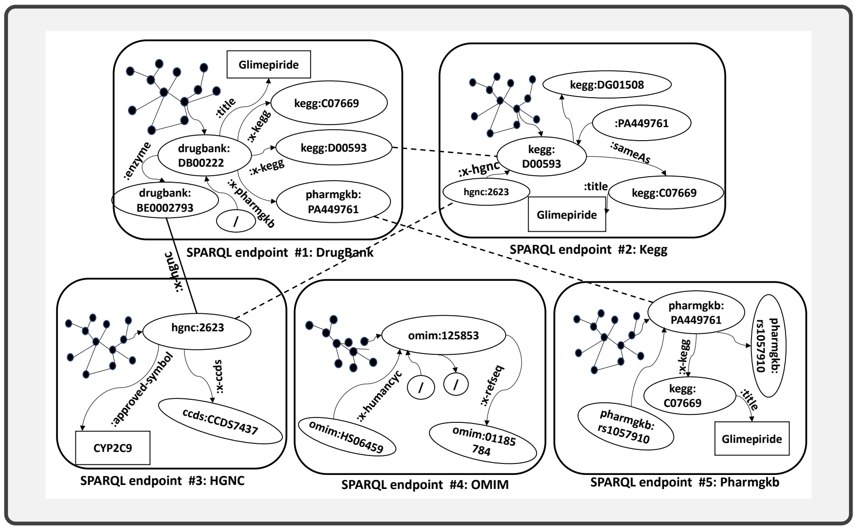

Fig. 1: Navigation Across Federated SPARQL endpoints

For instance, consider a scenario where a biomedical expert is trying to discover the paths between a drug ($drugbank : DB00222$) and a drug compound ($kegg : C07669$). The Figure 1 shows a group of datasets (DrugBank, KEGG, Hgnc, OMIM, and Pharmgkb) hosted at five different SPARQL endpoints. The source drug ($drugbank : DB00222$) is located in the SPARQL endpoint for DrugBank, whereas the target drug compound ($kegg : C07669$) exists in the SPARQL endpoints 1 (DrugBank), 2 (KEGG), and 5 (Pharmgkb). The source ($drugbank : DB00222$) – Glimepiride – is an antidiabetic drug whereas the target ($kegg : C07669$) represents a drug compound associated with Glimepiride.

If the biomedical expert needs to establish associations between the drug and its compound, he/she can find the direct correlation by querying the SPARQL endpoint for DrugBank. However, for further analysis to understand the mechanism of a drug compound and affected biological pathways, the expert needs to explore and discover associations in various others biomedical datasets. Figure 1 shows three paths starting from the $drugbank : DB00222$ (SPARQL endpoint 1) where the target ($kegg : C07669$) is available in (i) the DrugBank endpoint itself via the *:x-kegg* property; (ii) the Kegg endpoint via the *:x-kegg* and *:sameAs* properties; and (iii) the Pharmgkb endpoint via the *:x-pharmgkb* and *:x-kegg* properties. The fourth path at the HGNC endpoint contains a node ($hgnc : 2623$) via the *:enzyme* and *:x-hgnc* properties, however, the target node ($kegg : C07669$) doesn't exist in this endpoint.

We believe that a technology that can enable querying paths/associations among two or more biological entities across distributed repositories would be a great help to biologist and practitioners working in the cancer genomics as well as in the larger healthcare and life sciences area.

3 Preliminaries

Consider a network of distributed datasets where each dataset loaded with $G = (V, P)$ directed graph. V is the set of vertices and P is the set of paths (edges). The vertices (v_i, v_j) are associated through a set of paths (edges) p_ij. If there exists a path between (v_i, v_j) then v_j is the *successor* of v_i and v_i is the predecessor of v_j. For RDF graph, we define as follows:

Definition 1 ***(RDF Triple & Graph)*** *The set of RDF terminologies consists of the set of IRIs I, the set of blank-nodes B and the set of literals L. An RDF Triple $T := (s; p; o)$ is an element of the set $G := (I \cup B) \times I \times (I \cup L \cup B)$. The set G is a finite set of triples called RDF graph. A RDF graph G_i of a single triple T_i represents a vertex (subject) s_i associated with another vertex (object) o_i via a path (property) p_i.*

In a federated environment, if a path between source and target vertices does not exist within a local single graph, other remote graphs are scanned and queried to find the path existence between given vertices over the network.

Definition 2 ***(FedS Source Selection)*** *In a federated query environment, given a triple pattern T with subject s_i and object o_i vertices in a bgp in a query Q executed against data sources $\mathcal{D}$, the set of relevant sources for T in $\mathcal{D}$ is the set $\mathcal{R}_t \subseteq \mathcal{D}$ of data sources that can provide answers when queried with T. We use the notation $\mathcal{R}_T$ to denote the set of revelent data sources for source (subject) and target (object) nodes and use $\mathcal{R}$ when the context does not require specifying the triple patterns.*

FedS performs a source selection process where a set of relevant data sources for a given query are discovered by scanning all the given data sets $\mathcal{D}$ and input triple pattern T.

Definition 3 ***(FedS Reachability)*** *In case of a labeled directed RDF graph G, the reachability relation is the transitive closure of RDF properties p(s,o) such that for the set of all ordered paired of subjects (s) and objects (o) there exists a sequence of subjects and objects $s = v_o$, $v_i, ..., k = o$ where the property $p(v_{i-i}, v_i)$ is in p(s,o) for all $1 \leq i \leq k$.*

For a given query to find the path between (s,o), if these two are connected through any number of paths $p_i, ..., n$ we say that path exists and source(s) and target(o) are reachable. While if there is no path between (s,o) it means that (s,o) is not reachable.

4 Related Work

Numerous amount of work has been done on the graph navigation and pathfinding problems [13] and variety of algorithms have been proposed to compute the shortest path within the given networks. Similarly, there is an extensive set of works [6, 14, 7, 4] related to the performance of pathfinding algorithms in very large graphs. They employ different techniques to get the optimised performance, for instance, in case of [7, 4], the algorithms only work on compressed datasets stored in customised databases (e.g., HDT, RDF-3X), while others have worked on the graph partitioning problem. Our previous work [17] along with others [8, 16, 10, 11] proposed extensions for SPARQL 1.1 property paths, but none of them consider multiple triple stores during the graph navigation process. Therefore, we have no work to compare that does graph traversal over federated triple stores. Thus the motivation for our research is to investigate, how to enable path evaluation over federated triple stores. To the best of our knowledge, FedS is the first attempt to retrieve (evaluate) paths across the graphs hosted in federated triple stores.

5 FedS

FedS approach works as peer-to-peer network of triple stores where every triple store has FedS running on it. FedS extends previous work [17] to federate path traversal across the network. The FedS architecture is summarised in Figure 2, which shows its four core components: (i) Source Selection: performs source selection based on the availability of a source node within triple stores; (ii) Path Computation: once a subset of triple stores are identified that host the source node, path traversal starts from one of the selected triple srtores; (iii) Path Federation: when the path traversal starts at host triple store, the FedS probes other RDF stores – during the iteration – wherever the target node is available; and finally (iv) Path Merger: it aggregates all the path retrieved from different triple stores and prepare a list of paths between the source and target nodes.

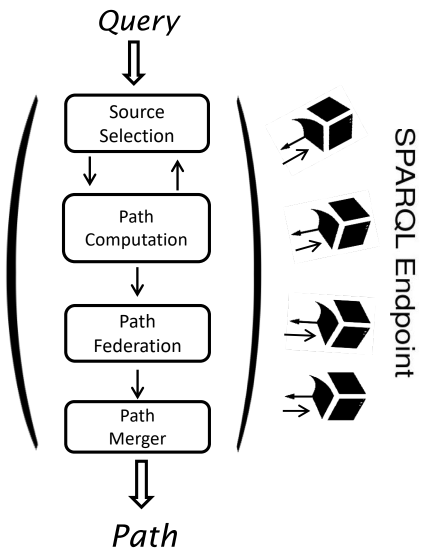

Fig. 2: FedS Architecture

Source Selection: FedS performs a partially index free source selection of relevant triple stores (see definition 2), where only the address IRIs are stored. The source selection is performed by using the SPARQL ASK query that returns the availability of a source and target in different triple stores. For example, given the list of five RDF stores (DrugBank, HGNC, PharmGKB, OMIM, and KEGG) [see Figure 1], the source selection process identifies three triple stores (DrugBank, KEGG, Pharmgkb) where the source ($drugbank : DB00222$) and target ($keg : C07669$) nodes exist. Then, one of the identified triple store – where the source node is found – is randomly selected to initialize the path traversal process.

Path Computation The traversal starts from the identified triple store that contains the source node and navigates along all the paths *(successors)* and check the reachability between source and target nodes (see definition 3). If the target node is found in the hosted triple store, it feeds (reports) the retrieved path to the "Path Merger" component. However, if path from source to target node is not reachable – within the hosted triple store – we defined it as a *partial path*. The partial paths are created from the source to all its successors up to the leaf (last) nodes in the current triple store. For instance, two simple partial paths; *kegg:D00593* and *pharmgkb: PA449761* are one hop away from source *DB00222*, therefore are the leaf nodes. However, there always be complex partial paths where many nodes exist from root to last node (source) to its last node. Feds considers all the nodes of partial paths to check their availability in other datasets and how this happens is explained in "Path Federation" component. Further, there may also exist indirect paths between source and target. For instance, in figure 1 enspoint 1 is linked with endpoint 3 through DB00222$\rightarrow$:anzyme $\rightarrow$ BE0002793 $\rightarrow$ x:hgnc $\rightarrow$ hgnc:2623. Endpoint 3 does not contain the target node C07669, however, hgnc:2623 is connected to endpoint 2 where actual target node C07669 does exist. To compute the indirect paths, FedS performs the following steps; (i) partial path is created DB00222$\rightarrow$:anzyme $\rightarrow$ BE0002793 $\rightarrow$ x:hgnc $\rightarrow$ hgnc:2623, (ii) hgnc:2623 is not target node therefore FedS ASK this node and finds that endpoints 2 has hgnc:2623. From endpoint 2 partial path – hgnc:2623 $\rightarrow$:xhgnc $\rightarrow$ D00593 $\rightarrow$:sameAs $\rightarrow$ C07669 – is returned and is reported to the component "Path Merger".

Path Federation: This component iterates over the list of partial paths where each element (node) involved in partial path along with actual target node is checked (using the SPARQL ASK construct) for their availability in the already identified triple stores. Upon receiving the federated request, the traversal process starts in all those triple stores which contain the corresponding nodes and returns the retrieved paths towards the endpoint from where the request was dispatched. For example, the two dashed lines between **DrugBank-Kegg** and **DrugBank-Pharmgkb** endpoints shows the availability of nodes ($kegg : D00593$ and $pharmgkb : PA449761$) in these two endpoints.

Path Merger: This component stores and merge all the partial paths and the paths retrieved from different SPARQL endpoints. When the paths are received from different SPARQL endpoints, this component concatenates each retrieved path to the corresponding node in the partial list such that a complete path is completed (i.e., partial from local and partial from remote enapoint) paths from source to target nodes. For example, two direct complete paths are: (i) DB00222$\rightarrow$ **:x-pharmgkb** $\rightarrow$PA449761 $\rightarrow$ **:x-kegg** $\rightarrow$ C07669; and (ii) DB00222$\rightarrow$ **:x-kegg**$\rightarrow$ D00593$\rightarrow$ **:sameAs**$\rightarrow$ C07669]). There also exists an indirect path which is;[DB00222$\rightarrow$ **:anzyme** $\rightarrow$ BE0002793 $\rightarrow$ **x:hgnc** $\rightarrow$ hgnc:2623$\rightarrow$ **x:hgnc** $\rightarrow$ D00593 $\rightarrow$ **:sameAs** $\rightarrow$ C07669]

The main objective of FedS is to retrieve paths over the federated network of triple stores and the major advantages are: (i) FedS doesn't need priori knowledge of underlying schema of RDF graphs hosted in different SPARQL endpoints; (ii) In order to traverse the paths, FedS does not require to merge the federated RDF graphs into a single graph; and (iii) finally, FedS is still be able to compete with state-of-the-art approaches.

6 Results and Discussion

The experimental setup comprises of datasets (i.e., SPARQL endpoints) and six input queries. we compared Feds against three systems (Virtuoso, Blazegraph, and HDT-Bidirectional-TopK).

Datasets: We have downloaded the release-4 of Bio2RDF datasets[4]. The data cumulatively is around 3.89 GB – DrugBank (size 1.18GB), KEGG (size 471.9MB), PharmaGKB (size 29.2MB), OMIM (size 1.61GB), and HGNC (size 592MB) – with 22,36,32,57 triples, 2,343,770 subjects, 334 predicates and 81,599,44 objects. Table 1 shows the number of triples, subjects, predicates, and objects in each dataset and the hardware specs of five desktop machines used to conduct the experiments are shown in Table 2.

Table 1: Datasets statistics

Dataset	Size	Triples	Subject	Predicates	Objects
Drugbank	1.18GB	5151714	421348	104	2472011
Kegg	479.1MB	3281579	358844	63	1835508
Pharmgkb	29.2MB	191379	19905	32	123336
Omim	1.61GB	9687186	1127394	93	1415364
HGNC	592MB	4051399	416279	42	2313725

Queries: The table 3 shows six evaluation queries. The column two in Table 3, shows the number of datasets (SPARQL endpoints) selected for each query. For example, in the case of **Q1** DrugBank and Pharmgkb are needed to retrieve the complete paths. Similarly, **Q5** needed 3 SPARQL endpoints (DrugBank, Kegg, and Pharmgkb) to retrieve all three paths. The "Hops" column denote the number of properties (named edges) between source and target nodes.

Table 2: Specifications of virtual machines used in experriments

OS	Data loaded	RAM	Hard disk	Processor
MAC	Omim	16GB	500GB	2.6 GHz Intel Core i5
Ubuntu	Drugbank	32GB	500GB	2.9 GHz Intel Core i7
Windows	Kegg	8GB	250GB	2.2 GHz Intel Core i7
Ubuntu	Hgnc	8GB	300GB	2.5 GHz Intel Core i5
Windows	Pharmgkb	16GB	250GB	2.2 GHz Intel Core i5

Performance Analysis: In order to compare against the triple stores (Virtuoso, Blazegraph, and HDT-BidirectionalTopK) which support the SPARQL 1.1 Property Path specification. We performed following steps: (i) merged all the data within five SPARQL endpoints into a single graph and loaded it into each triple store (*i.e.*, Virtuoso, Blazegraph, HDT-BidirectionalTopK); and (ii) executed SPARQL 1.1 property path query without specifying any predicate in path expression (e.g., (: |! :)∗) on each triple store. As shown in the Figure 3, HDT-Bidirectional-TopK outperforms all other systems simply because it works only on HDT-based compressed RDF data rather than raw RDF data.

FedS, Virtuoso, and Blazegraphs work only on raw RDF data and therefore results are comparable. The key point to be noted here is: FedS has to cope with additional messaging over the network (*e.g.* network cost due to SPARQL ASK

[4] http://download.openbiocloud.org/release/4/

Table 3: Six Input Queries and Results

Query	Dataset selected	Source	Target	Paths	Hops
Q1	Drugbank — Pharmgkb	drugbank:DB00072	clinicaltrials:NCT01959490	1	2
Q2	Drugbank—HGNC	drugbank:DB01268	hgnc.symbol:FLT3	1	3
Q3	Drugbank	drugbank:DB00134	kegg:D04983	1	2
	Drugbank—Kegg	drugbank:DB00134	kegg:D04983	1	1
Q4	Drugbank	drugbank:BE0002362	hgnc.symbol:CYP3A5	1	1
Q5	Drugbank	drugbank:DB00222	kegg:C07669	1	1
	Drugbank—Kegg	drugbank:DB00222	kegg:C07669	1	2
	Drugbank—Pharmgkb	drugbank:DB00222	kegg:C07669	1	2
	Drugbank—HGNC—Kegg	drugbank:DB00222	kegg:C07669	1	4
Q6	Drugbank	omim:147470	hgnc.symbol:IGF2	3	3x1

queries, machines with different specs, source selection, federation request, etc.). We also noticed that Virtuoso and Blazegraph were not able to return the results for **Q6**. In case of Virtuoso, all queries were showing *time-out* if the queries were executed without specifying the named graphs[5].

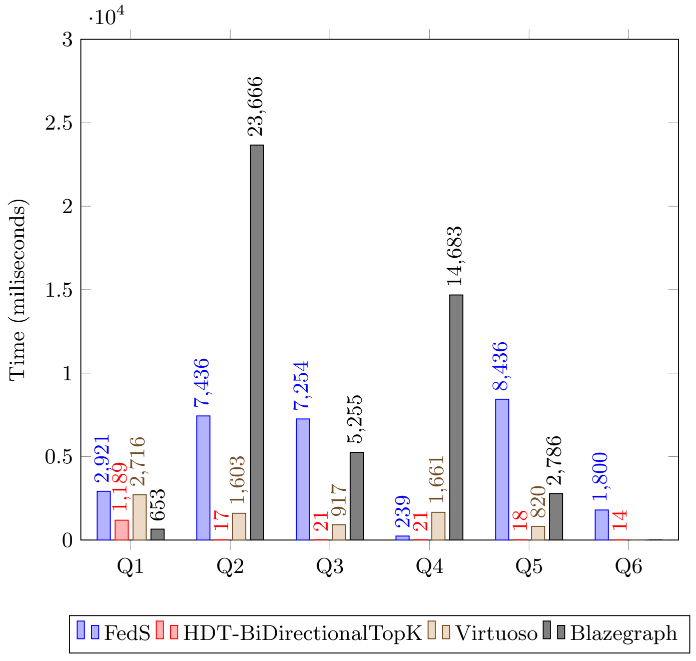

Fig. 3: Comparing total query execution time

Path Analysis: Table 4 shows the comparison of *total number of paths* retrieved and their corresponding *path length* (*i.e.*, Hops). In terms of result completeness – as triple stores are designed to retrieve complete sets of path in

[5] https://www.w3.org/TR/rdf-sparql-query/#namedGraphs

a single graph – for all the six queries, FedS retrieved the equal number of paths in a federated environment.

Table 4: Comparison of the Total Paths **#TP** and Path Hops **#PH**

	FedS		HDT-BiDirectionalTopK		Virtuoso		Blazegraph	
Query	#TP	#PH	#TP	#PH	#TP	#PH	#TP	#PH
Q1	1	2	1	2	1	2	1	2
Q2	1	3	1	3	1	3	1	3
Q3	1	2		2	1	2	1	2
	1	1	1	1	1	1	1	1
Q4	1	2	1	2	1	2	1	2
Q5	1	2	1	2	1	2	1	2
	1	2	1	2	1	2	1	2
	1	1	1	1	1	1	1	1
	1	4	1	4	1	4	1	4
Q6	3	3x1	3	3x1	-	-	-	-

7 Conclusion and Future work

In this paper we propose FedS, a path traversal approach that federates across multiple SPARQL endpoints. This work is motivated by the needs of biomedical domain to find associations (paths) across different biological entities. The current SPARQL 1.1 Property Path specification and the standard traversal algorithms(BFS, DFS, A*, etc.) assume (or require) a single graph – or many graphs merged into a centralised graph – for graph traversal. FedS proposes a four step process that enables graph traversal in a federated environment. Our initial evaluation results are encouraging where FedS retrieves all the paths in a competing query processing time when compared to state-of-the-art triple stores approaches. In terms of future work, there are a number of possible routes to optimise the current four step process (i) we plan to implement a ranking mechanism in the source selection process to select the *top-k* SPARQL endpoints to initialise the path navigation process; (ii) next obvious step is to devise a method that ranks (*top-k*) the retrieved paths between source and target nodes; (iii) we plan to include larger data sets with significant number of paths (i.e., 10+) between source and target nodes; and finally, (iv) the current implementation navigates only in the forward direction, to further optimise, we plan to implement navigation in both the directions.

Acknowledgements

The work presented in this research paper has been funded by Science Foundation Ireland under Grant No. SFI/12/RC/2289.

References

1. *Bio2RDF Release 3: A larger, more connected network of Linked Data for the Life Sciences*, volume 1272 of *CEUR Workshop Proceedings*, Riva del Garda, Italy, 2014. CEUR-WS.org.
2. François Belleau and et al. Bio2rdf: Towards a mashup to build bioinformatics knowledge systems. *Journal of Biomedical Informatics*, 2008.

3. Javier D. Fernández, Miguel A. Martínez-Prieto, Claudio Gutiérrez, Axel Polleres, and Mario Arias. Binary RDF representation for publication and exchange (HDT). *J. Web Sem.*, 19:22–41, 2013.
4. Erwin Filtz, Vadim Savenkov, and Jürgen Umbrich. On finding the k shortest paths in rdf data. In *5th International Workshop (IESD 2016) co-located with the (ISWC 2016)*, volume 18, 2016.
5. Jianjiong Gao, Bülent Arman Aksoy, Ugur Dogrusoz, Gideon Dresdner, Benjamin Gross, S Onur Sumer, Yichao Sun, Anders Jacobsen, Rileen Sinha, Erik Larsson, et al. Integrative analysis of complex cancer genomics and clinical profiles using the cbioportal. *Science signaling*, 6(269):pl1–pl1, 2013.
6. Andrew V Goldberg. Point-to-point shortest path algorithms with preprocessing. In *SOFSEM*. Springer, 2007.
7. Andrey Gubichev and et al. Fast and accurate estimation of shortest paths in large graphs. In *Proceedings of the 19th ACM CIKM*. ACM, 2010.
8. Andrey Gubichev and Thomas Neumann. Path query processing on very large rdf graphs. In *WebDB*. Citeseer, 2011.
9. Wei Hu, Honglei Qiu, and Michel Dumontier. Link analysis of life science linked data. In *14th - ISWC 2015*, volume 9367 of *Lecture Notes in Computer Science*, pages 446–462, PA, USA, 2015. Springer.
10. Krys J Kochut and Maciej Janik. Sparqler: Extended sparql for semantic association discovery. In *European Semantic Web Conference*, pages 145–159. Springer, 2007.
11. Egor V Kostylev, Juan L Reutter, Miguel Romero, and Domagoj Vrgoč. Sparql with property paths. In *International Semantic Web Conference*, pages 3–18. Springer, 2015.
12. Zoé Lacroix, Hyma Murthy, Felix Naumann, and Louiqa Raschid. Links and paths through life sciences data sources. In *Data Integration in the Life Sciences, First International Workshop, DILS 2004*, Lecture Notes in Computer Science, Leipzig, Germany, 2004. Springer.
13. Amgad Madkour, Walid G Aref, Faizan Ur Rehman, Mohamed Abdur Rahman, and Saleh Basalamah. A survey of shortest-path algorithms. *arXiv preprint arXiv:1705.02044*, 2017.
14. Rolf H Möhring, Heiko Schilling, Birk Schütz, Dorothea Wagner, and Thomas Willhalm. Partitioning graphs to speedup dijkstra's algorithm. *Journal of Experimental Algorithmics (JEA)*, 11:2–8, 2007.
15. Ioannis Papadakis and et al. Top-k shortest paths in large typed RDF datasets challenge. In *Semantic Web Challenges - Third SemWebEval Challenge at ESWC 2016, Heraklion, Crete, Greece, May 29 - June 2, 2016*. Springer, 2016.
16. Martin Przyjaciel-Zablocki and et al. Rdfpath: Path query processing on large rdf graphs with mapreduce. In *ESWC*. Springer, 2011.
17. Vadim Savenkov, Qaiser Mehmood, Jürgen Umbrich, and Axel Polleres. Counting to k, or how sparql1.1 property paths can be extended to top-k path queries. SEMANTICS 2017.
18. Richard Simon and Sameek Roychowdhury. Implementing personalized cancer genomics in clinical trials. *Nature Reviews Drug Discovery*, 12(5):358–369, 2013.