P3RPQ: Pregel-Based Parallel Provenance-Aware Regular Path Query Processing on Large RDF Graphs

Yueqi Xin^{†§}, Bingyi Zhang^{†§}, Xin Wang^{†§*}, Qiang Xu^{†§}, Zhiyong Feng^{‡§}

[†]School of Computer Science and Technology, Tianjin University, Tianjin, China

[‡]School of Computer Software, Tianjin University, Tianjin, China

[§]Tianjin Key Laboratory of Cognitive Computing and Application, Tianjin, China
{xinyueqi, byzhang, wangx, xuqiang3, zyfeng}@tju.edu.cn

ABSTRACT

This paper proposes a novel method for answering Pregel-based Parallel Provenance-aware Regular Path Queries (P3RPQ) on large RDF graphs. Our method is developed using the Pregel framework, which utilizes Glushkov automata to keep track of the matching process of RPQs in parallel. Meanwhile, four optimization strategies are devised, which can reduce the response time of the basic algorithm dramatically and overcome the counting paths problem to some extent. The experiments are conducted to verify the performance of our algorithms on both synthetic and real-world datasets.

1 INTRODUCTION

RDF is a flexible graph-like data model which can represent large-scale knowledge bases. *Regular path queries* (RPQs) are recognized as one of the fundamental operations for querying graph data, which can explore RDF graphs in a navigational manner. Answering an RPQ Q over an RDF graph T is to find a set of pairs of resources (v_0, v_n) such that the path between v_0 and v_n satisfies Q . For example, an RPQ $Q_1 = (x, (\text{predecessor}|\text{father})^+, y)$ on an RDF graph T_1 in Fig. 1(a) asks to find the set of (v_0, v_n) such that v_0 can navigate to v_n via one or more predecessor or father edges. The answers are shown in Fig. 1(b), from which one cannot figure out how each resource is reached from another. In order to provide the *provenance* of each answer, we focus on the *provenance-aware* RPQs that return a *subgraph* of the RDF graph consisting of all the “witness triples”. For Q_1 , the subgraph is exactly T_1 , which effectively encode the conventional answers of Q_1 shown in Fig. 1(b).

Dey et al. [1] had done the first work to investigate provenance-aware RPQs by translating them into Datalog queries, which is hardly scalable for evaluating large graph data. Wang et al. [3] employed product automata to solve this problem, which may result in excessive communication cost for large-scale RDF graphs.

Compared with the previous approaches, our main contributions include: (1) we propose a *Pregel-based* parallel method, which is the first work that answers provenance-aware RPQs using the Pregel

Figure 1: An RDF graph T_1 and answers of RPQ Q_1

graph parallel computing framework; (2) four optimization strategies are presented to reduce the overhead of the basic algorithm and partly address the *counting paths* problem; and (3) extensive experiments were conducted to verify the efficiency and scalability of our proposed method on both synthetic and real-world datasets.

2 PROBLEM AND OUR APPROACH

An RDF graph T is a set of triples, each triple (s, p, o) representing a statement of a relationship (predicate) p between a subject s and an object o . An RDF graph can also be modeled as a directed labeled graph $T = (V, E, l)$ with node set V , edge set E , and the function l that assigns a label to each edge. For each $(s, p, o) \in T$, there exists an edge $e = (s, o) \in E$ and $l(e) = p$. Given an RPQ $Q = (x, r, y)$ over T , r is a *regular expression* over the set Σ of edge labels. The regular expression is recursively defined as $r ::= \varepsilon \mid p \mid r/r \mid r| r^*$, where $p \in \Sigma$ and $/$, $|$, and $*$ are concatenation, alternation, and the Kleene closure, respectively. The shorthands r^+ for r/r^* and $r?$ for $\varepsilon|r$ are allowed. $L(r)$ denotes the language expressed by r and $\lambda(\rho)$ is the label of path ρ . The answer set to conventional semantics of an RPQ Q over an RDF graph T is defined as $\llbracket Q \rrbracket_T = \{(v_0, v_n) \mid \exists \text{ a path } \rho \text{ in } T \text{ from } v_0 \text{ to } v_n \text{ s.t. } \lambda(\rho) \in L(r)\}$, whereas the provenance-aware answer set of Q is defined as $\llbracket Q \rrbracket_T^p = \{(s, p, o) \mid \exists \text{ a path } \rho \text{ in } T \text{ s.t. } (s, p, o) \text{ is an edge in } \rho \text{ and } \lambda(\rho) \in L(r)\}$.

The computation in Pregel consists of a series of supersteps [2], which is processed at each vertex *in parallel*. The Pregel-based evaluation for provenance-aware RPQs is shown in Algorithm 1, in which we construct a *Glushkov automaton* A_Q based on r in an RPQ $Q = (x, r, y)$. Let $Pos(r) = \{1, \dots, |r|\}$ be the set of positions in r , where $|r|$ is the length of r , and $i \in Pos(r)$ is the index of $r[i]$. The function $first(r)$ (resp. $last(r)$) is the set of positions in r that can match the first (resp. last) symbol of a string in $L(r)$. The function $follow(r, i)$ is the set of positions in r that can follow position i when matching some string in $L(r)$. Thus, A_Q is defined as a 5-tuple $(St, \Sigma, \delta, q_0, F)$, where (1) $St = \{0\} \cup Pos(r)$, (2) Σ is

* Xin Wang is the corresponding author.

This paper is published under the Creative Commons Attribution 4.0 International (CC BY 4.0) license. Authors reserve their rights to disseminate the work on their personal and corporate Web sites with the appropriate attribution.

WWW '18 Companion, April 23–27, 2018, Lyon, France

© 2018 IW3C2 (International World Wide Web Conference Committee), published under Creative Commons CC BY 4.0 License.

ACM ISBN 978-1-4503-5640-4/18/04.

<https://doi.org/10.1145/3184558.3186908>

the alphabet of r , (3) $\delta(q_0, a) = \{i \mid i \in \text{first}(r) \wedge r[i] = a\}$, $\delta(q, a) = \{i \mid i \in \text{follow}(r, q) \wedge r[i] = a\}$, for $a \in \Sigma \wedge q \in \text{Pos}(r)$, (4) $q_0 = 0$ is the initial state, and (5) F is the set of final states, if $\varepsilon \in L(r)$, then $F = \{q_0\} \cup \text{last}(r)$, otherwise $F = \text{last}(r)$.

The matching process is mainly divided into the following phases: (1) in the first superstep, $v \in V$ is considered to be matched with $q \in \text{first}(r)$; for the remaining supersteps, it is matched forward w.r.t. the receiving message set M_r ; (2) M_s is generated and $\text{sendMsg}(v, v', M_s)$ is invoked at v , in which M_s is sent from v to v' . It can be proved that the complexity of P3RPQ is bounded by $O(|r|^{S+2})$, where S is the number of supersteps.

Algorithm 1: P3RPQ

Input : An RDF graph $T = (V, E, I)$ and an RPQ $Q = (x, r, y)$
Output : The provenance-aware answer set $\llbracket Q \rrbracket_T^p$

```

1 Build Glushkov automaton  $A_Q = (St, \Sigma, \delta, q_0, F)$  based on  $Q$ ;
2  $M_r \leftarrow \emptyset, N_v \leftarrow$  all neighboring vertices for  $v$ ;
3 foreach  $v \in V$  do // compute in each vertex in parallel
4   foreach outgoing edge label  $a$  for  $v$  do // superstep 0
5     if  $\exists q \in \text{first}(r) \wedge r[q] = a$  then  $M_s \leftarrow \{(v, r[q], q)\}$ ;
6     foreach  $v' \in N_v$  do  $\text{sendMsg}(v, v', M_s)$ ;
7   foreach  $\rho' \in M_r$  do // the remaining supersteps
8      $(v_t, r[q], q) \leftarrow$  the last one in  $\rho'$ ;
9     if  $q \in \text{last}(r)$  then
10        $\rho \leftarrow \rho' \cup \{(v, r[q], q_f)\}$ ; //  $\lambda(\rho) \in L(r), q_f \in F$ 
11        $Val(v) \leftarrow Val(v) \cup \{\rho\}$ ; // set of paths
12        $R_{q'} \leftarrow \text{follow}(r, q)$ ; // next states for  $q$ 
13       if  $\exists q' \in R_{q'} \wedge r[q'] = a$  then
14          $M_s \leftarrow M_s \cup \{\rho'\} \cup \{(v, r[q'], q')\}$ ;
15       foreach  $v' \in N_v$  do  $\text{sendMsg}(v, v', M_s)$ ;
16 foreach  $v \in V$  do  $\llbracket Q \rrbracket_T^p \leftarrow \llbracket Q \rrbracket_T^p \cup Val(v)$ ;
17 return  $\llbracket Q \rrbracket_T^p$ ;
```

The optimized algorithm P3RPQ_{opt} with *edge filtering* and *candidate state* only loads the edges that are relevant to Q and precomputes each state $q \in St$ that v can match. Due to the *counting paths* problem, it may result in a large number of intermediate results when querying an RDF graph consisting of some vertices that can incur Cartesian products, just like v_2 in T_1 in Fig.1(a). To this end, we propose another optimized algorithm P3RPQ_{cnt} with *message selection* and *message compression* to decrease the times of matching at vertices and compress the messages to be sent.

3 EXPERIMENTS

The proposed method was implemented in Scala using Spark GraphX deployed on a 10-site cluster. Each of the sites installs a 64-bit CentOS 7.3 operating system and has a 4-core CPU and 16GB memory. We use five experimental datasets LUBM100 (10.9M triples), LUBM200 (27.6M triples), DBpedia (18.2M triples), WatDiv1M (1.1M triples), and WatDiv10M (10.9M triples). An RPQ is complex if it has the closure operators $*$ or $+$, otherwise it is simple. We designed 12 representative RPQs listed in Table. 1.

Fig. 2(a) shows the average query response times of the above 12 queries over LUBM100, LUBM200, and DBpedia. It can be observed that P3RPQ_{opt} outperforms P3RPQ for all queries. We used

Table 1: Regular path queries

Simple RPQs			
$Q_1 = (a/b)/(c/d)$	$Q_2 = a/b/c/d$	$Q_3 = a/b/(c/d/e)$	$Q_4 = (a/b)/(a/c)$
Complex RPQs			
$Q_5 = (a/a)^+$	$Q_6 = (a/b/c)^+$	$Q_7 = a^+$	$Q_8 = a/(a/b/c)^*$
$Q_9 = (a/b)^+/(c/d)^+$		$Q_{10} = (a/(b/c)^*)^+/(d/e)^+$	
$Q_{11} = ((a/b)/(c/d)^*)^+/(e/f)^*$		$Q_{12} = (a/b)^+/(c/d)^+$	

LUBM100 and DBpedia as datasets, with 4 queries (Q_2, Q_4, Q_5 , and Q_{11}) randomly selected from Table. 1 to evaluate the scalability, as shown in Fig. 2(b). Moreover, we generated RDF graphs w.r.t. the data model of WatDiv by constructing some vertices with m incoming and n outgoing edges, and designed the query $Q_c = a_1/\dots/a_{10}$. In Fig. 2(c), we can observe that P3RPQ and P3RPQ_{opt} cannot finish within the time limit (10^4 s), whereas P3RPQ_{cnt} can return the answers in 78.39s and 377.56s, over the RDF graphs consisting of 1 million and 10 million triples, respectively.

Figure 2: Experimental results

4 CONCLUSION

In this paper, we propose a novel method for answering RPQs with provenance-aware semantics over large RDF graphs using the Pregel parallel graph model. We devise four optimization techniques, which can reduce the query time dramatically and partly overcome the counting paths problem. Extensive experiments were conducted on both synthetic and real-world datasets, which have verified the effectiveness and efficiency of our proposed method.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (61572353) and the Natural Science Foundation of Tianjin (17JCYBJC15400).

REFERENCES

- [1] Saumen Dey, Victor Cuevas-Vicentín, Sven Köhler, Eric Gribkoff, and Michael Wang. 2013. On implementing provenance-aware regular path queries with relational query engines. In *Proceedings of EDBT/ICDT*. ACM, 214–223.
- [2] Grzegorz Malewicz, Matthew H Austern, Aart JC Bik, James C Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. 2010. Pregel: a system for large-scale graph processing. In *Proceedings of SIGMOD*. ACM, 135–146.
- [3] Xin Wang, Jun Ling, Junhu Wang, Kewen Wang, and Zhiyong Feng. 2014. Answering provenance-aware regular path queries on rdf graphs using an automata-based algorithm. In *Proceedings of WWW*. ACM, 395–396.