Estimating the Evaluation Cost of Regular Path Queries on Large Graphs

Van-Quyet Nguyen
Dept. of Electronics and Computer Engineering
Chonnam National University
quyetict@utehy.edu.vn

Kyungbaek Kim
Dept. of Electronics and Computer Engineering
Chonnam National University
kyungbaekim@jnu.ac.kr

ABSTRACT

Regular path queries (RPQs) are widely used on a graph whose answer is a set of tuples of nodes connected by paths corresponding to a given regular expression. Traditional approaches for evaluating RPQs are restricted in the explosion of graph size and/or highly complex query (e.g., nested query). Consequently, evaluating an RPQ on a large graph often takes high cost, causing substantial memory spaces and long response time. Recently, cost-based optimizations of RPQs have been proved to be effective when they are applied to large graphs. However, these techniques could not guarantee the minimum evaluation cost all the time. Therefore, estimating the evaluation cost of RPQs is an important topic which opens the way to cost-based graph query processing.

In this paper, we present a novel approach for estimating the evaluation cost of RPQs on large graphs. Our method exploits graph schema to make a so-called USCM (*Unit-Subquery Cost Matrix*), which presents the evaluation cost of the unit-subqueries (i.e. every smallest possible subquery). We propose some cost functions based on USCM to estimate the evaluation cost of an RPQ by decomposing it into a set of unit-subqueries. We also present a case study which applies our proposed idea for parallel RPQs evaluation. Experimental results show that our estimation method obtains high accuracy approximately 87% in average. Moreover, two comparisons with automata-based and rare label based approaches demonstrate that USCM-based approach outperforms traditional ones.

CCS CONCEPTS

• **Computing methodologies** → **Search methodologies**; *Parallel computing methodologies*;

KEYWORDS

Regular Path Queries, RPQ Cost Estimation, Graph Querying

ACM Reference Format:
Van-Quyet Nguyen and Kyungbaek Kim. 2017. Estimating the Evaluation Cost of Regular Path Queries on Large Graphs. In *SoICT '17: Eighth International Symposium on Information and Communication Technology, December 7–8, 2017, Nha Trang City, Viet Nam.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3155133.3155160

SoICT '17, December 7–8, 2017, Nha Trang City, Viet Nam

ACM ISBN 978-1-4503-5328-1/17/12…$15.00
https://doi.org/10.1145/3155133.3155160

1 INTRODUCTION

A regular path query (RPQ) is introduced as a part of a query language for graph databases, which are represented as graphs whose nodes are objects and edge labels specify relationships between them [19]. The answer of an RPQ is a set of tuples of nodes which are connected with edge labels in some ways by the paths specified by a regular language [1, 4–6, 18]. RPQs have been utilized in many applications such as friends recommendations in social networks [14] and detecting signal pathways in protein interaction networks [21]. In such systems, databases could store extremely large graphs in practice (i.e. hundreds of millions nodes and edges on Twitter social network [27], billions nodes/edges on Friendster social network [28]). Hence, evaluating an RPQ on such graphs takes high cost causing substantial memory spaces and long response time. Therefore, estimating the evaluation cost of RPQs opens the way to improve the performance of query evaluation. For instance, to improve response time for evaluating an RPQ, we can split original RPQ into smaller subqueries, evaluate them parallelly and combine partial answers. In this case, estimating the cost of each subquery is one of the key points to choose the split labels which help separating original RPQ in an efficient way.

RPQs evaluation on graph database has been studied intensively in the literature [2, 11, 12, 16, 18, 23, 26]. A common approach is to use automata [11]. However, the drawback of the automata-based approach is that the states of automaton are mapped onto the graph, which could cause long response time due to a large graph. To address this issue, there have been several studies focusing on optimizing the evaluation cost of RPQs. The first technique is rewriting regular path queries [4, 9]. In which, a given regular expression is converted into another one that helps reducing search space by searching only a portion of the data. But, this approach still has a limitation when dealing with rewriting highly complex RPQs (e.g. nested RPQs with modifier recursion).

In recent years, cost-based optimizations of RPQs have been proved to be effective when they are applied to large graphs. In [15] and an extension of that work [16], the authors proposed an approach for answering RPQs using ideas from cost-based query optimization. Their works use a cost-based technique for determining which labels in the graph are considered to be rare. By using rare labels as start-, end-, and way-points during traversal, this approach could decrease the search space. However, the major drawback of this approach is that the algorithm depends on the presence of rare labels and the number of rare labels in the graph and query. In the case of poor rare labels or long queries, this approach still takes a high cost and can reach to the complexity $O(n^2)$, where n is the number of edges of the graph. We will compare our estimation

cost based approach to rare label based approach in case of parallel RPQs evaluation (for more details, see Section 4).

An area, where the efficiency of evaluation RPQs is important, is querying on distributed graphs. A survey about the state of the art of evaluating queries on distributed graphs is presented in [17]. Dan Suciu presented a distributed query evaluation approach on semi-structured data [22]. Their algorithm takes a bounded complexity $O(n^2)$ for the amount of data transfers via the network, where n is the total of cross-edges. Wenfei Fan et al. in [8] proposed efficient algorithms for answering three classes of regular reachability queries on distributed graphs based on a technique named *partial evaluation*. However, it faces a communication bottleneck problem when assembling all distributed partial query results. This problem is addressed in [20, 25]; therein, a large amount of redundant data is detected and removed before assembling at the coordinate site.

Despite many studies have focused on RPQs evaluation, to the best of our knowledge, there has been very little research studying on estimating evaluation cost of RPQs and its effectiveness. Silke et al. in [24] provided functions to estimate the sizes of result sets and the response times to evaluate reachability and path queries. Davoust et al. in [7] presented estimation cost functions to provide strategies for evaluating RPQs on distributed graphs. However, this work mainly focuses on estimating the amount of data to be transferred via the network during evaluating an RPQ. None of these works above provides estimation cost functions relying on operators in RPQs and connectivity of labels. In this paper, we propose a novel approach for estimating the evaluation cost of RPQs on large graphs. Our method exploits graph schema to make a *Unit-Subquery Cost Matrix* (called USCM) which presents the evaluation cost of the unit-subqueries (i.e. every smallest possible subquery). We provide cost functions based on USCM to estimate the evaluation cost of an RPQ by decomposing the original query into a set of unit-subqueries.

Our idea can also be used for variations of the general RPQ problem, such as optimal grouping of RPQs into some groups with balanced evaluation cost, or query planning on graph-based system. Especially, estimation cost plays an important role in parallel query processing. Based on the estimated evaluation cost, we can choose split labels which minimize the graph searching cost of the split subqueries, so we can reduce more response time for evaluating RPQs on large graphs.

Our work makes the following contributions.

- We define a Unit-Subquery Cost Matrix according to graph schema. A unit-subquery can be a part of an RPQ, so we can use USCM to estimate the evaluation cost of RPQ.
- According to USCM, we propose a novel approach for estimating evaluation cost of a given RPQ. Three main operators in an RPQ including concatenation, alternation, and bounded Kleene operator, are considered to estimate the cost. Moreover, we also discuss the estimation of highly complex RPQs.
- We present a case study which shows our idea can be applied for parallel evaluation of RPQs to reduce the searching cost.
- We conduct extensive experiments which show that our estimation method obtains high accuracy approximately 87% in average. Moreover, two comparisons with automata-based and rare label based approaches demonstrate experimentally that our approach outperforms traditional ones in the aspect of parallel RPQs evaluation.

The rest of this paper is organized as follows. In Section 2, we present terms and definitions related to regular path queries. In Section 3, we describe our method of estimating the evaluation cost of RPQs: a Unit-Subquery Cost Matrix (USCM) and estimating the evaluation cost of RPQs by using USCM. We conduct the experimental evaluation using both real-life and synthetic graphs in Section 4. Section 5 concludes with a summary and shows our future work.

2 PRELIMINARIES

2.1 Graph Data and Regular Path Queries

We consider an edge-labeled directed graph $G = (V, E, \Sigma)$, where V is a finite set of nodes, Σ is a finite set of labels, and $E \subseteq V \times \Sigma \times V$ is a finite set of edges. An edge (v, a, u) denotes a directed edge from node v to u labeled with $a \in \Sigma$.

A path ρ between nodes v_0 and v_k in G is a sequence

$$\rho = v_0 a_0 v_1 a_1 v_2 ... v_{k-1} a_{k-1} v_k$$

such that each (v_i, a_i, v_{i+1}), for $0 \leq i < k$, is an edge. The sequence of labels of a path ρ, denoted L(ρ), is the string $a_0 a_1 ... a_{k-1} \in \Sigma^*$, where Σ^* is a set of all possible strings over the set of labels Σ. We also define the *empty* path as (v, ε, v) for each $v \in V$; the label of such a path is the empty string ϵ.

An RPQ with a regular expression R is a query of the form $Q(R) = v \xrightarrow{L(R)} u$, where $L(R) \in \Sigma^*$ is a regular language. So, a path ρ satisfies $Q(R)$ on the graph G iff L(ρ) $\in$ L(R), then ρ is an answer of $Q(R)$. Here, R is a regular expression over Σ,

$$R = \epsilon \mid a \mid R \circ R \mid R \cup R \mid R^{[i,j]},$$

where ϵ is an empty value; a is a label in Σ; $R \circ R$, $R \cup R$, and $R^{[i,j]}$ denote concatenation, alternation, and Kleene operator with bounded recursion $[i, j]$, where $i < j$ and $i, j \in \mathbb{N}$, respectively.

Note that, in the syntax of regular expression we use a bounded Kleene operator which bounds recursion with $[i, j]$ instead of an unbounded Kleene operator (e.g., *, +). This is motivated by the following three observations. Firstly, the bounded Kleene operator is supported by graph query languages in practice, such as Neo4j's Cypher[1]. Secondly, bounded recursion on regular path queries evaluation has been studied in the literature [10]. Finally, it is not difficult to find that for any graph G, there exists a natural number n such that for every RPQ, $Q(R)$, there is a possible case that $R^* = R^{[0,n]}$. Similarly, other modifiers (+ and ?) can be represented as follows: $R+ = R^{[1,n]}$ and $R? = R^{[0,1]}$. While evaluating an RPQ with unbounded recursion on a large graph is a non-trivial task.

Example 1: Figure 1(a) illustrates a graph G of social network, where each node denotes a person with his/her name and each edge represents the relationship between two people, in which an edge is labeled by an element in a set of labels

$$\Sigma = \{supervisor, colleague, friend, married, knows\}.$$

In this graph, a regular path query $Q(R)$ with

[1] http://neo4j.com/docs/developer-manual/current/cypher/syntax/patterns/

- $R = supervisor \circ (colleague \cup friend)$ finds all paths between any $supervisor(s)$ and $colleagues$ or $friends$ of his/her employees. In this case, the result includes three paths as follows.

$$\text{Bill} \xrightarrow{supervisor} \text{Dan} \xrightarrow{colleague} \text{Jun}$$
$$\text{Bill} \xrightarrow{supervisor} \text{Dan} \xrightarrow{friend} \text{Tea}$$
$$\text{Job} \xrightarrow{supervisor} \text{Ben} \xrightarrow{friend} \text{Son}$$

- $R = supervisor \circ friend^{[1,2]} \circ married$ finds all paths from any $supervisor(s)$ to the people who married with $friend$ or $friend$ of $friend$ of his/her employees. In this case, the result includes two paths as follows.

$$\text{Bill} \xrightarrow{supervisor} \text{Dan} \xrightarrow{friend} \text{Tea} \xrightarrow{friend} \text{Ken} \xrightarrow{married} \text{Lee}$$
$$\text{Job} \xrightarrow{supervisor} \text{Ben} \xrightarrow{friend} \text{Son} \xrightarrow{married} \text{Ann}$$

2.2 Evaluation of an RPQ

Informally, the evaluation of an RPQ, $Q(R)$, is to find all paths between pairs of nodes in a graph G, such that the path from one node to the other matches a given regular expression R. There are two types of RPQs which have been considered in the literature. In which, *multi-source queries* start a search at every node in the graph, and *single-source queries* start a search at a single given *start node*. In this paper, we consider estimating the evaluation cost of the multi-source queries. For better understanding about the estimation cost, we will describe the basic of RPQ evaluation using automata-based technique.

Query Automaton. To process an RPQ, a regular expression can be converted into an automaton then used to matching paths. We use a deterministic finite automata (DFA) to represent query where the definition DFA as in [13]. That is, an RPQ, $Q(R)$, is represented by an automaton A_R which is a 5-tuple as the following:

$$A_R = \{Q, \Sigma, \mu, q_0, F\},$$

where Q is a finite set of states, Σ is a finite set of labels (or symbols), μ is the transition function, that is, μ: $Q \times \Sigma \rightarrow Q$, q_0 is an initial (or start) state and $q_0 \in Q$, F is a set of terminal states and $F \subset Q$.

Query Evaluation. A well-known method for query evaluation [13] based on automata consists of the steps as follows:

- Build a finite automaton A_R associated with the regular expression R. The initial state of A_R is q_0, the accepted states are $\{q_t\}$, where $q_t \in F$.
- Consider graph G as an automaton A_G with nodes as states, edges as transitions and compute the cross-product of the automata $A_P = A_R \times A_G$.
- Apply any graph search algorithm such as breadth-first or depth-first to find all pairs of nodes related by the regular path: search A_P from all initial states (q_0, v_i) to find all reachable accepted states (q_t, v_j). All pairs of nodes (v_i, v_j) are answers to the RPQ.

The evaluation cost of the algorithm above consists of the cost of building the query automaton, plus the cost of building and searching the product automaton. In practice, the searching cost is the determinant of evaluation cost, especially in the case of handling large graphs. We define the evaluation cost of an RPQ as the number of traversed edges for searching paths corresponding to the RPQ.

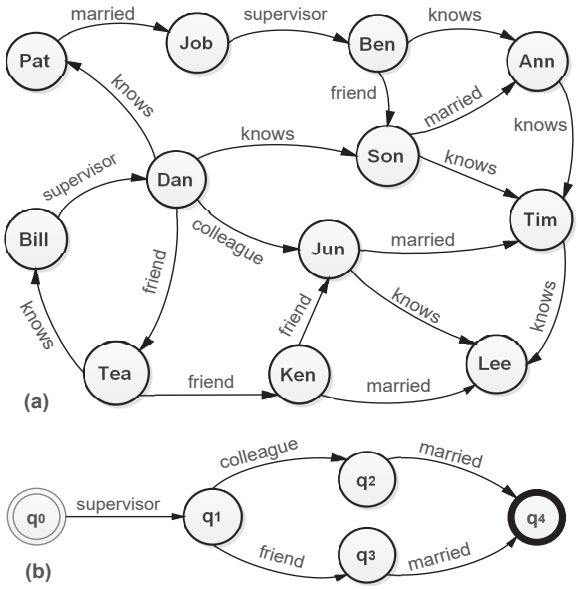

Figure 1: (a) An example of a social network as a directed edge-labeled graph; (b) A simple regular path query as an automaton.

Example 2: Suppose that we have a graph G as described in Example 1. We consider to the evaluation of an RPQ, $Q(R)$, with

$$R = supervisor \circ (colleague \cup friend) \circ married.$$

First, we convert $Q(R)$ into finite automata A_R as shown in Figure 1(b). Next, we look up the *start nodes* from graph G. To be able to efficiently gather *start nodes* from the graph, we assume that an index of the edge label also encodes their incoming and outgoing nodes. This index gives us the list of *start nodes* of the label in the graph. In this case, we have two *start nodes* $\{Bill, Job\}$. Starting from node *Bill*, there is a node, *Dan*, matching with a state in A_R (q_1). From *Dan*, we find the nodes for next searching. Here, four edges are traversed and two nodes, $\{Jun, Tea\}$, are considered as the next starting nodes. Then, the searching process continues from these nodes, there are four more edges traversed and only one node, Tim, is matched with the final state q_4. Thus, for searching from *Bill*, the searching cost is nine. Similarly, starting from *Job*, we can found only one path satisfying $Q(R)$,

$$\text{Job} \xrightarrow{supervisor} \text{Ben} \xrightarrow{friend} \text{Son} \xrightarrow{married} \text{Ann},$$

with the searching cost is five. As the result, we have the evaluation cost of Q(R) in this case is fourteen. We are going to compare this the result with our estimation cost in the next section.

3 ESTIMATING THE EVALUATION COST OF RPQS

In this section, we present a novel approach for estimating the evaluation cost of RPQs. We first define a Unit-Subquery Cost Matrix (USCM). We then present how to estimate the evaluation cost by using USCM.

Table 1: An example of Unit-Subquery Cost Matrix

Label:Count	supervisor	colleague	friend	married	knows	Total
supervisor:2	0	1	2	0	3	6
colleague:1	0	0	0	1	1	2
friend:4	0	0	2	3	3	8
married:4	1	0	0	0	2	3
knows:8	1	0	0	2	4	7

3.1 Unit-Subquery Cost Matrix (USCM)

Intuitively, an RPQ is composed of multiple small subqueries with a few operators such as concatenation, alternation, and bounded Kleene. Then, we can define a *unit-subquery* as the smallest subquery which is concatenated by two labels from Σ; the *start label* and the *end label*. For example, in the graph G of Figure 1, a subquery $Q(supervisor \circ colleague)$ is a unit-subquery, where *supervisor* is the *start label* and *colleague* is the *end label*. In practice, any query which is defined as in Section 2.2 can be split into multiple unit-subqueries even if the query which contains the Kleene operators with bounded recursion. For example, the subquery $Q((colleague \cup friend) \circ married)$ can be split into two unit-subqueries $Q(colleague \circ married)$ and $Q(friend \circ married)$; meanwhile, the query $Q(supervisor \circ (colleague \circ friend)^{[1,2]})$ is composed of six unit-subqueries including $Q(supervisor \circ colleague)$, $Q(supervisor \circ friend)$, $Q(colleague \circ friend)$, $Q(colleague \circ colleague)$, $Q(friend \circ colleague)$, and $Q(friend \circ friend)$.

The cost of a unit-subquery is defined as the number of edges with the *end label*, which is connected to the edges with the *start label*. For example, in the graph G of Figure 1, the cost of a subquery $Q(supervisor \circ colleague)$ is one because there is only one edge (Dan, *colleague*, Jun) labeled with *colleague*, which is connected to uni-directional edges labeled with *supervisor*.

With the definition of the cost of unit-subqueries, we can generate a *Unit-Subquery Cost Matrix (USCM)* which represents the cost of all possible unit-subqueries from Σ. An example of USCM is shown in Table 1. The size of USCM is n by $n+1$ where n is the number of distinct labels in Σ. A cell (i, j) of USCM, except the last column where j is $n+1$, represents *the cost of a unit-subquery*, $Q(a_i a_j)$, whose *start label* is $a_i \in \Sigma$ and *end label* is a_j. For clarity of presentation, we drop the explicit use of the concatenation, and we use the symbol | for alternation operator in terms and equations, only keep the symbols $\circ$ and $\cup$ in the examples (from now on). In the last column, a cell (i, j) represents the cost of a unit-subquery $Q(a_i\_)$, that is, the summation of the costs of unit-subqueries whose *start label* is a_i. Additionally, USCM contains the number of edges with a given label (Count) like the first column of USCM.

Because the contents of USCM are constant in a given graph G, we can prepare USCM for just one time unless the graph G is updated. The complexity of building USCM is $n \times (|E| + |E|)$.

3.2 USCM-Based Estimating of Evaluation Cost

In this section, we propose cost functions for estimating the evaluation cost of an RPQ, $Q(R)$, in three main cases of regular expression R: (1) a simple regular expression with concatenation operator; (2) a regular expression with alternation operator; and (3) a regular expression with bounded Kleene operator. We also discuss estimating the evaluation cost of highly complex RPQs.

3.2.1 An RPQ with concatenation. In our approach, the evaluation cost of an RPQ is estimated by splitting the original RPQ into multiple unit-subqueries and gathering the cost of each successive unit-subqueries.

Let us assume that there is an RPQ, $Q(R)$, where $R = a_0 a_1 .. a_n$ as a string that is concatenated by $(n+1)$ labels $a_i \in \Sigma$. Then, $Q(R)$ can be split into $Q(a_0 a_1)$, $Q(a_1 a_2)$, .., $Q(a_{n-1} a_n)$, and the evaluation cost for $Q(R)$ is defined as summation of cost of each successive unit-subquery like Equation 1.

$$C_{Q(R)} = \sum_{i=0}^{n-1} C_{Q(a_i a_{i+1})} = \sum_{i=0}^{n-1} C_i \tag{1}$$

For C_0, the evaluation starts from the edge labeled with a_0 and tries to find the path to a_1. Accordingly, C_0 composed of the cost of finding the next search nodes and the cost of searching a_1. That is, $C_0 = \delta(a_0) + \xi(a_0)$, where $\delta(a_i)$ is the number of edges given label a_i which is the *Count* value for the first column of USCM and $\xi(a_i)$ is the cost of $Q(a_i\_)$ which is the value of the last column of USCM.

For C_i, where $i > 0$, we do not consider the searching cost for finding edges with label a_i, because this cost is already considered in the previous step C_{i-1}. So, for C_i, we only consider the searching cost for finding edges with label a_{i+1}. However, this cost is affected by the number of search nodes which are found in the previous step(s). To consider this effect, we can calculate the probability of how many edges labeled with a_i related to the unit subquery $Q(a_{i-1} a_i)$ are found among the all the edges labeled with a_i, and apply to the searching cost to edges labeled with a_{i+1} from edges labeled with a_i. That is, C_i can be represented like Equation 2, where $\mu(a_{i-1}, a_i)$ is the cost of unit-subquery $Q(a_{i-1} a_i)$, which is the first value of each cell of USCM.

$$C_i = \frac{\mu(a_0, a_1)}{\delta(a_1)} \times \cdots \times \frac{\mu(a_{i-1}, a_i)}{\delta(a_i)} \times \xi(a_i) \tag{2}$$

Example 3: Suppose that we have a graph G as described in Example 1. An actual situation as the following: for a marketing strategy in a company, the board of directors wants to introduce products through all paths from the supervisors to people who are married to friends of the employees in that company. A regular expression R which represents that situation is $R = supervisor \circ friend \circ married$. Here, we do not focus on evaluating this query

but estimate the evaluation cost of $Q(R)$. In this case, the evaluation cost can be estimated as follows.

$$C_0 = C_{Q(supervisor \circ friend)} = \delta(supervisor) + \xi(supervisor) = 2 + 6 = 8$$

$$C_1 = C_{Q(friend \circ married)} = \frac{\mu(supervisor, friend)}{\delta(friend)} \times \xi(friend) = 2/4 \times 8 = 4.$$

Thus, the total estimated cost is twelve. It is equal to the true cost of evaluating $Q(R)$ in the graph G by using automata-based approach.

3.2.2 An RPQ with alternation operator. We assume that an RPQ, $Q(R)$, is defined by a regular expression,

$$R = a_0 \ldots a_{i-1}(a_i|a_{i+1})a_{i+2} \ldots a_n,$$

where $a_i \in \Sigma$. Herein, R has an alternation operator between a_i and a_{i+1}. In this case, the original $Q(R)$ can be split into three subqueries $Q(a_0 \ldots a_{i-1}), Q(a_{i-1}(a_i|a_{i+1})a_{i+2})$, and $Q(a_{i+2} \ldots a_n)$. For the subqueries $Q(a_0 \ldots a_{i-1})$ and $Q(a_{i+2} \ldots a_n)$, we can estimate their cost by using our method in Section 3.2.1.

For evaluating $Q(a_{i-1}(a_i|a_{i+1})a_{i+2})$, we need to consider two different steps: $Q(a_{i-1}(a_i|a_{i+1}))$ and $Q((a_i|a_{i+1})a_{i+2})$. In the first step, $Q(a_{i-1}(a_i|a_{i+1}))$ can be considered by two subqueries $Q(a_{i-1}a_i)$ and $Q(a_{i-1}a_{i+1})$. Here, evaluating both of these subqueries starts from edges labeled with a_{i-1}, and during a single evaluation time we can traverse the edges labeled with a_i as well as a_{i+1}. So, the cost of $Q(a_{i-1}(a_i|a_{i+1}))$ can be estimated by the cost of either $Q(a_{i-1}a_i)$ or $Q(a_{i-1}a_{i+1})$. On the other hands, $Q((a_i|a_{i+1})a_{i+2})$ can be decomposed into $Q(a_ia_{i+2})$ and $Q(a_{i+1}a_{i+2})$, and the evaluation process of these subqueries is different to each other. So, C_A, the estimated cost of $Q(a_{i-1}(a_i|a_{i+1})a_{i+2})$, can be estimated by the summation of the costs of $Q(a_{i-1}a_i), Q(a_ia_{i+2})$, and $Q(a_{i+1}a_{i+2})$ like Equation 3.

$$C_A = C_{Q(a_{i-1}a_i)} + C_{Q(a_ia_{i+2})} + C_{Q(a_{i+1}a_{i+2})} \tag{3}$$

Equation 3 can be represented as an explicit formula by two cases as the following:

- i = 1

$$C_A = \delta(a_0) + \xi(a_0) + \frac{\mu(a_0, a_1)}{\delta(a_1)}\xi(a_1) + \frac{\mu(a_0, a_2)}{\delta(a_2)}\xi(a_2) \tag{4}$$

- i > 1

$$C_A = \frac{\mu(a_0, a_1)}{\delta(a_1)} \times \cdots \times \frac{\mu(a_{i-2}, a_{i-1})}{\delta(a_{i-1})} \times \left(\xi(a_{i-1}) + \frac{\mu(a_{i-1}, a_i)}{\delta(a_i)}\xi(a_i) + \frac{\mu(a_{i-1}, a_{i+1})}{\delta(a_{i+1})}\xi(a_{i+1})\right) \tag{5}$$

Example 4: We illustrate our idea of estimating evaluation cost in case of query has alternation operator by an example. In which, the regular expression $R = supervisor \circ (colleague \cup friend) \circ married$. By using Equation 4, the evaluation cost of $Q(R)$ can be estimated as follows.

$$C_{Q(R)} = C_A = \delta(supervisor) + \xi(supervisor) + \frac{\mu(supervisor, colleague)}{\delta(colleague)}\xi(colleague) + \frac{\mu(supervisor, friend)}{\delta(friend)}\xi(friend) = 2 + 6 + (1/1) \times 2 + (2/4) \times 8 = 14$$

In this case, the total estimated cost is fourteen. It equals the true cost of evaluating $Q(R)$ in the graph G by using automata-based approach as we mentioned in Example 2.

3.2.3 An RPQ with bounded Kleene operator. Let us assume that there is an RPQ, $Q(R)$, where

$$R = a_0a_1 \ldots a_{k-1}a_k^{[i,j]}a_{k+1} \ldots a_n$$

with a bounded Kleene operator. To estimate the cost of $Q(R)$, we can split this query into three subqueries including

$$Q(a_0 \ldots a_{k-1}), Q(a_{k-1}a_k^{[i,j]}a_{k+1}), \text{ and } Q(a_{k+1} \ldots a_n),$$

then the evaluation cost for $Q(R)$ is defined as summation of cost of each subquery. We can estimate the costs of the subqueries $Q(a_0 \ldots a_{k-1})$ and $Q(a_{k+1} \ldots a_n)$ by using the proposed method described in Section 3.2.1. The subquery $Q(a_{k-1}a_k^{[i,j]}a_{k+1})$ is composed of the unit-subqueries: a $Q(a_{k-1}a_k)$, $(j-1)$ times $Q(a_ka_k)$, and a $Q(a_ka_{k+1})$. So, the estimated cost of $Q(a_{k-1}a_k^{[i,j]}a_{k+1})$, C_K, is defined as shown in Equation 6.

$$C_K = \frac{\mu(a_0, a_1)}{\delta(a_1)} \times \cdots \times \frac{\mu(a_{k-1}, a_k)}{\delta(a_k)}\left(1 + \frac{\mu(a_k, a_k)}{\delta(a_k)} + \frac{\mu(a_k, a_k)}{\delta(a_k)} \times \frac{\mu(a_k, a_k)}{\delta(a_k)} + \cdots\right)\xi(a_k) \tag{6}$$

Let $\omega = \frac{\mu(a_k, a_k)}{\delta(a_k)}$, $\forall \omega \neq 1$, the estimated cost, C_K, can be formalized as follows.

$$C_K = \frac{\mu(a_0, a_1)}{\delta(a_1)} \times \cdots \times \frac{\mu(a_{k-1}, a_k)}{\delta(a_k)} \times \frac{\omega^j - 1}{\omega - 1}\xi(a_k) \tag{7}$$

Equation 7 shows that the cost of evaluating an RPQ is not depended on lower bound (i) of Kleene operator, but depends on upper bound (j) of Kleene operator. That is, the high value of j will take a high evaluation cost of $Q(R)$.

Example 5: In this example, we will estimate the evaluation cost of RPQ, $Q(R)$, with $R = supervisor \circ friend^{[1,3]} \circ married$. In this case, $C_{Q(R)} = C_0 + C_K$, with $C_0 = \delta(supervisor) + \xi(supervisor) = 2 + 6 = 8$; and $C_K = \frac{\mu(suppervisor, friend)}{\delta(friend)} \times \frac{\omega^3 - 1}{\omega - 1}\xi(friend)$, where $\omega = \frac{\mu(friend, friend)}{\delta(friend)} = 2/4 = 0.5$. So, $C_K = 2/4 \times \left(0.5^3 - 1\right)/(0.5 - 1)\Big) \times 8 = 7$. Finally, we have $C_{Q(R)} = 8 + 7 = 15$. It is close to true cost, *sixteen*, of evaluating $Q(R)$ on graph G.

3.2.4 Estimating Highly Complex RPQs. Our approach is not only effective with simple RPQs but also highly complex RPQs such as the query with a regular expression of the form $R \circ S^{[i,j]} \circ T$ or $R \circ \left(S^{[i,j]} \circ T\right)^{[x,y]} \circ U$, where R, S, T, and U are regular expressions, and i, j, x, and y are natural numbers, in which $i < j$ and $x < y$. To

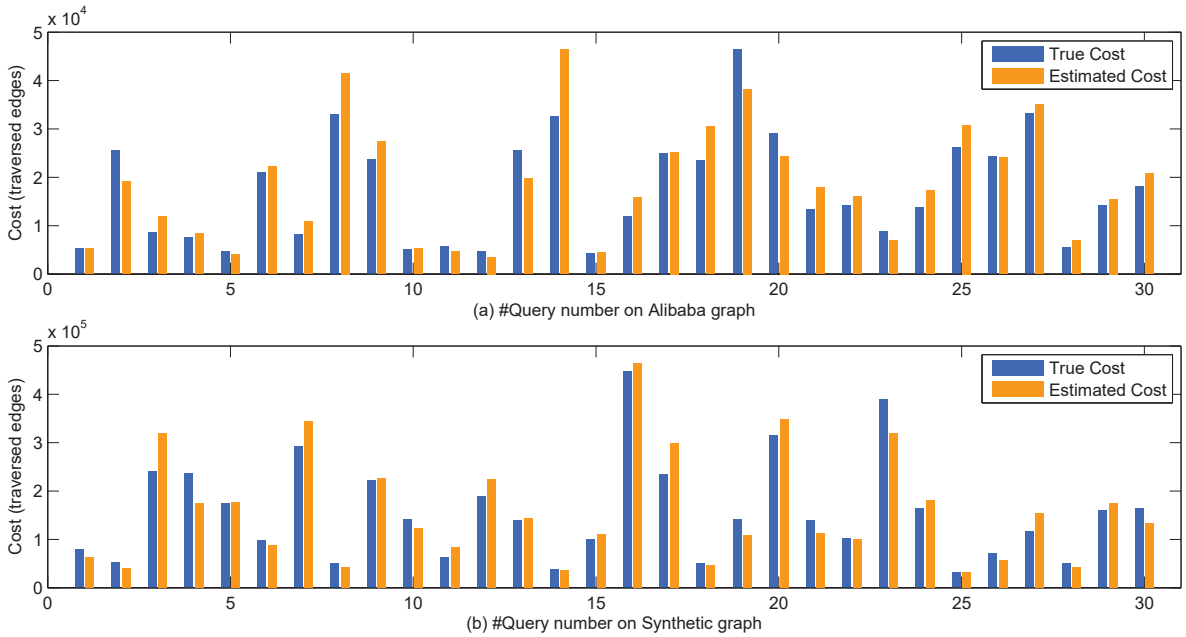

Figure 2: Comparison of the true cost and the estimated cost.

estimate the cost of a complex RPQ, we first decompose such query into the queries containing at least one of three cases of operators as described in Section 3.2. We then use USCM to estimate the cost for decomposed queries. For example, an RPQ, Q(R), with $R = supervisor \circ (colleague \cup friend)^{[1,2]} \circ married$ can be decomposed into two queries:
(1) $supervisor \circ colleague \circ (colleague \cup friend) \circ married$ and
(2) $supervisor \circ friend \circ (colleague \cup friend) \circ married$.
It is not difficult to estimate the evaluation cost of these queries by using our proposed as presented above.

4 EXPERIMENTAL EVALUATION

To evaluate the effectiveness of our proposed approach, we conducted two main experiments: the first one is to compare our estimated cost with the true cost which is the number of traversed edges during evaluation RPQs by using automata-based approach, and the other one is to compare our USCM-based approach with the automata-based approach (AUT) [11] and the threshold-rare label based approach (TRL) [16] in the aspect of the response time of parallel RPQs evaluation on large graphs.

4.1 Evaluation Settings

Environments. Our experiments were conducted on a personal computer which has 3.5 GHz Intel Core i3, 4 CPU cores, and 8.0GB of RAM. All algorithms are implemented in Java.

Data and Queries set. We adopted a real graph from a research on biology (called Alibaba) and the synthetic graphs for the evaluation.

We used Alibaba graph and the queries set given by previous research [16]. The graph is a network of protein-protein interactions which is used regularly in biology systems, for instance, to discover protein functions and pathways in biological processes [29]. This graph has 52.050 nodes, 340.775 edges, and 649 labels. We analyzed 10.000 queries in the queries set and found the following properties. The queries set has around 87% proportion of having simple RPQs, 3% proportion of having nested RPQs without recursive modifiers, and 10% proportion of having nested RPQs with recursive modifiers. For setting the queries set to be the same between our approach and the others, we replaced the modifiers (*, +, ?) by a fixed bounded recursion from 1 to 5.

We used Gephi [3] to create the synthetic graphs with varying number of nodes and number of edges (for more details, see Table 2 and Table 3). We used 15 distinct labels to annotate edges for these graphs. The occurrence of labels follows the Zipfian distribution. Then, we generated randomly 1,000 RPQs with various lengths between 6 and 12. This queries set has about 5% proportion of having the alternation and 30% proportion of having the bounded Kleene operator with bounded recursion in the fixed range from 1 to 5.

Algorithms. To evaluate the accuracy of our estimation method, we reimplemented AUT approach to measure the true cost, and measure the estimated cost by implementing our proposed method. For each query in queries set, the closeness between the estimated cost, e_i, and the true cost, t_i, is calculated by the fraction $\frac{e_i}{t_i}$ in the case of $e_i \leq t_i$, otherwise it is $\frac{t_i}{e_i}$.

In order to show the efficiency of our proposed method, we implemented a case study which the estimated cost is used to split original RPQs into smaller subqueries in an efficient way, then we evaluated them in parallel and combined partial answers. In

Table 2: Accuracy evaluation with varying graph sizes

|V|	|E|	Average Degree	Accuracy
2,000	38,142	19	87.62
4,000	76,860	19	88.43
8,000	152,245	19	89.02
16,000	306,806	19	88.70
32,000	615,047	19	89.18

Table 3: Accuracy evaluation with varying average degrees

|V|	|E|	Average Degree	Accuracy
16,000	63,540	4	75.12
16,000	128,297	8	81.52
16,000	255,578	16	88.00
16,000	513,225	32	89.06
16,000	1,024,183	64	88.45

general, the response time of parallel query processing consists of three major metrics: query splitting time, searching time, and combination time of partial answers. Therein, the searching time takes most of the response time in case of evaluation of RPQs on large graphs. Therefore, it is more relevant here to focus on reducing the searching time. In practice, the searching time depends on the evaluation of subquery which has the highest cost. Thus, we can improve the performance evaluation of an RPQ by spitting it into a few subqueries satisfying the evaluation cost of the RPQ is minimal. To do this, our proposed idea can be used as the following.

Step 1: Find all N possible sets of subqueries, $S = \{S_1, S_2, .., S_N\}$, for a given RPQ. For example, we can find three sets of subqueries for an RPQ, $Q(R)$, with $R = knows \circ supervisor \circ colleague \circ married$ as follows.

$S_1 = \{knows \circ supervisor; supervisor \circ colleague \circ married\}$

$S_2 = \{knows \circ supervisor \circ colleague; colleague \circ married\}$

$S_3 = \{knows \circ supervisor; supervisor \circ colleague; colleague \circ married\}$

Step 2: For each set of subqueries S_i, $1 \leq i \leq N$, estimate the evaluation cost of S_i, which is defined as maximum of evaluation cost of the subqueries belong to S_i.

Step 3: Compare the evaluation cost of all sets of subqueries in S to find out the set S_i which has the minimum evaluation cost.

Note that, in *Step 1* above, we find all of the possible combination of sequenced labels, and it takes polynomial time. Here, we consider only the labels as the split labels if it is not at the position of the labels with bounded Kleene operator or inside a bracket of an alternation operator. After finding the set of subqueries in *Step 3*, each subquery is evaluated on different CPU in parallel by using automata-based approach, and the results are gathered for the answer of the original RPQ.

We also reimplemented AUT approach and the threshold rare label based approach [16] to compare their response time with our USCM-based approach.

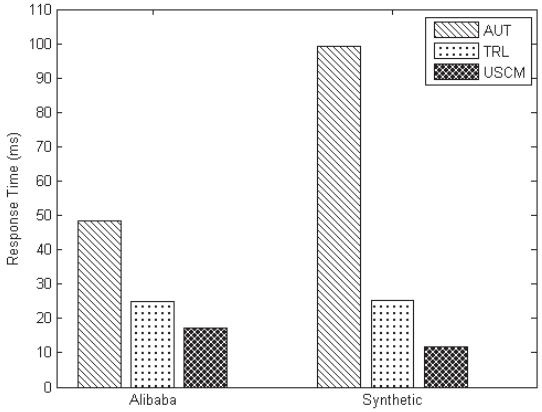

Figure 3: Comparing the response time of parallel RPQs evaluation on large graphs.

4.2 Experimental Results

Exp-1: Accuracy of our estimation method

In order to evaluate the accuracy of our estimation method, we first used Alibaba graph and a synthetic graph with 16,000 nodes and 306,806 edges, and the queries sets are described above. The results show that our estimation method obtained high accuracy: around 85% in the case of Alibaba graph and 89% in case of the synthetic graph. As an example, Figure 2 illustrated a comparison of the estimated cost and the true cost of 30 random queries on each dataset. We observed that the estimated cost is close to the true cost, excepts some queries with high cost caused by the bounded Kleene operators.

Next, we evaluate the accuracy with varied graph size (|V| + |E|). We generated synthetic graphs by varying the size and average degree of the graphs. In directed graphs, the average degree is defined by the fraction $\frac{|E|}{|V|}$. In the first case, we scale graph size from around 40K to 640K nodes and edges, but we keep the same average degree for the graphs. As the results shown in Table 2, our estimation method obtained high accuracy at most 89% for all varied size of the graphs. In another case, we generated five synthetic graphs by fixing the number of nodes |V| = 16K and varying the number of edges |E| from 64K to 1.0M. Consequently, the average degree of the graphs is varied from 4 to 64. We observed that the estimation accuracy for the graphs having average degree greater or equals than 16, which is mostly around 89%, are higher than those in the cases of average degree of 4 and 8 (75.12% and 81.52%, respectively). The results are reasonable because the higher average degree of the graph is, the higher probability of a label connected to anyone else, which helps to increase the estimation accuracy of our method, is.

Exp-2: Efficiency of USCM-based parallel RPQs evaluation

We implemented our algorithm for parallel RPQs evaluation as described in Section 4.1. To ensure the parallel evaluation of split subqueries, the split subqueries are evaluated on different CPU and the results are gathered for the answer of an RPQ. To measure the response time, we get the timestamp difference between issuing an RPQ and getting the answer of an RPQ. That is, the response time

of USCM-based approach includes the time for splitting the RPQ and combining partial answers. Figure 3 illustrated the average response times of three different approaches. We observed that our USCM-based parallel RPQs evaluation outperforms AUT. That is, using estimated cost of RPQ is necessary to reduce the evaluation cost of RPQs in parallel. We also observed that in the case of Alibaba graph, our approach reduced the average response time around 45% comparing to TRL; meanwhile, in the case of the synthetic graph, the reduction is 120%.

Summary. From the experimental results, we find the following. (1) Our estimation method obtains high accuracy and scales well with the size of graphs. (2) The estimation accuracy on the graphs with a high average degree (e.g., greater or equals than 16) is higher than smaller one. (3) Our proposed method is efficient when it is applied to parallel RPQs evaluation on large graphs.

5 CONCLUSIONS AND FUTURE WORK

We proposed a novel approach of estimating the evaluation cost of RPQs on large graphs. By exploiting graph schema and characters of regular path queries, we defined an Unit-Subquery Cost Matrix (USCM) which consists of all possible unit-subqueries of every RPQs. According to USCM, we formalized estimating the evaluation cost of a given RPQ. Our approach is not only effective with simple RPQs but also highly complex RPQs. Moreover, we proved the efficiency of our approach through a case study which the estimated cost is used to split an original RPQ into smaller subqueries by an efficient way for parallel evaluation. Experimental results illustrated our method can estimate evaluation cost for RPQs with high accuracy approximately 87% in average, and USCM-based approach outperforms traditional ones in the aspect of parallel RPQs evaluation.

We envision several directions of our work, one of them is extending the estimation of the evaluation cost for highly complex RPQs with respect to unbounded recursion. Beside that, motivated by the absence of benchmarks devoted to RPQs, we want to develop a benchmark for estimating cost as well as evaluating RPQs. Moreover, we will focus on how to apply this approach for solving RPQ problems in various fields such as transportation analysis under disaster situations and social network analysis for recommendation systems.

ACKNOWLEDGMENTS

This research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2017-2016-0-00314) supervised by the IITP (Institute for Information & communications Technology Promotion). This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (NRF-2017R1A2B4012559).

REFERENCES

[1] Pablo Barceló, Leonid Libkin, Anthony W Lin, and Peter T Wood. 2012. Expressive languages for path queries over graph-structured data. *ACM Transactions on Database Systems (TODS)* 37, 4 (2012), 31.

[2] Pablo Barceló Baeza. 2013. Querying graph databases. In *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGAI symposium on Principles of database systems*. ACM, 175–188.

[3] Mathieu Bastian, Sebastien Heymann, Mathieu Jacomy, et al. 2009. Gephi: an open source software for exploring and manipulating networks. *ICWSM* 8 (2009), 361–362.

[4] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y Vardi. 1999. Rewriting of regular expressions and regular path queries. In *Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 194–204.

[5] Mariano P Consens and Alberto O Mendelzon. 1990. GraphLog: a visual formalism for real life recursion. In *Proceedings of the ninth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*. ACM, 404–416.

[6] Isabel F Cruz, Alberto O Mendelzon, and Peter T Wood. 1987. A graphical query language supporting recursion. In *ACM SIGMOD Record*, Vol. 16. ACM, 323–330.

[7] Alan Davoust and Babak Esfandiari. 2016. Processing Regular Path Queries on Arbitrarily Distributed Data. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. Springer, 844–861.

[8] Wenfei Fan, Xin Wang, and Yinghui Wu. 2012. Performance guarantees for distributed reachability queries. *Proceedings of the VLDB Endowment* 5, 11 (2012), 1304–1316.

[9] Mary Fernandez and Dan Suciu. 1998. Optimizing regular path expressions using graph schemas. In *Data Engineering, 1998. Proceedings., 14th International Conference on*. IEEE, 14–23.

[10] George HL Fletcher, Jeroen Peters, and Alexandra Poulovassilis. 2016. Efficient regular path query evaluation using path indexes. (2016).

[11] Roy Goldman and Jennifer Widom. 1997. DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases. In *VLDB'97, Proceedings of 23rd International Conference on Very Large Data Bases, August 25-29, 1997, Athens, Greece*. 436–445. http://www.vldb.org/conf/1997/P436.PDF

[12] Gösta Grahne and Alex Thomo. 2000. An optimization technique for answering regular path queries. In *WebDB (Selected Papers)*. Springer, 215–225.

[13] John E Hopcroft, Rajeev Motwani, and Jeffrey D Ullman. 2006. Automata theory, languages, and computation. *International Edition* 24 (2006).

[14] Ioannis Konstas, Vassilios Stathopoulos, and Joemon M Jose. 2009. On social networks and collaborative recommendation. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. ACM, 195–202.

[15] André Koschmieder. 2010. Cost-Based Optimization of Regular Path Queries on Large Graphs. *Grundlagen von Datenbanken* 581 (2010).

[16] André Koschmieder and Ulf Leser. 2012. Regular path queries on large graphs. In *Scientific and Statistical Database Management*. Springer, 177–194.

[17] Donald Kossmann. 2000. The state of the art in distributed query processing. *ACM Computing Surveys (CSUR)* 32, 4 (2000), 422–469.

[18] Leonid Libkin and Domagoj Vrgoč. 2012. Regular path queries on graphs with data. In *Proceedings of the 15th International Conference on Database Theory*. ACM, 74–85.

[19] Alberto O Mendelzon and Peter T Wood. 1995. Finding regular simple paths in graph databases. *SIAM J. Comput.* 24, 6 (1995), 1235–1258.

[20] Quyet Nguyen-Van, Le-Duc Tung, and Zhenjiang Hu. 2013. Minimizing data transfers for regular reachability queries on distributed graphs. In *Proceedings of the Fourth Symposium on Information and Communication Technology*. ACM, 325–334.

[21] Jacob Scott, Trey Ideker, Richard M Karp, and Roded Sharan. 2006. Efficient algorithms for detecting signaling pathways in protein interaction networks. *Journal of Computational Biology* 13, 2 (2006), 133–144.

[22] Dan Suciu. 2002. Distributed query evaluation on semistructured data. *ACM Transactions on Database Systems (TODS)* 27, 1 (2002), 1–62.

[23] Silke Trißl. 2007. Cost-based optimization of graph queries. In *Proceedings of the SIGMOD/PODS PhD Workshop on Innovative Database Research (IDAR)*.

[24] Silke Trißl and Ulf Leser. 2010. Estimating Result Size and Execution Times for Graph Queries.. In *ADBIS (Local Proceedings)*. 11–20.

[25] Le-Duc Tung, Quyet Nguyen-Van, and Zhenjiang Hu. 2013. Efficient query evaluation on distributed graphs with Hadoop environment. In *Proceedings of the Fourth Symposium on Information and Communication Technology*. ACM, 311–319.

[26] Nikolay Yakovets, Parke Godfrey, and Jarek Gryz. 2016. Query planning for evaluating SPARQL property paths. In *Proceedings of the 2016 International Conference on Management of Data*. ACM, 1875–1889.

[27] Jaewon Yang and Jure Leskovec. 2011. Patterns of temporal variation in online media. In *Proceedings of the fourth ACM international conference on Web search and data mining*. ACM, 177–186.

[28] Jaewon Yang and Jure Leskovec. 2015. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems* 42, 1 (2015), 181–213.

[29] Javad Zahiri, Joseph Hannon Bozorgmehr, and Ali Masoudi-Nejad. 2013. Computational prediction of protein–protein interaction networks: algorithms and resources. *Current genomics* 14, 6 (2013), 397–414.