A Reachability Query Method Based on Labeling Index on Large-Scale Graphs

Yuqing Duan[1]
Department of Electronic & Computer Engineering
Hong Kong University of Science and Technology
Hong Kong, China

Xuecheng Li[2], Linlin Ding[2+]
School of Information
Liaoning University
Shenyang, China
dinglinlin@lnu.edu.cn

***Abstract*—As a general data structure, graphs have been applied widely in many fields, for instance, geographical navigation, web semantic analysis, XML databases, etc. With the successful application of graphs in various fields, the rampant growth of graph data, and the structures of graph data becoming more complex, the analysis, storage and management for graph data are facing the unprecedented challenge. As a common analysis technology on large-scale DAG data, reachability query has been widely studied. Whereas, the existing reachability query mechanisms on large-scale graphs have some problems such as index creation requires a lot of time consumption and large storage space, query efficiency is low and so on. Therefore, in order to solve the above problems and implement efficient reachability query, we propose a labeling index method based on graph stratification (GSL), this method utilizes the properties of bipartite graph to link all vertexes into a mutually disjoint chain structure, creates unique chain-in label and chain-out label for each vertex; Besides this paper proposes two kinds of reachability query methods: chain-in reachability query and chain-out reachability query; Extensive experimental results verify the reachability query methods we propose have good query performance on the real-world networks, large-scale sparse graphs and dense graphs, increase efficiency of reachability query on large-scale DAG vastly.**

Keywords—large-scale DAG; reachability query; graph stratification; maximum matching graphs; labeling index scheme

I. INTRODUCTION

In recent years, with the advent of big data era, more and more scientific branches use graph data to model, utilize graph data to record the relationship between various entities and their attributes. A complex data structure emerges which has been widely applied in some emerging fields such as XML databases, social networks, geographical navigation, etc. With application and promotion of online communication platform such as Facebook, registration amount and usage amount are increasing at an alarming rate which drives rapid expansion of graph data scale. Graph data structure is getting more and more complexity that leads to new challenges for managing large-scale graphs effectively. However it has important significance for theoretical study and practical application.

With population growth in social platforms, strangers can contact each other through their respective circle of friends, then know and concern each other that makes more and more unfamiliar people communicate with each other. The relationship of two people is reflected by reachability, if the reachability is true, it illustrates these two have friend relation directly or indirectly, and they can even recommend their respective friends for each other, thus expand their circle of friends. In theoretical research of graphs, reachability is widely mentioned to reflect the capacity of one vertex to another. Therefore reachability query plays the fundamental role in large-scale graph analysis. However, the existing research about index and query are more applicable to the management of small and medium-scale graphs, the methods of handling large-scale graphs are insufficient. Whether large-scale graphs can be effective queried and optimized are critical to the management of graph databases. Meanwhile, many emerging areas have urgent need to handle reachability query effectively on graphs. Hence it has become new research hot.

So far, except traditional query methods, reachability query algorithms are applied to graphs could be classified to three categories roughly: transitive closure [1], online index [2] and hop encoding [3].

Paper [4] proposes tree index structure which can compress parts of transitive closure preferably. Whereas, this method need precompute transitive closure of each vertex, so space cost is quite expensive even creating index on small-scale graphs. Paper [5] improves storage overhead through removing single-source transitive closure, but this method still need oversize time and space cost. Although online index can realize the reachability query on large-scale graphs, but it has low query efficiency and poor reliability. Paper [6] proposes GRIPP, which needs little extra work during the process of query that increases response time of query immensely. Paper [7] proposes a reachability index (GRALL) on directed graphs, and this method is suitable for large-scale graphs, whereas interval inclusion relation is the need not sufficient condition for reachability, that influences reliability of reachability query seriously. Paper [8] proposes a 3-hop index strategy, and it has compressed index space and reduced the index creation time, whereas its extensibility is poor.

978-1-4673-9795-7/15 $31.00 © 2015 IEEE
DOI 10.1109/CSCI.2015.100

To sum up, existing reachability query index mechanism on large-scale graphs presents the following problems: expensive storage overhead, too long index creation time and low query efficiency. To solve the above problems, this paper proposes an index method based on graph stratification, it decomposes large-scale DAG to create reasonable label indexes for different types of vertexes to realize effective and efficient query. Meanwhile we propose two reachability query methods according to the types of vertexes so as to reduce the query response time further.

The rest of the paper is organized as follows. In Section 2 we review related work of existing reachability query. In Section 3 we present the index method based on graph stratification (GSL) in detail. We present two reachability query methods based on GSL in Section 4. Section 5 reports the experimental results and Section 6 concludes the paper.

II. RELATED WORK

At present, the existing reachability query on small and medium-scale graphs could mainly divided into several categories as follows: chain-decomposition, tree cover, path tree and hop encoding.

Reachability query based on chain-decomposition mainly applies path decomposition to improve computing efficiency of transitive closure and reduce its storage space. Chen Y et.al [9] propose an algorithm based on chain-decomposition that its time complexity is $O(n^2 + bn\sqrt{b})$, thereinto n is the amount of vertexes, and b is the depth of DAG. This method uses a mass of virtual vertexes to create chain structure that leads to oversize storage cost and needs large volume of memory support. With the decomposing of DAG, these neighbor vertexes will be assigned into different chains arbitrarily that will significantly affect the compression ratio of graphs and cause more expensive storage cost. Hence methods based on chain-decomposition are not suitable for large-scale graphs.

There are some other methods are based on the thought of tree cover. These methods transform a graph to a spanning tree, then utilize parent-child relationship of the spanning tree to allocate interval label for each vertex. Making use of inclusion relation of interval labels, reachability query can be realized. Paper [10] proposes interval-based method, which creates a mass of virtual coding that will lead to too long index creation time so that it cannot satisfy the reachability requirement of large-scale graphs.

Based on the thought of path tree, paper [11] proposes a novel efficient reachability query method on medium-scale graphs. The method partitions the original graph into several small-scale subgraphs, and then creates a ternary reachability label y for each vertex of the graph to identify the reachability between vertexes. Ruoming Jin et.al[12] propose a path-tree reachability query algorithm. Path-tree tries to find the most eclectic size of labels to reduce index storage cost.

With the advent of big data era, in recent years, many reachability query methods on large-scale graphs are proposed one after another, they mainly execute the query through online query [13]. For example, Hilmi Yildirim et.al first propose GRALL [14, 15] which supports online query. This method creates multiple reachability interval labels for each vertex based on the thought of random interval reachability index label, and through judging the relationship of interval labels, unreachable vertexes will be pruned. We can see that GRALL is more suitable for identifying the negative query between vertexes (unreachability), but for solving practical query paths between vertexes is still the bottleneck which is difficult to break through. Ruoming Jin et.al propose a novel reachability query frame named SCARAB [16]. It not only could make the reachability index extensible, but also accelerate online reachability query, so it is more suitable for large-scale graphs. But this method is not suitable for reachability query of dynamic graphs and the cases that query conditions are restricted. For finding better solution to positive query (i.e., directly identify reachability between vertexes), Stephan Seufert et al propose a reachability index method named FERRARI [17] which has adaptability, the primary innovation of this method is to set the range of spatial domain and two kinds of interval labels. Through two kinds of interval labels, it can optimize GRALL index structure, so as to query large-scale graphs more efficiently. Though it can identify reachability quickly through the two interval labels, but it is equivalent to create double interval labels for each vertex that leads to enormous increase of index creation cost, and the space complexity of index is O (n^2) in the best case.

Through the analysis of above methods, we can see the existing methods mostly are not suitable for large-scale graphs, and there are some problems such as oversize storage cost, too long index creation time and low query efficiency. It also provide direction for our research, in this paper we try to propose a method which can both reduce index creation time or index cost and proceed reachability query efficiently on large-scale graphs.

III. A LABELING INDEX METHOD BASED ON GRAPH STRATIFICATION

In this paper we mainly research reachability query on large-scale directed acyclic graphs (DAG). Because large-scale graphs often have thousands of vertexes, so we propose a labeling index method based on graph stratification named GSL, it transforms a large-scale DAG into a kind of chain structure which can represent reachability relationship between vertexes, and then creates reachability labels for each vertex on chains. GSL first applies graph stratification into the preprocessing stage, and decomposes the original graph into the chain structure utilizing the maximum matching graph of the bipartite graph. Then on the basis of generated chain structure, it creates independent reachability labels for vertexes on each chain by hierarchy. It creates relational reachability labels for vertexes on different chains according to the mutual relation between vertexes. Then the reachability relations between vertexes are preserved completely, and it can efficiently response reachability query in linear time.

A. Graph Stratification

Definition 1(Graph stratification): G=(V,E) represent a DAG ,The stratification of G is the decomposition of V into

one or more subsets $V_1, V_2 .... V_h$(i.e. $V = V_1 \cup V_2 \cup ... V_h$), and subnodes of each vertex in V_i only appear in $V_{i-1}, ..., V_1 \ (i = 2, \ldots, h)$, where h is the length of the longest path in G, i.e., the number of layers after stratification.

The common symbols and descriptions are listed in table I.

TABLE I. SYMBOLS AND DESCRIPTIONS

symbols	descriptions
G	DAG
$V_1, V_2, ..., V_h$	Subsets of vertex sets after strtification
Level(v)=i	Vertex v's layer number in V_i is i
C_j(v) (j < i)	The set that v points to and these subnodes belong to subsets V_j
G_1 / G_2	Subgraph that G_1 cuts out edges of G_2
$G_1 \cup G_2$	Subgraph that adds edges of G_1 and G_2
（v, u）	The edge from v to u
d(v)	Outdegree of v

The stratification method in this paper first finds all vertexes at the lowest layer, i.e., the vertexes that the outdegree are 0; Secondly it finds parent vertexes linking to these bottom vertexes, and judges whether other child vertexes of these parent vertexes belong to the lowest layer vertex sets, then gets the subnode sets of second layer; After that, it partitions large-scale directed acyclic graphs by iteration. The method of graph stratification is presented in Algorithm 1.

Algorithm 1: Graph Stratification

Input：DAG G
Output：results of graph stratification
(1) V_1 := all edges that its outdegree is 0;
(2) for i = 1 to h-1 do
(3) { T := all the vertexes which have one least child in V_i;
(4) for each vertex v in T do
(5) { let v_1, ..., vk be v's children and storage in V_i;
(6) $C_i(v)$:= {links to v_1, ..., v_k};
(7) if $d(v)$ > k then remove v from T;
(8) $G := G_i\{(v, v_1), ..., (v, v_k)\}$;
(9) $d(v) := d(v) - k$;
(10) end if
(11) }
(12) $V_{i+1} := T$;
(13) end for}

Known from the above algorithm, we can see that the features of vertexes generated by graph stratification method as follows: (1) The vertexes are mutual independence and cannot reach each other at the same layer; (2) The vertexes in i layer only point to the vertex in the next layer or the vertex which the layer number is less than i. And the vertexes at any layer cannot reach vertexes at their upper layer.

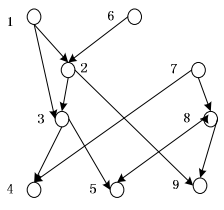


Fig. 1. DAG G

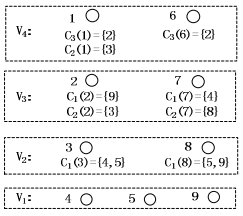


Fig. 2.The result of G stratification

As an example, in Fig.1, there are 9 vertexes in G. Fig.2 presents the stratification result by the above algorithm, the figure only shows the stratification result of vertexes. The vertexes in Fig.1 are divided into 4 layers: V_1={4,5,9}, V_2={3,8}, V_3={2,7} and V_4={1,6}. We can see that, for the vertexes at each layer, their pointing vertexes are all allocated to other layers.

B. Creating Maximum Matching Graph

Generating independent layers of vertex sets through graph stratification, and any two vertex sets have not intersection. Then it links vertexes of different layers based on maximum matching graph of bipartite graphs to generate a cluster of disjoint chains, so it convenient for creating reachability labels and compressing transitive closure.

Definition 2.2(Bipartite graph): Given an undirected graph $G(V,E)$, its vertex sets can be divided into two subsets P and Q, and it meets two conditions:(1) $V=P \cup Q$; (2) $P \cap Q=\Phi$ $\forall \langle u, v \rangle \in E$, $u \in P$, $v \in Q$

If a graph meets above conditions, it is bipartite graph, represented by G (P, Q, E).

Definition 2.3(Matching): For a bipartite graph G, X is a subgraph of G, if any two edges in the edge sets of X do not connect to the same vertex, then it will be named a matching of the bipartite graph. The subgraph with the most edges is called the maximum matching subgraph.

The main idea of creating maximum matching graph is as follows: first to get the maximum matching graph, it regards the vertex sets in the lowest two layers as two subsets of the bipartite graph. And then the generated maximum matching graph is combined with the vertex set in the third layer into a new bipartite graph, and continues getting the second maximum matching graph, by this analogy. Finally merging all generated maximum matching graphs to get the maximum matching graph of the original graph, the generated structure contains n disjoint chains and the information of all vertexes and edges on the original graph. The process of creating maximum matching graph need virtual vertexes [18], but these vertexes don't participate creating reachability labels, so it won't occupy any index storage space, it only affects a little index creation time. The method of creating maximum matching graph is shown in Algorithm 2.

Algorithm 2: Creating Maximum Matching Graph

input：result of stratification of G
output：the maximum matching
(1) find M_1 of G(V_2, V_1: C_1); $M_{1'} := M_1$; $V_{1'} := V_1$; $C_1 := C_1$;
(2) for i = 2 to h - 1 do
(3) { construct virtual vertexes for V_i according to M_i-1;
(4) let U be the set of the virtual vertexes added into V_i;
(5) let W be the newly generated edges incident to the new vertexes in V_i
(6) let W' be a subset of W, containing the edges from V_{i+1} to U;
(7) $V_{i'} := Vi \cup U$; $C_{i'} := C_i \cup W'$;
(8) find a maximum matching $M_{i'}$ of $G(V_{i+1}, V_{i'}; C_{i'})$;
(9) }
(10) end for
(11) return $M_{1'} \cup M_{2'} \cup \ldots \cup M_{h-1'}$.

C. Generating Reachability Labels

If large-scale graphs aren't decomposed, their transitive closure is tremendous. Whereas this paper decomposes the original graph into a cluster of disjoint chains through stratification and generating maximum matching graph of bipartite graphs, and the generated chains contain all information about vertexes and edges. Because the graphs in this paper are presented by chain structure, each vertex just exists in one chain, so as to better create reachability labels. The paper sets two kinds of reachability labels: chain-in label and chain-out label according to the location on chains and relatedness between vertexes.

We can see that each vertex has one and only one chain-in label, because each vertex just exist in one chain; Secondly according to the thought of stratification, the bottom vertexes in each chain (i.e., the vertex which the outdegree is 0) haven't chain-out labels, and the vertex in non-bottom chains has least one chain-out label. In this process, the reachability information of any edge could be preserved essentially, and its time complexity could be minimized to O (nh^2), where n is the amount of vertexes, and h is the amount of chains.

For example, we still take Fig.1 creating reachability labels for all vertexes in the new chains by the above mentioned algorithm. The result is shown in Fig. 3.

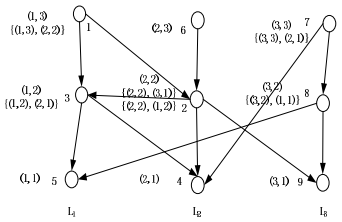


Fig.3.Result of creating reachability label

IV. Reachability Query Based on the GSL

In allusion to the chain-in label and chain-out label, this section will propose Chain-in Reachability Query (CIRQ) method and Chain-Out Reachability Query (CORQ) method.

A. Chain-in Reachability Query

For the reachability query of two vertexes u and v, we need compare the chain numbers of vertex u and v, if the chain number of vertex u equals vertex v's, we know that two vertexes exist in the same chain, chain-in reachability query can be proceeded directly.

In this paper, the graph stratification of DAG is bottom-up on the basis of the reachability relation between vertexes, that is to say lower layer vertexes cannot reach upper layer vertexes. Therefore, if two vertexes are in the same chain, the reachability of vertexes can be determined just by the layer number. It is not necessary to compare the chain-out label of two vertexes. In other words, in the case of the reachability query of vertexes that are in the same chain, moreover the amount of vertexes in the chain is oversize, and the CIRQ can finish the judgment just by once reachability query. Hence, the time complexity of CIRQ is O (1). It will reduce the time of reachability query enormously and increase the efficiency of reachability query effectually.

B. Chain-out Reachability Query

For two vertexes that the chain numbers are unequal, this paper proposes Chain-out Reachability Query (CORQ) method, the concrete operations as follows:

First, comparing the chain numbers i_u and i_v of vertex u and v (assume the labels of vertex u and v are (i_u, j_u), (i_v, j_v)).if $i_u \neq i_v$, then it illustrates that the vertex u and v are not in the same chain, chain-out reachability query is needed.

Secondly, comparing the containment of chain-in labels (i_u,j_u), (i_v,j_v) and chain-out labels $\{(i_u,v_u)$, $(j_u,w_u)\}$,$\{(i_v,v_v)$, $(j_v,w_v)\}$. If one chain-out label of vertex u contains the chain-in label of vertex v, then it illustrates vertex u and v are reachable; If anyone chain-out label of vertex u doesn't contain the chain-in label of vertex v, then it illustrates vertex u and v aren't reachable.

It is known by the character of the maximum matching graph, the maximal length of each chain is k that is to say after graph stratification, the maximal number of stratification is k. And then it is known that for each vertex on the graph, the maximal number of the chain-out label is $(k$-1$)(n$-1),n is the number of chain-in the maximum matching graph. In the other words, the chain-out reachability query of two vertexes that in different chains need the judgment of containment at most (k-1)(n-1). Hence, the time complexity of CORQ is O (kn).

V. Experimental Evaluation

We present the evaluation results of reachability query method based on GSL in this section. In terms of execution efficiency and performance, we compare our method with other existing reachability query methods of DAG, such as Optimal-chain, Dual-Labeling, 2-Hop, Tree-Path and FERRARI. We compare the index creation time, storage overhead and reachability query time on the different data volume levels, and analyze the experimental results.

A. Experiment Environment

The experiments are evaluated on a commodity computer which has two Intel Core2 T6400@2.00GHz CPUs, 4GB memory and 1TB hard disk. It can process large-scale DAG graph datasets that are set in this paper. The operation system is Windows 7 Ultimate (32 bit/SP1). Visual Studio2012 is installed on the computer, and all the algorithms are implemented in java.

B. Datasets

We conduct our experiments on simulated datasets and real-world datasets [21, 22], whose statistics are described in table II. However, simulated datasets and large-scale DAG graph are generated by Scale-Free Model [20] and Gtgraph [21] randomly.

TABLE II. BASIC INFORMATION OF REAL-WORLD DATASETS

Dataset	#vertexes	#edges
Wiki	478467	889361
EuAll	578042	1870425
Facebook	489214	1092482
Google	842371	4329532

C. Experiment Results Analysis

In this paper, our experiments are evaluation through two aspects. On the one hand we verify whether the index creation time and storage overhead for large-scale DAG with GSL method are reasonable. On the other hand we verify whether the methods we propose can realize the reachability query between vertexes on the large-scale DAG.

Experiment 1: Simulated sparse graph experiment

Table III lists the comparison of index creation time and storage overhead on sparse graph with GSL method and other five methods. It is observed that GSL and Tree-path method have more obvious advantage in index creation time. GSL method transforms the large-scale DAG into a chain, distributes the relational vertexes and edges in the same chain, and then that is convenient to create the effective index for each vertex on the graph. On the aspect of storage overhead, GSL method is not optimal, but comparing with 2-Hop method, the occupied storage space is much less.

TABLE III. TIME OF INDEX CREATION AND STORAGE OVERHEAD ON SPARSE GRAPH

Method	Index creation time(s)	Storage overhead(16bits)
Optimal-chain	75.450	198273
Dual-Labeling	53.932	420893
2-Hop	298.357	1092482
Tree-Path	23.345	401233
FERRARI	40.426	163574
GSL	28.345	468425

Fig.4(a). shows the average running time of reachability query on the sparse graph with GSL method and other methods. It is observed that GSL method is optimal, this is because the large-scale graph in this paper is decomposed into the minimized chain structure (that is to say the owned the number of the chain is least), moreover vertexes with reachability relation are distributed in the same chain as much as possible, therefore it improves the query efficiency.

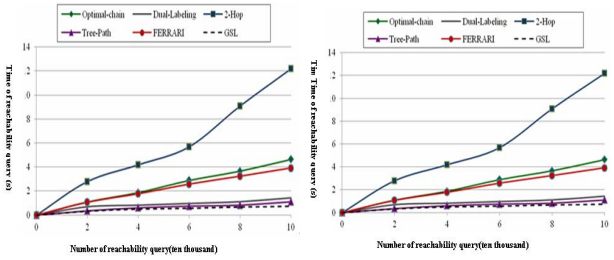
Experiment 2: Simulated dense graph experiment

Table IV lists the comparison of index creation time and storage overhead on dense graph with GSL method and Optimal-chain method, Dual-Labeling method, Tree-Path method and FERRARI method. Because 2-Hop method only compresses the transitive closure between vertexes, ignores labels of edges, it is not suitable for the dense graph. As shown in table IV, using GSL method on the dense graph is average level. This is because dense graph has major edges, that is to say there are major pairs of reachability relation between vertexes, and it is needed to create major chain-out labels. But if vertexes on the dense graph can be gathered into several groups nicely and the reachability relations between vertexes are distributed in several concentrated subgraphs, then vertexes on each subgraph will better exist in the same chain among chain structure and chain-out labels between vertexes will be less, so index creation time and storage overhead will be optimized and reachability query is more efficient.

TABLE IV. TIME OF INDEX CREATION AND STORAGE OVERHEAD ON DENSE GRAPH

Method	Index creation time(s)	Storage overhead(16bits)
Optimal-chain	223.921	406698
Dual-Labeling	2083.341	980642
Tree-Path	94.019	174804
FERRARI	1763.01	372821
GSL	1437.264	680192

Fig.4 (b) shows the comparation of reachability query on the dense graph with GSL method and other four methods. It is observed intuitively that the query performance of GSL method is optimal, and this is because if the query proceeds in vertexes that are in the same chain, the time complexity of query is O (1). So among the existing reachability query methods aiming at large-scale DAG, and the query time using the method we propose is minimum.



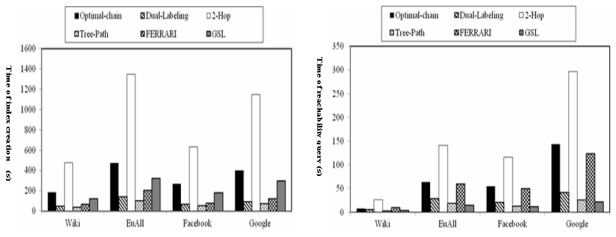
(a) (b)

Fig.4 Time of reachability query on the sparse graph

Experiment 3: Real-world datasets experiment

The comparison of index creation time shows in Fig.5(a), which is tested on real-world datasets using GSL method, Optimal-chain method, Dual-Labeling method, 2-Hop method, Tree-Path method and FERRARI method. It can be found that index creation time using six methods presents the trend of rising with the expansion of real-world datasets.

The comparison of reachability query time on real-world datasets of different scale using GSL method in this paper and other five methods shows in Fig.5(b). When the amount of edges on the original graph is less, the query efficiency of six methods is approximate. Moreover, query time shows obvious change with the increase of graph scale. Therein reachability query time using GSL method is minimum, Dual-Labeling and Tree-Path take second place, Optimal-chain and FERRARI are closely, and 2-Hop is worst. GSL method we propose has obvious advantage that is because when the reachability query is processed, if two vertexes are in the same chain, then the judgment of reachability can be processed through the size of labels. It will save query time immensely.



(a) (b)

Fig.5 Time of reachability query on real-world datasets

Synthesizing the above two sets of experiment data, we know that Optimal-chain, Dual-Labeling, 2-Hop and Tree-Path are more suitable for small-scale DAG, moreover GSL and FERRARI can be well applied to reachability query on large-scale DAG, meanwhile query efficiency of GSL is optimal.

VI. CONCLUSIONS

We propose a labeling index method based on graph stratification (GSL, Graph Stratification Labeling), utilizing this method can create standalone chain-in reachability label and chain-out reachability label for each vertex effectively Then based on the GSL method, we propose two kinds of reachability query methods that use for adapting the requirement of reachability query on large-scale DAG. Through a mass of simulation experiment analysis aiming at different kinds of graph, it illuminates that the method we propose is efficient and feasible. It can increase the efficiency of reachability query on large-scale DAG on the premise of guaranteeing accuracy of reachability judging results.

ACKNOWLEDGMENT

This work was supported by National Natural Science Foundation of China under Grant (Nos.61472169, 61502215); Science Research Normal Fund of Liaoning Province Educatio n Department (No.L2015193);Doctoral Scientic Research Star t Foundation of Liaoning Province(No.201501127);Young Re search Foundation of Liaoning University under Grant (No. L DQN201438).

REFERENCES

[1] van Schaik S J, de Moor O. A memory efficient reachability data structure through bit vector compression[C]//Proceedings of the 2011 ACM SIGMOD International Conference on Management of data. ACM, 2011: 913-924.

[2] Yıldırım H, Chaoji V, Zaki M J. GRAIL: a scalable index for reachability queries in very large graphs[J]. The VLDB Journal—The International Journal on Very Large Data Bases, 2012, 21(4): 509-534.

[3] Cai J, Poon C K. Path-hop: efficiently indexing large graphs for reachability queries[C]// Proceedings of the 19th ACM international conference on Information and knowledge management. ACM, 2010: 119-128.

[4] Jin R, Hong H, Wang H, et al. Computing label-constraint reachability in graph databases[C]//Proceedings of the 2010 ACM SIGMOD International Conference on Management of data. ACM, 2010: 123-134.

[5] Xu K, Zou L, Yu J X, et al. Answering label-constraint reachability in large graphs[C]// Proceedings of the 20th ACM international conference on Information and knowledge management. ACM, 2011: 1595-1600.

[6] Trißl S, Leser U. Fast and practical indexing and querying of very large graphs[C]// Proceedings of the 2007 ACM SIGMOD international conference on Management of data. ACM, 2007: 845-856.

[7] Yildirim H, Chaoji V, Zaki M J. Grail: Scalable reachability index for large graphs[J]. Proceedings of the VLDB Endowment, 2010, 3(1-2): 276-284.

[8] Jin R, Xiang Y, Ruan N, et al. 3-hop: a high-compression indexing scheme for reachability query[C]//Proceedings of the 2009 ACM SIGMOD International Conference on Management of data. ACM, 2009: 813-826.

[9] Chen Y, Chen Y. An efficient algorithm for answering graph reachability queries[C]//Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on. IEEE, 2008: 893-902.

[10] Wang H, Li J, Luo J, et al. Hash-base subgraph query processing method for graph-structured XML documents[J]. Proceedings of the VLDB Endowment, 2008, 1(1): 478-489.

[11] Jin R, Xiang Y, Ruan N, et al. Efficiently answering reachability queries on very large directed graphs[C]// Proceedings of the 2008 ACM SIGMOD international conference on Management of data. ACM, 2008: 595-608.

[12] Jin R, Ruan N, Xiang Y, et al. Path-tree: An efficient reachability indexing scheme for large directed graphs[J]. ACM Transactions on Database Systems (TODS), 2011, 36(1): 7.

[13] van Schaik S J, de Moor O. A memory efficient reachability data structure through bit vector compression[C]//Proceedings of the 2011 ACM SIGMOD International Conference on Management of data. ACM, 2011: 913-924.

[14] Yıldırım H, Chaoji V, Zaki M J. GRAIL: a scalable index for reachability queries in very large graphs[J]. The VLDB Journal—The International Journal on Very Large Data Bases, 2012, 21(4): 509-534.

[15] Yildirim H, Chaoji V, Zaki M J. Grail: Scalable reachability index for large graphs[J]. Proceedings of the VLDB Endowment, 2010, 3(1-2): 276-284.

[16] Jin R, Ruan N, Dey S, et al. SCARAB: scaling reachability computation on large graphs[C]//Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. ACM, 2012: 169-180.

[17] Seufert S, Anand A, Bedathur S, et al. Ferrari: Flexible and efficient reachability range assignment for graph indexing[C]//Data Engineering (ICDE), 2013 IEEE 29th International Conference on. IEEE, 2013: 1009-1020.

[18] Chen Y, Chen Y. An efficient algorithm for answering graph reachability queries[C]//Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on. IEEE, 2008: 893-902.

[19] Wang H. Managing and mining graph data[M]. New York: Springer, 2010.

[20] Madduri K, Bader D A. GTgraph: A suite of synthetic random graph generators[J]. 2012.

[21] Jin R, Ruan N, Dey S, et al. SCARAB: scaling reachability computation on large graphs[C]//Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. ACM, 2012: 169-180.

[22] Cheng J, Shang Z, Cheng H, et al. K-reach: who is in your small world[J]. Proceedings of the VLDB Endowment, 2012, 5(11): 1292-1303.