Neurocomputing 162 (2015) 67–84

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

A hierarchical model for label constraint reachability computation

Luo Chen *, Ye Wu, Zhinong Zhong, Wei Xiong, Ning Jing

College of Electronic Science and Engineering, National University of Defense Technology, Changsha 410073, China

ARTICLE INFO

Article history:
Received 14 December 2014
Received in revised form
22 March 2015
Accepted 1 April 2015
Communicated by Rongrong Ji
Available online 17 April 2015

Keywords:
Label-constraint reachability
Bipartite hierarchical model
Two-hop encoding
Hierarchical labeling

ABSTRACT

Reachability query is a fundamental problem on networks currently emerging in various application domains. However, very little existing work has studied reachability with label constraints imposed on the edges/vertices of graphs. In this paper, we study the problem of answering *L*abel-*C*onstraint *R*eachability (LCR) in a labeled directed graph, i.e., whether a directed path confined to a given label set from a source query vertex to a target query vertex in the input graph exists. We propose a model called the *Bi*partite *Hi*erarchical (BiHi) to speed up the LCR computation. The BiHi model adopts a representation organized as hierarchies of bipartitions, and a vertex-cover-based algorithm to label the to-most remaining subgraph. In addition, by extending the two-hop labeling, the BiHi model introduces a top-down hierarchical labeling strategy to compress the index. Extensive theoretical analyses and experimental investigations on both real-life and synthetic datasets demonstrate that our method can reduce the memory cost and construction time of index construction while maintaining high query processing efficiency.

1. Introduction

Graphs are common in real-life settings: web-links [1], mobile networks [2], road networks [3], social networks [4], and social media [5], to name a few. In fact, graphs are becoming an essential tool in modeling sophisticated entities and their interactions. Among many types of graph queries, graph reachability query has been the most fundamental research problem, which asks whether a path exists from one entity to another. Efficient processing of reachability queries is a critical issue in the graph research field.

In the last several years, answering graph reachability queries has drawn considerable research interest [6]. In short, graph reachability query methods could be largely classified into three categories [7]: transitive closure compression, hop labeling, and online search. The transitive closure compression approaches precompute all path-labels between any two vertices; they tend to be fast but can hardly scale to large graphs due to their high computation cost. The online search methods answer the reachability query in real-time; they are generally slower but can easily scale to large graphs [8]. The hop labeling method is a trade-off between the transitive closure and online search methods; it is able to deliver more compact indices than the transitive closure and also offers better query performance than the online search methods.

Furthermore, typical hoping methods include tree cover [9], chain cover [10], dual labeling [11], GRIPP index [12], path-tree cover [13], 2-hop cover [14], and randomized interval labeling [15]. The hop labeling methods utilize intermediary vertices to encode the reachability, i.e., each vertex u records a list of intermediate vertices it can reach ($L_{out}(u)$) and a list of intermediate vertices that can reach it ($L_{in}(u)$). These two sets record all of the necessary information to calculate the reachability of any pair of vertices.

In fact, many real-world graphs are associated with labels on vertices or edges to denote various types of entities or relationships, such as friendship, support, co-membership, and marriage. One often wants to query the connectivity of a pair of vertices via edges or vertices of particular types. In many real-world applications, the answers to reachability queries are meaningful only if the edge/vertex labels are also captured in the reported path, as illustrated by the following application examples.

Transportation network: If a firm plans to ship its products from a factory to a retail store in a distant location by truck. Taking into account of the public health and safety, a truck is permitted to pass through some places. An LCR query can help to determine whether there is a feasible delivery route between these two locations.

Biological networks: Metabolism can be viewed as a network of chemical reactions catalyzed by enzymes and connected via their substrates and products; a metabolic pathway is then a coordinated series of reactions. Given a set of co-regulated genes, finding a path or pathway that could be formed with the catalyzed reactions can be modeled as the LCR query problem.

* Corresponding author.
E-mail address: luochen@nudt.edu.cn (L. Chen).

http://dx.doi.org/10.1016/j.neucom.2015.04.003
0925-2312/

There have been few works considering the label constraint on paths. To balance between storage and construction efficiency, Jin et al. [16] employ a sampling tree and a partial transitive closure to compress the full transitive closure. During the query stage, a combination of the tree structure and partial closure is constructed to restore the complete transitive closure between two vertices. However, there are three limitations of the sampling tree method: (1) A full transitive closure still needs to be computed first, whose time and space complexity are $O(n^3)$ and $O(n^2)$, respectively. (2) A single spanning tree cannot compress the transitive closure greatly, especially for dense graphs. (3) The construction time of a sampling tree is large even though on it is used small graphs, which becomes worse and unaffordable for large graphs. Xu and Zou [17] further improve the efficiency of the sampling tree method by removing redundant paths. Their studies may have been more reasonable if they had abandoned the materialization of the transitive closure.

The path query is closely related to the label-constraint reachability problem. To improve query efficiency, one of the common techniques is to partition a graph to build a hierarchical structure. The pioneer work can be traced to path computation in navigation systems. HEPV [18] partitions a large graph into smaller subgraphs and encodes partial paths in a hierarchical manner. By recursively extracting the reachability backbone from the origin graph [8], one can obtain a multi-level structure, such that non-backbone vertices can reach backbone vertices in limited steps. When answering reachability queries, lower-level vertices need to go through upper-level vertices (but not vice versa). Furthermore, by recursively folding an input graph into half each time, the TF-label [19] and IS-label [20] are used to reduce the label size, thus improving query efficiency greatly. However, graph partition techniques for LCR queries impose many research challenges, of which we consider two in this paper: (1) how to retain constraint reachability while folding or partitioning a graph; (2) for non-planar graphs, the number of crossing edges between two subgraphs may be very large, leading to too many boundary vertices and a large backbone. Jin [21] and Akiba [3] propose a pruning strategy during the breadth-first search for distance labeling, which can be used to answer distance queries in large networks. However, label-constraint reachability is different from the distance query in that there will be more than one qualified path between two vertices, making it considerably more complicated.

To address these issues, in this paper, we propose a compressed constraint path index to answer the LCR query. Because a vertex-labeled constraint can be translated into an edge-labeled constraint, we mainly focus on graphs with the label constraint on edges. Specifically, the major contributions are summarized as follows:

1. We design a hierarchical model to compute the hop index by exploiting bipartition and independent property, which reduces the size of the graph and keeps Reach-Label maintenance property recursively.
2. A cover-based 2-hop labeling algorithm is proposed on the topmost remaining graph by exploring a greedy strategy, which utilizes vertex order and hop coverage.
3. We propose a top-down hierarchical labeling algorithm for index construction and study a LCR query strategy based on the labeling. In addition, the correctness proof and complexity analyses of the proposed algorithms are given.
4. Extensive experiments on real-life and synthetic networks are conducted. The construction time of our method is orders of magnitude faster than the traditional transitive closure and sampling tree methods. The proposed algorithm has a comparable or faster query time than the state-of-the-art methods on large graphs and is in microseconds and remains stable regardless of the network size, density or constraint label size.

The rest of the paper is organized as follows. Section 2 gives the preliminaries and problem definition. Section 3 gives a detailed description of the proposed model and algorithms. Section 4 shows our experimental results on real and synthetic networks. We conclude our paper and offer further study in Section 5.

2. Preliminaries

Definition 1 (*Edge-labeled graph*). An edge-labeled (directed) graph is defined as a 4-tuple $G=(V,E,\mathbb{L},f_e)$, where V is the set of vertices, E is the set of edges, $\mathbb{L}$ is the set of edge labels, and $f_e: E\mapsto\mathbb{L}$ defines a mapping function from edges to label set, which assigns each edge $e\in E$ to a label $f_e(e)\in\mathbb{L}$. Furthermore, we use $L(p)$ to denote the set of all labels on the path p.

An arbitrary number of labels on a single edge is allowed. This is a reasonable generalized assumption for possible applications. For example, in a social network, two people may communicate with each other by phone, email or short messaging service. Edges with no label are permitted, because the relation of arbitrary type is not handled. Generally, self-loops, i.e., edges starting from and ending with the same vertex with no intermediate vertices, are removed, as one can always reach itself.

Definition 2 (*Edge-labeled constraint reachability*). Given two vertices $u,v\in G$, and a label set $\mathcal{S}\subset\mathbb{L}$, if there is a path p from vertex u to v satisfying $L(p)\subseteq\mathcal{S}$, then u can reach v with the label-constraint $\mathcal{S}$, denoted as $u\overset{\mathcal{S}}{\rightsquigarrow}v$.

Example 1. Throughout the paper, we use Fig. 1 as a running example, where integers represent vertices and, letters and different edge colors represent edge labels. As an example, vertex 10 can reach vertex 13 with labels $\{a,c\}$ and $\{a,b\}$. Namely, vertex $10\overset{\mathcal{S}}{\rightsquigarrow}13$, where $\mathcal{S}=\{a,c\}\cup\{a,b\}$. Furthermore, path $(10,7,8,12,13)$ is a $\{a,b\}-$path between 10 and 13.

In practice, labels are also associated with vertices to describe entities. Similar to an edge-labeled graph, a vertex-labeled graph is defined as $G=\{V,E,\mathbb{L},f_v\}$, where $f_v: V\mapsto\mathbb{L}$ is a function that assigns labels $f_v(u)\in\mathbb{L}$ to a vertex $u\in V$. Usually, it is feasible to translate vertex-labeled constraints to edge-labeled constraints as illustrated in Fig. 2. A path p goes through vertex v_i and v_j, and their labels are A and B, respectively. If we wish to translate it into an edge-labeled form, a replaced path p' should be equipped with labels A and B between v_i and v_j. Therefore, for an arbitrary edge $e\in E$ ending with v_i and v_j, we introduce a dummy vertex v_k, such that $f_e(v_i,v_k)=f_v(v_i)$ and $f_e(v_k,v_j)=f_v(v_j)$. Then, the LCR query on a vertex-labeled graph can be equivalently answered by a query on a edge-labeled graph. As a result, we mainly focus on the edge-labeled constraint reachability query in the following.

To enhance the readability of this paper, Table 1 lists the symbols frequently used in this paper.