Information Systems 69 (2017) 180–193

Contents lists available at ScienceDirect

Information Systems

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

Shrink: Distance preserving graph compression

Amin Sadri[a,*], Flora D. Salim[a], Yongli Ren[a], Masoomeh Zameni[b], Jeffrey Chan[a], Timos Sellis[c]

[a] School of Sience, RMIT University, Melbourne, Australia
[b] School of Computing and Information Systems, University of Melbourne, Melbourne, Australia
[c] School of Software and Electrical Engineering, Swinburne University of Technology, Melbourne, Australia

ARTICLE INFO

Article history:
Received 10 January 2017
Revised 19 April 2017
Accepted 1 June 2017
Available online 7 June 2017

Keywords:
Graph compression
Graph simplification
Graph databases
Shortest paths

ABSTRACT

The ever increasing size of graphs makes them difficult to query and store. In this paper, we present *Shrink*, a compression method that reduces the size of the graph while preserving the distances between the nodes. The compression is based on the iterative merging of the nodes. During each merging, a system of linear equations is solved to define new edge weights in a way that the new weights have the least effect on the distances. Merging nodes continues until the desired size for the compressed graph is reached. The compressed graph, also known as the coarse graph, can be queried without decompression. As the complexity of distance-based queries such as shortest path queries is highly dependent on the size of the graph, *Shrink* improves the performance in terms of time and storage. *Shrink* not only provides the length of the shortest path but also identifies the nodes on the path. The approach has been applied to both weighted and unweighted graphs including road network, friendship network, collaboration network, web graph and social network. In the experiment, a road network with more than 2.5 million nodes is reduced to fifth while the average relative error is less than 1%.

Crown Copyright © 2017 Published by Elsevier Ltd. All rights reserved.

1. Introduction

Nowadays, it is increasingly common to find graphs with millions of nodes in various domains. Due to the commonness of large graphs, graph compression is becoming an important research topic. In this process, also known as graph simplification, the complexity of the graph is reduced while certain characteristics of the graph are preserved. Graph compression can be performed by reducing the number of edges, nodes or extracting a high-level abstraction of the graph. A similar problem is graph summarization where summary graph uncovers the underlying topology of the original graph [12,13,24,34].

Graph compression has three main purposes. First, graph compression algorithms produce a simpler graph that can be queried faster than the original graph. This is useful for graph-based mining algorithms and also more complex problems (e.g. measuring similarities between graphs). Distance-based queries such as shortest-path have a great importance [6,7,19,35,42]. Many fundamental tasks in graph mining, such as computing diameter, closeness, centrality, and betweenness centrality, are dependent on computing shortest path distance. It has countless applications in transportation networks [3,23,43], networking [8,40], and databases [33]. Distance preserving graph compression speeds up the shortest path queries because they run on a smaller graph. Second, the compressed graph, also known as the coarse graph, can be stored in less space. Over the past few years, due to increasing the size of graph-structured databases, it becomes challenging and expensive to store the data, and graph compression techniques are deployed to reduce space consumption [10,24,25]. Third, graph compression algorithms help the users to understand and visualize the high-level structure of the graph [4,21,30] [21] [30]. It is almost impossible to understand the information encoded in large graphs with thousands or even millions of nodes by only visual inspection [36]. The coarse graph, produced by compression, is smaller and easier to be visualized [18].

Despite the importance of distance-based queries, only a few compression methods were proposed to preserve the distance [32,37]. Moreover, most of these existing methods are designed to compress unweighted graphs, and are not compatible with weighted graphs [9,10,14]. Furthermore, another challenge is that sometimes, the compression ratio needs to be chosen by the user. In this case, the user is able to control the trade-off between the accuracy and the size of the compressed graph. However, in exist-

* Corresponding author.
E-mail addresses: amin.sadri@rmit.edu.au (A. Sadri), flora.salim@rmit.edu.au (F.D. Salim), yongli.ren@rmit.edu.au (Y. Ren), mzameni@student.unimelb.edu.au (M. Zameni), jeffrey.chan@rmit.edu.au (J. Chan), tsellis@swin.edu.au (T. Sellis).

http://dx.doi.org/10.1016/j.is.2017.06.001
0306-4379/Crown Copyright © 2017 Published by Elsevier Ltd. All rights reserved.

ing methods [16,32], the compression ratio is determined by the method, not the user.

In this paper, we introduce a novel distance preserving compression method *Shrink*, which can be used to query and store both weighted and unweighted graphs, e.g. enhancing shortest-path or other distance-based algorithms. Compressing with *Shrink* has the least effect on the distances between nodes. Specifically, when merging two nodes to a supernode, a system of equations is introduced to minimize the distance variations caused by this merge. The rationale behind this is to keep the mean of caused error equal to zero. These equations determine the edge weights connected to the supernode. After each merging, the number of nodes decreases by one and merging stage, called coarsening stage, continues until the desired size is achieved. The next stage is executing stage where the distance based query runs on the coarse graph. The last stage, refining stage, is optional that provides the path between the queried nodes. We have theoretically proved that for long paths, the error of compression converges to zero if merging errors are independent.

To sum up, *Shrink* possesses the following features: (1) it is linear in the number of nodes when $\sigma \ll |V|$, where $|V|$ is the number of nodes and σ is the average degree (see Section 4.2). This is common in large graphs with thousands or millions of nodes; (2) the larger the original graph is, the more accurate *Shrink* is. The reason is that large graphs usually have long paths and *Shrink* has less effect on the length of the long paths (see Section 3.2 and 6.4); (3) the error rate and the compression ratio are adjustable; (4) it provides not only distances but also the corresponding instances (nodes) of the shortest paths; (5) it is applicable to all types of distance queries, including reachability, single-source shortest-path (SSSP), all-pairs shortest path (APSP), closeness centrality and betweenness centrality. The experiment results show that compressing a two-million-node graph into fifths has the average error less than 1%.

The main contributions of the paper are as follows:

- We propose a compression method *Shrink* that has the least effect on the distances between the nodes.
- *Shrink* is fast and linear time complexity in the number of nodes, $O(|V|)$. Hence, it is applicable to large graphs.
- The proposed method is evaluated on both weighted and unweighted real-world data sets, including road network, friendship network, collaboration network, web graph and social network, etc.

In the next section, we have definitions and problem statement. In Section 3, first, we present the baselines for defining the equations and discuss why the equations are suitable for assignment of the new weights. Then, an overview of *Shrink* is provided. Three stages of *Shrink* are described in section 4 and 5. We evaluate our method in terms of time, accuracy and storage in Section 6. Finally, Section 7 and 8 discuss related work and conclusion, respectively.

2. Problem formulation

In this section, we first introduce the necessary notation to describe the problem formulation, and then we state the problem.

Definition 1. *Original graph* is the input graph which is a triple $G = (V, E, w)$ where V is a set of nodes (or vertices), $E \subset V \times V$ denotes edges, and $w : E \to R^+$ assigns a non-negative weight to each edge $e \in E$.

In this paper, the original graphs can be either unweighted or weighted. For unweighted graphs, the same weight can be assigned to all edges. The notations used in this paper are listed in Table 1.

Definition 2. *Coarse graph*, $G' = (V', E', w')$, is the compressed graph. $V' = \{v_1', \ldots, v_n'\}$ is a partition of V (i.e. $v_i' \subset V$ for all $i, \bigcup_i v_i' = V$, and $v_i' \bigcap v_j' = \emptyset$ for all $v_j \neq v_i$). Namely, each node $v_i' \in V'$, also known as a supernode, may consist of some nodes in G. E' denotes the edges set $E' \subset V' \times V'$, $w' : E' \to R^+$. In contrast to the nodes, there is no mapping between the edges.

Table 1
Definition of the variables.

Variable	Definition
G	Original graph: $G = (V, E, w)$
V	Set of the nodes in the original graph
E	Set of the edges in the original graph: $E \subset V \times V$
w	Weights on E in the original graph:$w : E \to R^+$
x, y	two nodes in the original graph $x \in V, y \in V$
u, v	To be merged nodes in the original graph $u \in V, v \in V$
G'	Coarse graph: $G' = (V', E', w')$
V'	Set of the nodes in the coarse graph
E'	Set of the edges in the coarse graph: $E' \subset V' \times V'$
w'	Weights on E' in the coarse graph: $w' : E' \to R^+$
v'	Supernode in the coarse graph $v' \in V'$
k	Number of v's neighbors
$N(u)$	Set of u's neighbors that are not connected to v
$N(v)$	Set of v's neighbors that are not connected to u
$N(uv)$	Set of Common neighbors of u and v
N	Set of neighbors of u and v: $N = N(u) \cup N(v) \cup N(uv)$
v_i	A neighbour of u or v
w_{vi}	weight of the edge between v_i and v
w_{ui}	weight of the edge between v_i and u
w'_{vi}	weight of the edge between v_i and v'
$l(v_i, v_j)$	Length of the path that connects v_i and v_j and crosses u or v in the original graph
$l'(v_i, v_j)$	Length of the path that connects v_i and v_j and crosses v' in the coarse graph

$$E' = \{(u', v') | u \in u', v \in v', (u, v) \in E\} \quad (1)$$

Specifically, two supernodes are connected if and only if there is a node in one supernode that is connected to a node in the other supernode. Here, the main problem is assigning weights to the new edges. To this end, we define and solve equations to have new weights with the least effects on the distances between nodes.

Definition 3. The *distance* between x and y is $d(x, y)$, which is the length of the shortest path between x and y in the original graph. The shortest path is a path with the lowest total sum of edge weights. Similarly, $d'(x, y)$ denotes the length of the shortest path between the supernodes that contain x and y in the coarse graph.

Definition 4. Error of the compression for nodes x and y, $Err(x, y)$, is the difference between the distance of x and y in the original graph and the distance of the supernodes that x and y belongs to in the coarse graph.

$$Err(x, y) = \left| d(x, y) - d'(x, y) \right| , x \neq y \quad (2)$$

Definition 5. Normalizing $Err(x, y)$ with $d(x, y)$, we have the *relative error* of nodes x and y.

$$RErr(x, y) = \frac{|d(x, y) - d'(x, y)|}{d(x, y)} , x \neq y \quad (3)$$

where $RErr(x, y)$ denotes the relative error.

Definition 6. Given G and G', the compression ratio of the coarsening stage is defined as $CR(G') = \frac{|V'|}{|V|}$. The number of edges is not included in the definition.

Problem: Given the original graph G and a compression ratio CR, $0 < CR < 1$, how to define G' such that the sum of the errors is minimum over all pairs. Specifically, the cost function that should be minimized is as follow:

$$\sum_{x,y \in V} Err(x, y) , x \neq y \quad (4)$$

Download English Version:

https://daneshyari.com/en/article/4945103

Download Persian Version:

https://daneshyari.com/article/4945103

Daneshyari.com