Expert Systems With Applications 67 (2017) 1–11

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Refreshment of the shortest path cache with change of single edge

Xiaohua Li[a], Tao Qiu[a], Ning Wang[b,*], Xiaochun Yang[a], Bin Wang[a], Ge Yu[a]

[a] School of Computer Science and Engineering, Northeastern University, Liaoning, China
[b] Department of Information Management, School of Management, Shanghai University, Shanghai, China

ARTICLE INFO

Article history:
Received 1 December 2014
Revised 30 July 2016
Accepted 2 August 2016
Available online 3 August 2016

Keywords:
Shortest path caching
Cache refreshment strategy
Change of road network
Affected shortest paths

ABSTRACT

The problem of caching shortest paths has been widely studied. All of existing methods that address this problem assume that the condition of road networks does not change with time. In this paper, we study how to refresh a cache when one edge of the underlying road network (graph) changes. A bitmap-based cache structure is proposed to store and give access to shortest paths. In the following, algorithms are developed to detect shortest paths that are affected by the change of edge. After detecting affected paths, several heuristic-based refreshment strategies are proposed to update the cache. We have conducted a series of experiments to compare the performance of proposed strategies. It shows that replacing affected shortest paths with new paths whose benefit values are the largest should be applied in the shortest path caching applications such as navigation and map services.

1. Introduction

Shortest path query systems are now widely seen in our daily life (Cheng, Ke, Chu, & Cheng, 2012; Liu & Yang, 2011; Potamias, Bonchi, Castillo, & Gionis, 2009; Wei, 2010; Wu et al., 2012). People use map services for shortest paths to restaurants, parking lots, cinemas, playgrounds, shopping malls and so on. With the uberisation, drivers of private cars also frequently use map services for various destinations. When a user has a shortest path query, a cache in a local server is accessed; the result, if existing, will be directly returned to the user. If the cache does not have an off-the-shelf result to the query, the system accesses a global server which runs a shortest path algorithm for the query. The latter situation undoubtedly costs communication and computational time (Altingovde, Ozcan, & Ulusoy, 2009; Baeza-Yates et al., 2007; Gan & Suel, 2009; Kriegel, Kroger, Renz, & Schmidt, 2008; Markatos, 2001). The cached contents are, therefore, very critical for the efficiency of the whole caching system.

In the shortest path caching problem, all of existing works discuss the cache initialization problem which addresses how to load an empty cache when a road network and a query log are known, so that the cache works in a high level of efficiency in the future. Researchers usually load paths with the highest query frequencies into the cache, or those paths that contain the most numbers of nodes (Li, Wang, Yang, Wang, & Yu, 2013; Thomsen, Yiu, & Jensen, 2012). However, existing works do not discuss how to refresh the cache when the road condition or characteristic of queries change. Actually, the road condition cannot remain the same all the time; on contrast, it changes frequently. For example, traffic congestion or a traffic accident affects the traffic capability of a road. If the cached paths are not adjusted after the condition of a road network changes, some paths in the cache may become invalid or the utilization of the cache decreases. How to maintain a cache after changes of a road network is seldom discussed in the literature. Apparently, a straightforward way to refresh a cache is to empty the cache and conduct an initialization process once more. Such a method is naive, because computing and evaluating all the shortest paths once again are time-consuming.

In this paper, we discuss how to refresh a cache when the weight of one single edge changes. Assuming that only one edge changes its weight is reasonable. On one hand, studying one single edge is a good point to set about. Some generic conclusions can be drawn from the single-edge scenario and be applied to multi-edge scenarios. On the other hand, single-edge does have real applications. For example, the traffic congestion may happen in main streets and these main streets are far away from each other and unrelated; the congestion in every main street can be considered separately as a single-edge scenario.

We first develop a cache structure composed of a path array and a bitmap. We then introduce the concept of "affected paths" and develop algorithms to detect affected shortest paths due to the change of edge weight. Next, we design four strategies to refresh the cache and compare their performance. Our work contributes

* Corresponding author.
E-mail addresses: lixiaohua@mail.neu.edu.cn (X. Li), qiutaoneu@gmail.com (T. Qiu), ningwang@shu.edu.cn (N. Wang), yangxc@mail.neu.edu.cn (X. Yang), binwang@mail.neu.edu.cn (B. Wang), yuge@ise.neu.edu.cn (G. Yu).

http://dx.doi.org/10.1016/j.eswa.2016.08.011
0957-4174/

to the literature in two aspects. First, to the best of our knowledge, our work is the first one to discuss the shortest path caching problem in a changing graph. This problem is undoubtedly more close to real applications. Second, four cache refreshment strategies are proposed to update the cache. It is noteworthy that this work is an extension of Li, Qiu, Yang, Wang, and Yu (2014) which is published in a conference. This paper extends the initial work, by (1) adopting a bitmap index to efficiently answer queries of shortest paths; (2) introducing several optimization techniques to further improve the efficiency of detecting affected shortest paths; (3) performing more comprehensive experiments on real data sets, including Aalborg, Beijing and Singapore road networks. Especially, the data set of Singapore is newly added and is not used in the conference version.

The rest of the paper is organized as follows. In Section 2, we review works related to the shortest path caching problem. In Section 3, we formally define the problem and design a bitmap-based cache structure to store shortest paths. Then we explore the properties of changed graphs and present our algorithm of detecting affected shortest paths in Section 4. In the following, four cache refreshment strategies are proposed in Section 5. In Section 6, we conduct experimental comparisons of the proposed methods on real data sets. Finally, we give closing remarks in Section 7.

2. Literature review

Our problem is related to works of two streams. One stream is to detect which shortest paths are changed after the graph changes. We call this kind of problem as "affected shortest path detection" problem. The other stream addresses the problem of caching shortest paths. In the following, we introduce concrete works in the two streams.

2.1. Affected shortest path detection

The shortest path of a certain node-pair may change after the weights of some edges change in a graph. The affected shortest path detection problem deals with finding node-pairs whose shortest paths are affected and update new shortest paths for these node-pairs. Based on whether the network changes in a regular manner, it has predictable and unpredictable networks.

Predictable network models aim at finding the expected fastest (shortest) paths based on the known distribution of predicted networks. In details, Kanoulas, Du, Xia, and Zhang (2006) developed the concept of speed pattern to specify the predicted network costs (weights) at different time, and the fastest paths corresponding to different time can be computed based on the predicted network cost. Fu and Rilett (1998) modeled the cost of each edge as a continuous stochastic process, and then they found the expected fastest paths. Similarly, Ding, Yu, and Qin (2008) used time-delay functions to represent the predicted edge costs.

Unpredictable network models assume that the costs of graph change randomly and cannot be depicted by a statistical distribution. Unpredictable network problem has offline and online versions.

For the offline research, it has pre-processing and needs extra space to store auxiliary information. In Tian, Lee, and Lee (2009), an index named QSI was used to identify the shortest paths affected by the change of network. QSI keeps the subnetworks that are involved in derivation of answering shortest paths. In Lee, Wu, and Chen (2007), in order to process more than one shortest path query simultaneously, a grid-based index structure was designed to record all the affecting areas. Besides, arc-flags is a typical index-based approach to compute the shortest paths. It has been adopted by Lauther (2004) for a static network. Berrettini, D'Angelo, and Delling (2009); D'Angelo, D'Emidio, and Frigioni (2014) studied the problem of updating arc-flags in dynamic networks.

For the online research, Bauer and Wagner (2009); D'Andrea, D'Emidio, Frigioni, Leucci, and Proietti (2013); Frigioni, Marchetti-Spaccamela, and Nanni (1996); McQuillan, Richer, and Rosen (1980); Narváez, Siu, and Tzeng (2000) studied how to update the shortest paths starting from a specified node. In general, these algorithms preprocess the shortest paths starting from a given source node, and store them in a shortest path tree (SPT). To update the SPT, the basic idea is to utilize the information of the outdated SPT and update only the part of the SPT that is affected by the changed edge. Lee et al. (2007) studied the problem of continuous evaluating shortest path queries. For several queries, they developed an ellipse bound method (EBM) where each shortest path corresponds to an elliptic geographical area, called affected area, and all the updated edges are considered to affect their corresponding paths.

It can be seen from the above review, offline study of shortest path detection problem uses extra space. For the online problem, some research maintain a novel designed data structure or they can only detect known paths. In our problem, the space is reserved for the cache; it cannot tell which paths are affected (or no longer the shortest) before the graph changes. Thus, extant algorithms for detecting affected shortest paths cannot be used in our problem directly.

2.2. Caching problem

Another steam is the caching problem. Caching techniques have been studied extensively in many domains (Markatos, 2001; O'Neil, O'Neil, & Weikum, 1993). The main focus is to decide the contents that are cached.

Web search caching. In Markatos (2001), caching strategies are classified into static caching and dynamic caching. Static caching does not change the contents of cache once the cache is loaded based on the historical log (Altingovde et al., 2009; Baeza-Yates et al., 2007; Baeza-Yates & Saint-Jean, 2003). The static caching usually updates periodically based on the latest query log. Dynamic caching focuses on suiting for new arrival queries and weeding out old contents (Gan & Suel, 2009; Long & Suel, 2005; Markatos, 2001).

Shortest path caching. Li et al. (2013); Thomsen et al. (2012); 2014) have studied the caching of shortest paths. In details, Thomsen et al. (2012) is the pioneer work on caching of shortest paths. They proposed a cost-oriented model to quantify the benefit of loading paths to a cache. Based on this model, a static caching strategy is proposed to initialize the cache. Li et al. (2013) also focuses on a static strategy. In order to store as many paths as possible, an improved cost-oriented model was proposed to measure paths appeared in the query log. Thomsen, Yiu, and Jensen (2014) proposed the notion of generically concise shortest paths that enables a trade-off between the size of paths and the number of queries answered by paths. They considered both the static and dynamic caching strategies for selecting generically concise paths. For the dynamic strategy, they update the cache in a least-recently-used policy when missing targets occur.

In applications of caching problem such as web search caching, queries change with time while the data (answers) will change in no way. The inefficiency of cache is fully resulted from the change of queries. Hence, "dynamic" caching strategies are motivated by the change of queries. However, in the shortest path caching problem, the change of graph (data) may also lead to the incorrectness of the answer and inefficiency of the cache. In this paper, we investigate the problem of shortest path caching when the graph