2018 17th International Symposium on Distributed Computing and Applications for Business Engineering and Science

More Accurate Estimation of Shortest Paths in Social Networks

Chaobing Feng, Ting Deng
Beijing Advanced Innovation Center for Big Data and Brain Computing
School of Computer Science and Engineering, Beihang University
Beijing, China
fengcb, dengting@act.buaa.edu.cn

***Abstract*—The shortest distance query between two given nodes is a fundamental but critical operation over social networks. Due to the computational efficiency of accurate methods is too low to be adopted for large-scale networks, recent researches about the shortest distance estimation mainly focus on approximate methods, in particular using landmark-based indexing strategy. A proper balance between computation rate and precision is greatly needed as the error of conventional landmark-based strategy is intolerable.**

In this paper, we analyze the deficiency in existing landmark embedding approaches. This paper mainly presents Local Subgraph Query (LSQ) algorithm for calculating the shortest path by dint of landmark embedding information. Besides, we propose an improved version of LSQ, termed Batch Subgraph Query (BSQ) algorithm, by aggregating landmarks and query nodes to raise accuracy. Experimental results on real-world datasets show that our methods outperform most of the state-of-the-art algorithms with significant estimation error decrease and few time penalty increases in social networks.

***Keywords*-shortest distances; query optimization; social networks;**

I. Introduction

Computing the shortest distance between given pair of nodes is a fundamental and frequently-used operation in graph algorithmics. With the rapid growth of social networks scale, the classical algorithms like BFS, Dijkstra [6] are no longer applicable to online query. Thus, some researchers divert their's attentions to more efficient algorithms such as A* [7], which relies on a heuristic approach to find the shortest path. To achieve higher computing speed, landmark-based [8] approaches have come in sight.

Based on the initial landmark-based algorithms, later studies propose certain improvement strategies correspondingly, such as graph pruning [9], compression [10] and so on. However, the accuracy of above optimization methods is still passable. By analysising the deficiencies of those methods, we propose several improved measures at different modules of the landmark-based method.

Challenges. The core challenges to the shortest path query can be attributed to time and space consumption. For the precomputation stage, the selection of landmarks has been proved to be a NP-hard problem [2], so we ought to devise an efficient selection measure. Then, the improvement to estimation accuracy should not be achieved at the expense of a high increase in the indexing complexity of landmark embedding. Besides, the complexity of the algorithm itself should be concise, or it will be too complicated to apply for different computing scene.

Contributions. Aiming at above challenges, the main contributions of this paper are summarized below: 1) We devise an outstanding landmarks selection scheme with consideration of the degree and distribution of nodes. 2) Based on landmarks set, we put forward LSQ to get more reliable approximate estimation. 3) We also propose improved schemes, termed as BSQ, by aggregating landmarks and query pairs for building a more sufficient subgraph compared to LSQ. 4) We design a mount of contrast experiments to check out the effectiveness of our plan.

Organization. The rest of the paper is organized as follows. In Section 2, we introduce relevant literature and related work. In Section 3, we describe the problems and give the related definition used in this paper. In Section 4, we propose several novel algorithms, including landmark selection, LSQ and BSQ. In Section 5, we perform extensive experiments on large-scale social networks and analyze the experimental results. Conclusions are reported in Section 6.

II. Related work

Landmark embedding techniques have been widely used to estimate distance in the social network [1]. The related works about methods mainly include the selection of landmarks in precomputation stage and the optimization of query algorithms. Most landmark-based methods adopt a random selection strategy, while others like [2], [3] take heuristic strategy.

Qiao et al. [4], [5] propose a novel shortest path tree based local landmark scheme and some optimization techniques, e.g. landmark indexes and graph compression. Greco et al. [11] propose some novel efficient incremental algorithms working both in main memory and disk. Akiba et al. [13], [14] propose a new exact method for dynamic shortest path queries on large-scale networks by pruning.

III. Preliminary

Network Graph. Let $G = (V, E)$ denote a graph with $|V|$ nodes and $|E|$ edges. For simplicity, we consider G is unweighted and undirected social network in the paper, but all the ideas can be easily applied to other weighted and/or directed graphs.

The Shortest Paths and Distances. Given a pair of nodes(u, v), denoted as $pair(u, v)$, and our purpose is to compute one of the shortest paths of $pair(u, v)$. We denote the exact shortest path as $sp(u, v)$ in graph. Meanwhile, let $\widetilde{sp}(u, v)$ denote the estimation of $pair(u, v)$ calculated by approximation methods.

Path Approximation. Given path $p < u_0, u_1, ..., u_{d_p} >$, $q < u_{d_p}, u_{d_p+1}, ..., u_{d_p+d_q} >$ with distance d_p, d_q respectively. Then concatenating two paths as path pq through nodes u_{d_p}, and the distance of pq is $d_p + d_q$. The initial approximation methods replace $sp(u, v)$ with $\widetilde{sp}(u, v)$ and the approximation $error(u, v)$ of which is denoted by:

$$error(u, v) = \frac{d_{\widetilde{sp}} - d_{sp}}{d_{sp}}, d_{sp} \le d_{\widetilde{sp}}. \quad (1)$$

Basic Landmark Embedding Algorithm. Before the query, we should precompute the landmark-embedding, which composed of the shortest paths and related distances, to support triangle inequality for calculating $\widetilde{sp}$. The number of landmarks, record as k, is determined by the demand of accuracy, due to the positive correlation between k and approximate precision. Based on landmark-embedding, denoted as $l_1, l_2, ..., l_k$, the calculation process of Basic Landmark Embedding(BLE) Algorithm for an arbitrary $pair(u_0, v_0)$ is shown below: 1) Compute the estimation through landmark l_1, denoted as $\widetilde{sp}_{l_1}(u_0, v_0)$, and get the estimation of each landmark in the same. 2) Compute the minimum estimation from $\widetilde{sp}_{l_1}(u_0, v_0)$ to $\widetilde{sp}_{l_k}(u_0, v_0)$ as estimation of $pair(u_0, v_0)$.

Problem Statement. Given a graph G, sometimes it makes sense to compute all/any pairwise distances, the corresponding algorithms are typically referred to as *APSP*. In order to facilitate the experimental analysis, we assign the k to 100 uniformly.

IV. Subgraph query scheme

In this section, we introduce a novel landmark-based shortest paths query algorithms. It consists of two major stages: 1) Landmark selection and precomputation to get landmark-embedding. 2) Approximate shortest path query.

A. Landmark Selection

The landmark selection of initial landmark-based algorithms is a random selection, also known as global selection, whose approximate error is up to 50%. For all we know, the landmark selection methods based on *degree* couldn't do better than *centrality* metrics, whereas the computation complexity of excellent metrics is too high to be used in the total graph. This suggests us to choose a graphic partitioning method aiming to solve above problem. We select the partitioning method named *METIS* [12] though it's higher computing efficiency, lower space requirements and better quality partitions. The basic idea behind METIS is the multilevel graph partition algorithm, which first coarsens graph down to a few hundred vertices, a bisection of this much smaller graph is computed, and then this partition is projected back towards the original graph, by periodically refining the partition.

Our landmark selection method first uses METIS to cut total graph. In each subgraph G_i, we select the node with the highest priority as landmark l_i. The priority formula for each node u_0 is:

$$Prio_{G_i}(u_0) = \frac{Degree_G(u_0)}{AvgDis_{G_i}(u_0)} \quad (2)$$

where the numerator indicates the degree of u_0 at G, the denominator indicates the average distance from u_0 to others at G_i. Compared to the metric of the whole graph, our scheme can avoid the landmarks gathering together. The dense landmarks distribution would lead to even worse estimation in most cases because of lower coverage.

B. Local Subgraph Query Algorithm

In this subsection, we analyse the defects of existing landmark-based algorithm and propose *LSQ*.

Here we introduce a scenario to demonstrate the computation process of A, which has been introduced in section 3. As illustrated in Figure 1, solid node l_1 is a landmark; u_0 and v_0 are query nodes, recorded as $pair(u_0, v_0)$; the path $< l_1, l_1', b, a, u_0 >$ and $< l_1, l_1', c, v_0 >$ are the shortest paths of $pair(l_1, u_0)$ and $pair(l_1, v_0)$ separately, also known as landmark embedding. The solid links in figure, like $< a, b >$ and $< b, c >$, represent real unweighted edges; otherwise, the dotted links such as $< u_0, a >$ represent simplified edges which leave out several unnecessary nodes between endpoint of the links. According to BLE, its estimation $\widetilde{sp}_{l_1}(u_0, v_0)$ would be $< 2 + 1 + 1 + \mathbf{2} + \mathbf{2} + 3 + 2 = 13 >$, which corresponding path $< u_0, a, b, l_1', l_1, l_1', c, v_0 >$.

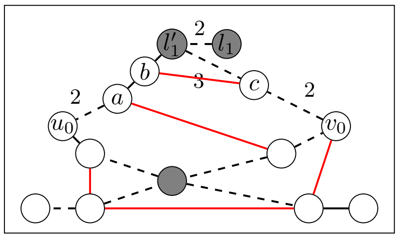


Figure 1: Shortest Path Query Graph for LSQ

It's easy to see the link$< l_1, l_1' >$ has been counted twice, causing a non-ignorable approximate error. As a consequence, many improvements use *LCA* to eliminate repeating edges like $< l_1, l_1' >$. Note that the LCA of $pair(u_0, v_0)$ in l_1-rooted tree is node l_1'. Compared to landmark l_1, the l_1' is a more query-dependent landmark of $pair(u_0, v_0)$ and the estimation of l_1', denoted as $\widetilde{sp}_{l_1'}(u_0, v_0)$, is more exact that of landmark l_1:

$$\widetilde{sp}_{l_1'}(u_0, v_0) \le \widetilde{sp}_{l_1}(u_0, v_0) \quad or \quad \widetilde{sp}_{l_1'}(u_0, v_0) = \widetilde{sp}_{l_1}(u_0, v_0) - \mathbf{2} * d_{<l_1, l_1'>} \quad (3)$$

Actually, we could get more accurate paths by leading excess links between nodes on landmark-based paths except links located on landmark-based paths. That is to say, we build a graph, denoted as $G_l(u, v)$ consist of nodes on landmark-based paths. The $G_l(u, v)$ is *maximum common subgraph* between G and the *fully connected graph* of nodes on landmark-based paths $p_{u,l}$, $p_{v,l}$. As illustrated in Figure 1, the excess link of $G_{l_1}(u_0, v_0)$ is $< b, c >$, as well as the approximate path correspondingly become $< u_0, a, b, c, v_0 >$, whose distance is $< 2+1+1+2 = 6 >$.

Compared to path $< u_0, ..., l_1, ..., v_0 >$, whose distance is 13, the approximate path of LSQ has shortened half the distance. The LSQ is depicted in Algorithm 1, where the purpose of function $buildLocalGraph(G, S)$ is to obtain the maximum complete subgraph of graph G and node-set S. The function $getSssp(G, u, v)$ can get single-source shortest path for pair (u, v) in graph G, which implemented by BFS to unweighted graph and $Dijkstra$ to weighted graph under normal conditions.

Algorithm 1: LSQ-Local Subgraph Query Algorithm

```
Input: A query pair $(u_0, v_0)$
Output: $\widetilde{sp}(u_0, v_0)$-The shortest path estimation of $(u_0, v_0)$
1  $\widetilde{sp}(u_0, v_0) := \infty$;
2  for $l_i \in$ landmarkSet do
3      Load node-set $S_{l_i}(u_0, v_0)$ on path embedding from disk;
4      $G_{l_i}(u_0, v_0) := buildLocalGraph(G, S_{l_i}(u_0, v_0))$;
5      $\widetilde{sp}_{l_i}(u_0, v_0) := getSssp(G_{l_i}(u_0, v_0), u_0, v_0)$;
6      if $\widetilde{sp}_{l_i}(u_0, v_0) < \widetilde{sp}(u_0, v_0)$ then
7          $\widetilde{sp}(u_0, v_0) := \widetilde{sp}_{l_i}(u_0, v_0)$;
8      end
9  end
10 return $\widetilde{sp}(u_0, v_0)$
```

C. Batch Subgraph Query Algorithm

In order to further improve the effectiveness of this method, we propose BSQ algorithm, which ensemble distinct query pairs and landmarks to build a subgraph, used for processing query pairs involved in this batch subgraph.

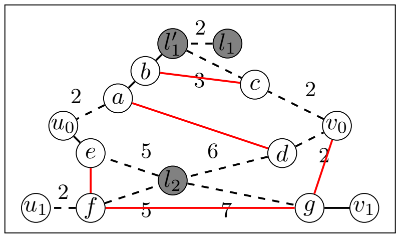


Figure 2: Shortest Path Query Graph for BSQ

For instance, we consider the ensemble batch scene of two landmarks and two query pairs, whose process mode is analogical with the case with more landmarks and query pairs. The batch subgraph produced by $pair(u_0, v_0)$, (u_1, v_1) in Figure 2, denoted as $G_L < (u_0, v_0), (u_1, v_1) >$, combined by $G_L(u_0, v_0)$ and $G_L(u_1, v_1)$. The union graph of two graphs can be understood as the union of nodes and edges set of them. Obviously, the full graph in Figure 2 is exactly the batch subgraph of $pair(u_0, v_0)$ and (u_1, v_1), just take landmarks l_1 and l_2 into total landmarks set L. As illustrated in Figure 2, compared to local subgraph $G_{l_i}(u_0, v_0)$, the batch subgraph $G_L < (u_0, v_0), (u_1, v_1) >$ introduces more helpful links as $< e, f >$, $< f, g >$ and $< g, v_0 >$ for querying $pair(u_0, v_0)$. As a result, the approximate shortest path of $pair(u_0, v_0)$ change to be $< u_0, e, f, g, v_0 >$, whose distance is $< 1+1+1+1 = 4 >$. The BSQ is depicted in Algorithm 3.

It is obvious that each local subgraph of LSQ is a subset of BSQ's batch subgraph, which ensures the accuracy of BSQ must be optimal. Although batch subgraph is even bigger than local subgraph, but the former avoids calculating subgraph time and again for different query pairs. Thanks to the diameter of social networks is ordinarily short, the extra expenses compared with BLE can be ignored in view of the advance of accuracy.

Algorithm 2: BSQ-Batch Subgraph Query Algorithm

```
Input: A query pair set P, which consist of k query pairs : $(u_1, v_1)$, $(u_2, v_2)$,..., $(u_k, v_k)$
Output: $\widetilde{sp}$-The shortest path estimation of $P$
1  $S_L P := \varnothing$;
2  for $l_i \in$ landmarkSet do
3      for $(u_j, v_j) \in P$ do
4          Load node-set $S_{l_i}(u_j, v_j)$ on path embedding from disk;
5          $S_L P := S_L P \bigcup S_{l_i}(u_j, v_j)$
6      end
7  end
8  $G_L P := buildLocalGraph(G, S_L P)$;
9  for $(u_i, v_i) \in P$ do
10     $\widetilde{sp}(u_i, v_i) := getSssp(G_L P, u_i, v_i)$;
11 end
12 return $\widetilde{sp}$
```

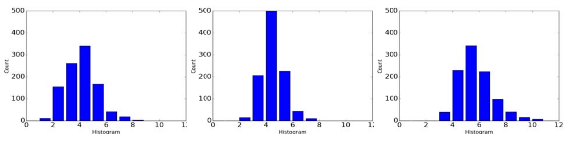
V. Experiments

In this section, we conduct comprehensive experiments to evaluate the performance of the proposed algorithms. All algorithms were implemented in $Java$ and tested on an $Ubuntu$ server using one 3.40 GHz CPU and 8 GB memory.

A. Dataset Description

Table I: NETWORK STATISTICS

Dataset	$\lvert V \rvert$	$\lvert E \rvert$	D	d'
Facebook	4,039	88,234	8	4.7
Slashdot	82,168	948,464	11	4.7
YouTube	1,134,890	2,987,624	20	6.5



(a) Facebook (b) Slashdot (c) YouTube

Figure 3: Shortest Distance Histogram

Facebook[1]**.** Facebook data was collected from survey participants using Facebook app.

Slashdot[2]**.** Slashdot is a technology-related news website introduced in 2002, where users can tag each other as friends or foes.

YouTube[3]**.** A YouTube video-sharing graph of over 1 million users and almost 3 million links that include a social network.

[1]http://snap.stanford.edu/data/egonets-Facebook.html
[2]http://snap.stanford.edu/data/soc-Slashdot0902.html
[3]http://snap.stanford.edu/data/com-YouTube.html

The test networks and their properties are listed in table 1. The D and d' are diameter and 90-percentile effective diameter separately. Figure 3 represents the histogram of the actual shortest distance distribution over the 1,000 query pairs in each dataset.

B. Evaluation Metrics

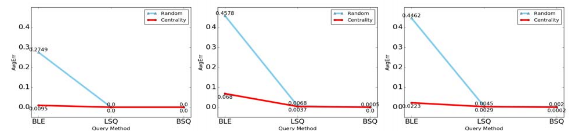
It is expensive to exhaustively test all pairs in the dataset, so we randomly sample 1000 pairs and compute the average relative error ($AvgErr$) on the sample set. The formula for computing $AvgErr$ is:

$$\overline{error} = \frac{\sum error(u, v)}{1000} \qquad (4)$$

Besides, the query processing time per pair and the embedding index size are also within the scope of consideration.

C. Experimental Results

We have implemented three different query methods, including BLE, LSQ and BSQ, under identical dataset, query pair set and landmark number(k=100). The query pair sets are selected by random and priority(proposed in subsection 4.1)-based ways, expressed as Random and Centrality separately. Figure 4 shows the average error and table 2 shows the number of correct query pairs results. Experimental results show: 1) Our LSQ obtains a significant increase in accuracy compared to BLE; From LSQ to BSQ, the test effect has risen steadily, along with nearly all the correct solutions are obtained. 2) Our landmark selection scheme is completely better than random way. The Centrality landmark selection is especially efficient for BLE. On the other hand, the LSQ and BSQ have better robustness for different landmark sets.



(a) Facebook (b) Slashdot (c) YouTube

Figure 4: The average error under different query method of each dataset

Table II: CORRECT PAIRS NO.

		Facebook	Slashdot	YouTube
Random	BLE	246	24	1
	LSQ	1000	976	978
	BSQ	1000	998	991
Centrality	BLE	990	767	898
	LSQ	1000	986	986
	BSQ	1000	1000	999

D. Comparison with Other Query Algorithms

As we know, the TreeSketch algorithm proposed in [1] prominently outperforms the others in the accuracy of approximate query, so we compare our BSQ with TreeSketch. We uniformity set k to 100, landmark selection to Centrality and test dataset to YouTube for a fair comparison. The comparison result are illustrated in table 4. By comparison, we can learn BSQ outperforms TreeSketch in each evaluation index. Beyond that, LSQ needs same Index Size with BSQ but can query faster than the other two methods. Thus we can see that if we pay more attention to accuracy, we will choose BSQ instead of LSQ for faster query efficiency.

Table III: COMPARISON RESULTS

Result	$\overline{error}$	Query Time(ms)	Index Size(GB)
TreeSketch	0.0006	25.84	10.2
LSQ	0.0029	0.53	5.9
BSQ	0.0002	21.52	5.9

VI. Conclusion

In this paper, we first present a landmark selection method which is largely better than initial random ways for social networks. Next, a series of subgraph query algorithm are proposed, known as LSQ and BSQ for short, which get a significant increase in the approximation accuracy, yet with few time and space consumption thanks to the very short diameter of social networks. Extensive experimental results demonstrate the accuracy and effectiveness of our shortest path query scheme.

Acknowledgment

Feng and Deng are supported in part by NSFC 61133002 and 61602023.

References

[1] A. Gubichev, S. Bedathur, S. Seufert, and G. Weikum, "Fast and accurate estimation of shortest paths in large graphs," in *Proceedings of the 19th ACM international conference on Information and knowledge management*. ACM, 2010, pp. 499–508.

[2] M. Potamias, F. Bonchi, C. Castillo, and A. Gionis, "Fast shortest path distance estimation in large networks," in *Proceedings of the 18th ACM conference on Information and knowledge management*. ACM, 2009, pp. 867–876.

[3] Z. Qi, Y. Xiao, B. Shao, and H. Wang, "Toward a distance oracle for billion-node graphs," *Proceedings of the VLDB Endowment*, vol. 7, no. 1, pp. 61–72, 2013.

[4] M. Qiao, H. Cheng, L. Chang, and J. X. Yu, "Approximate shortest distance computing: A query-dependent local landmark scheme," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 1, pp. 55–68, 2014.

[5] M. Qiao, H. Cheng, and J. X. Yu, "Querying shortest path distance with bounded errors in large graphs," in *International Conference on Scientific and Statistical Database Management*. Springer, 2011, pp. 255–273.

[6] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[7] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[8] A. V. Goldberg and R. F. F. Werneck, "Computing point-to-point shortest paths from external memory." in *ALENEX/ANALCO*, 2005, pp. 26–40.

[9] A. V. Goldberg, H. Kaplan, and R. F. Werneck, "Reach for a*: Efficient point-to-point shortest path algorithms," in *2006 Proceedings of the Eighth Workshop on Algorithm Engineering and Experiments (ALENEX)*. SIAM, 2006, pp. 129–143.

[10] R. Jin, Y. Xiang, N. Ruan, and D. Fuhry, "3-hop: a high-compression indexing scheme for reachability query," in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*. ACM, 2009, pp. 813–826.

[11] S. Greco, C. Molinaro, and C. Pulice, "Efficient maintenance of all-pairs shortest distances," in *Proceedings of the 28th International Conference on Scientific and Statistical Database Management*. ACM, 2016, p. 9.

[12] G. Karypis and V. Kumar, "Metis – unstructured graph partitioning and sparse matrix ordering system, version 2.0," Tech. Rep., 1995.

[13] T. Akiba, Y. Iwata, and Y. Yoshida, "Fast exact shortest-path distance queries on large networks by pruned landmark labeling," in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. ACM, 2013, pp. 349–360.

[14] T. Akiba, Y. Iwata, and Y. Yoshida, "Dynamic and historical shortest-path distance queries on large evolving networks by pruned landmark labeling," in *Proceedings of the 23rd international conference on World wide web*. ACM, 2014, pp. 237–248.