More Accurate Estimation of Shortest Paths in Social Networks

Chaobing Feng, Ting Deng
Beijing Advanced Innovation Center for Big Data and Brain Computing, 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, rencent 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 approachs. This paper mainly presents Local Subgraph Query (*LSQ*) algorithm for calculating shortest path by dint of landmark embedding information. Besides, we propose improved version of *LSQ*, termed Ensemble Subgraph Query (*ESQ*) and Batch Subgraph Query (*BSQ*) algorithm, by aggregating landmarks and query nodes to raise accuracy. Experimental results on real-world datasets show that our methods outperforms most of the state-of-the-art algorithms with significant estimation error decrease and few time penalty increases in social networks.

CCS CONCEPTS

• **Theory of computation → Shortest paths**; *Approximation algorithms analysis*;

KEYWORDS

Shortest Distances, Query Optimization, Social Networks

ACM Reference Format:
Chaobing Feng, Ting Deng. 2018. More Accurate Estimation of Shortest Paths in Social Networks. In *Proceedings of International Conference on Scientific and Statistical Database Management (SSDBM'18).* Bozen-Bolzano, Italy, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 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 With the rapid growth of social networks scale, the classical algorithms like BFS, Dijkstra [4] are no longer applicable to online query. Thus, some researchers divert their's attentions to more efficient algorithms such as A* [10], which relys on a heuristic approach to find the shortest path. To achieve higher computing speed, landmark-based approaches [7] have come in sight. In the family of this technique, k nodes are selected as landmark in the beginning. Afterwards the distances and the corresponding paths are precomputed from each node to some or all of the landmarks, used as landmark embedding.

Based on the initial landmark-based algorithms, later studies propose certain improvement strategies correspondingly, such as Lowest Common Ancestors (*LCA*) [19] of paths, graph pruning [6] or compression [11], 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 precalculation stage, the selection of landmarks has been proved to be an NP-hard problem [14], 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 enbedding. 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 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 *ESQ*/*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 checkout 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 present related definitions used in this paper. In Section 4, we propose novel algorithms, including *LSQ*, *ESQ*, *BSQ* and landmark selection. In Section 5, we perform extensive experiments on large-scale social networks and analyze the experimental results. Conclusions are reported in Section 6.

2 RELATED WORK

Landmark embedding techniques have been widely used to estimate distance in many applications, and our chief concern of which is related application of social network [3, 9, 18]. 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 random selection strategy, while others like [14, 15] take heuristic strategy. Potamias et al. [14] proposes an additional method that can compete with the highest degree approach thanks to several heuristics for landmark selection.

Qiao et al. [16, 17] propose a novel shortest path tree based local landmark scheme and some optimization techniques, e.g. landmark indexes and graph compression. Wei [20] propose TEDI, a

SSDBM'18, July 2018, Bozen-Bolzano, Italy

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

query processing scheme based on tree decomposition can be extended to different query algorithms. Greco et al. [8] propose some novel efficient incremental algorithms working both in main memory and disk. Akiba et al. [1, 2] propose a new exact method for dynamic shortest path queries on large-scale networks by pruning.

3 PRELIMINARY

Let $G = (V, E)$ denote a graph with $|V|$ nodes and $|E|$ edges.

Graph Background. 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. Let path $p =< u_0, u_1, ..., u_{len_p} >$, $q =< u_{len_p}, u_{len_p+1}, ..., u_{len_p+len_q} >$ and the lengths of them are len_p, len_q respectively. Then concating two paths as path pq through nodes u_{len_p}, and the length of pq is $len_p + len_q$. The initial approximation methods replace $sp(u, v)$ with $\widetilde{sp}(u, v)$, which is yet path pq, and the approximation $error(u, v)$ of this series path denoted by:

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

BLE-Basic Landmark Embedding Algorithm. Bofore 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 positive correlation between k and approximate precision. Based on landmark-embedding, denoted as $l_1, l_2, ..., l_k$, the calculation process of *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.

4 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.

4.1 Landmark Selection

The landmark selection of initial landmark-based algorithms is 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 total graph. This suggests us to choose a graphic partitioning methods aiming to solve above problem. We select the partitioning method named *METIS* [12] through it's higher computing efficiency, lower space requirements and better segmentation effect.

Our landmark selection method first use 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.

4.2 Local Subgraph Query Algorithm

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

The *BLE* has been introduced in section 3. As illustrated in Figure 1, node l_1 is 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 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 $< u_0, a, b, l_1', l_1, l_1', c, v_0 >$.

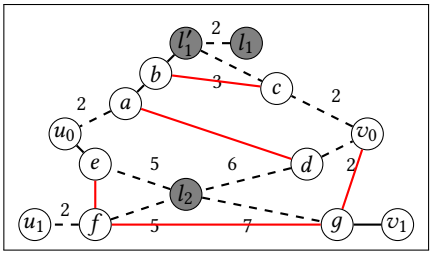


Figure 1: Shortest Path Query Graph

It's easy to see the link$< l_1, l_1' >$ has been counted twice, causing 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:

$$\widetilde{sp}_{l_1'}(u_0, v_0) \leq \widetilde{sp}_{l_1}(u_0, v_0) \quad or \quad \widetilde{sp}_{l_1'}(u_0, v_0) = \widetilde{sp}_{l_1}(u_0, v_0) - 2 * len_{<l_1, l_1'>}. \quad (3)$$

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 length is $< 2+1+1+2 = 6 >$. Compared to path $< u_0, ..., l_1, ..., v_0 >$, whose length is 13, the approximate path of *LSQ* has shortened half the length. The *LSQ* is depicted in Algorithm 1.

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)

$\widetilde{sp}(u_0, v_0) := \infty$;
for $l_i \in landmarkSet$ **do**
  Load node set $S_{l_i}(u_0, v_0)$ on path enbedding from disk;
  $G_{l_i}(u_0, v_0) := buildLocalGraph(G, S_{l_i}(u_0, v_0))$;
  $\widetilde{sp}_{l_i}(u_0, v_0) := getSssp(G_{l_i}(u_0, v_0), u_0, v_0)$;
  if $\widetilde{sp}_{l_i}(u_0, v_0) < \widetilde{sp}(u_0, v_0)$ **then**
    $\widetilde{sp}(u_0, v_0) := \widetilde{sp}_{l_i}(u_0, v_0)$;
  end
end
return $\widetilde{sp}(u_0, v_0)$

4.3 ESQ-Ensemble Subgraph Query Algorithm

In this subsection, the Algorithm 1 is improved by integrating more landmarks and query pairs together for building an ensemble subgraph, which performs a limited scope search that may find a shortcut with a smaller distance than *LSQ*.

As illustrated in Figure 1, except the query $pair(u_0, v_0)$ and landmark l_1, we introduce another landmark l_2, the corresponding path landmark embedding of which are $< l_2, e, u_0 >$ and $< l_2, d, v_0 >$; the landmark l_1 and l_2 compose the landmark set L. Simply, we just consider the ensemble scene of two landmarks, whose procedure is analogical with the case setting up more landmarks. The ensemble subgraph in Figure 1, denoted as $G_L(u_0, v_0)$, consist of extra nodes $< e, l_2, d, v_0 >$ compared to $G_{l_1}(u_0, v_0)$. Besides links on path landmark embedding of $< u_0, l_1 >, < v_0, l_1 >, < u_0, l_2 >$ and $< v_0, l_2 >$, the excess links of $G_L(u_0, v_0)$ are $< b, c >$ and $< a, d >$. By dint of ensemble subgraph, the approximate shortest path is becoming $< u_0, a, d, v_0 >$, whose length is $< 2 + 1 + 2 = 5 >$. The *ESQ* is depicted in Algorithm 2.

Algorithm 2: ESQ-Ensemble 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)

$S_L(u_0, v_0) := \varnothing$;
for $l_i \in landmarkSet$ **do**
  Load node set $S_{l_i}(u_0, v_0)$ on path enbedding from disk;
  $S_L(u_0, v_0) := S_L(u_0, v_0) \bigcup S_{l_i}(u_0, v_0)$
end
$G_L(u_0, v_0) := buildLocalGraph(G, S_L(u_0, v_0))$;
$\widetilde{sp}(u_0, v_0) := getSssp(G_L(u_0, v_0), u_0, v_0)$;
return $\widetilde{sp}(u_0, v_0)$

4.4 BSQ-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. 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 1, 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 1 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 1, compared ensemble subgraph $G_L(u_0, v_0)$, 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 length is $< 1 + 1 + 1 + 1 = 4 >$. The *BSQ* is depicted in Algorithm 3.

Algorithm 3: BSQ-Batch Subgraph Query Algorithm

Input: A query pair set P-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

$S_L P := \varnothing$;
for $l_i \in landmarkSet$ **do**
  for $(u_j, v_j) \in P$ **do**
    Load node set $S_{l_i}(u_j, v_j)$ on path enbedding from disk;
    $S_L P := S_L P \bigcup S_{l_i}(u_j, v_j)$
  end
end
$G_L P := buildLocalGraph(G, S_L P)$;
for $(u_i, v_i) \in P$ **do**
  $\widetilde{sp}(u_i, v_i) := getSssp(G_L P, u_i, v_i)$;
end
return $\widetilde{sp}$

Although batch subgraph is even bigger than ensemble 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.

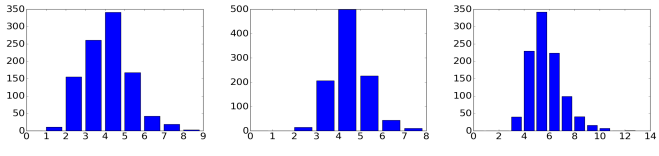
5 EXPERIMENTS

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

5.1 Dataset Description

Table 1: NETWORK STATISTICS

Dataset	|V|	|E|	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 2: Shortest Distance Histogram

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

[1] http://snap.stanford.edu/data/egonets-Facebook.html

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.

The test networks and their properties are listed in table 1. The D and d' are diameter and 90-percentile effective diameter separately. Figure 2 represents the histogram of the shortest distance distribution over the 1,000 query pairs in each dataset.

5.2 Evaluation Metrics

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

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

Besides, the query processing time per pair and the embedding index size are presented meanwhile.

5.3 Experimental Results

Table 2 shows the *AvgErr* and the number of correct query pairs of initial landmark-based, *LSQ*, *ESQ*, *BSQ* with landmark sets selected by Random and Centrality proposed in subsection 4.1 on the three social networks. Experimental results show: 1) Our *LSQ* obtains a significant increase in accuracy compared to initial landmark algorithm; From *LSQ* to *ESQ* 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 initial landmark algorithm. On the other hand, the *LSQ*, *ESQ* and *BSQ* have better robustness for different landmark sets.

Table 2: AVERAGE ERROR/CORRECT PAIRS NO.

		Facebook	Slashdot	Youtube
Random	Initial	0.2749/246	0.4578/24	0.4462/1
	LSQ	0.0/1000	0.0068/976	0.0045/978
	ESQ	0.0/1000	0.0028/990	0.0033/984
	BSQ	0.0/1000	0.0005/998	0.002/991
Centrality	Initial	0.0095/990	0.0680/767	0.0223/898
	LSQ	0.0/1000	0.0037/986	0.0029/986
	ESQ	0.0/1000	0.00058/998	0.00065/997
	BSQ	0.0/1000	0.0/1000	0.0002/999

5.4 Comparison with Other Query Algorithms

As we know, the TreeSketch algorithm proposed in [9] prominently outperforms the others, 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 contrast results is illustrated in table 4. By comparison we can learn *BSQ* outperforms TreeSketch in each evaluation index.

Table 3: COMPARISON RESULTS

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

[2]http://snap.stanford.edu/data/soc-Slashdot0902.html
[3]http://snap.stanford.edu/data/com-Youtube.html

6 CONCLUSIONS

In this paper, we first present a landmark selection method which is largely better than initial ways for social networks. Next, a series of subgraph query algorithm are proposed, known as *LSQ*/*ESQ*/*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 paths query scheme.

REFERENCES

[1] Takuya Akiba, Yoichi Iwata, and Yuichi Yoshida. 2013. 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, 349–360.

[2] Takuya Akiba, Yoichi Iwata, and Yuichi Yoshida. 2014. 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, 237–248.

[3] Atish Das Sarma, Sreenivas Gollapudi, Marc Najork, and Rina Panigrahy. 2010. A sketch-based distance oracle for web-scale graphs. In *Proceedings of the third ACM international conference on Web search and data mining*. ACM, 401–410.

[4] Edsger W Dijkstra. 1959. A note on two problems in connexion with graphs. *Numerische mathematik* 1, 1 (1959), 269–271.

[5] Simon Fear. 2005. *Publication quality tables in LaTeX*. http://www.ctan.org/pkg/booktabs.

[6] Andrew V Goldberg, Haim Kaplan, and Renato F Werneck. 2006. Reach for A*: Efficient point-to-point shortest path algorithms. In *2006 Proceedings of the Eighth Workshop on Algorithm Engineering and Experiments (ALENEX)*. SIAM, 129–143.

[7] Andrew V Goldberg and Renato Fonseca F Werneck. 2005. Computing Point-to-Point Shortest Paths from External Memory.. In *ALENEX/ANALCO*. 26–40.

[8] Sergio Greco, Cristian Molinaro, and Chiara Pulice. 2016. Efficient maintenance of all-pairs shortest distances. In *Proceedings of the 28th International Conference on Scientific and Statistical Database Management*. ACM, 9.

[9] Andrey Gubichev, Srikanta Bedathur, Stephan Seufert, and Gerhard Weikum. 2010. Fast and accurate estimation of shortest paths in large graphs. In *Proceedings of the 19th ACM international conference on Information and knowledge management*. ACM, 499–508.

[10] Peter E Hart, Nils J Nilsson, and Bertram Raphael. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics* 4, 2 (1968), 100–107.

[11] Ruoming Jin, Yang Xiang, Ning Ruan, and David Fuhry. 2009. 3-hop: a high-compression indexing scheme for reachability query. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*. ACM, 813–826.

[12] George Karypis and Vipin Kumar. 1995. *METIS – Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version 2.0*. Technical Report.

[13] Leslie Lamport. 1986. *LaTeX: A Document Preparation System*. Addison-Wesley, Reading, MA.

[14] Michalis Potamias, Francesco Bonchi, Carlos Castillo, and Aristides Gionis. 2009. Fast shortest path distance estimation in large networks. In *Proceedings of the 18th ACM conference on Information and knowledge management*. ACM, 867–876.

[15] Zichao Qi, Yanghua Xiao, Bin Shao, and Haixun Wang. 2013. Toward a distance oracle for billion-node graphs. *Proceedings of the VLDB Endowment* 7, 1 (2013), 61–72.

[16] Miao Qiao, Hong Cheng, Lijun Chang, and Jeffrey Xu Yu. 2014. Approximate shortest distance computing: A query-dependent local landmark scheme. *IEEE Transactions on Knowledge and Data Engineering* 26, 1 (2014), 55–68.

[17] Miao Qiao, Hong Cheng, and Jeffrey Xu Yu. 2011. Querying shortest path distance with bounded errors in large graphs. In *International Conference on Scientific and Statistical Database Management*. Springer, 255–273.

[18] Matthew J Rattigan, Marc Maier, and David Jensen. 2006. Using structure indices for efficient approximation of network properties. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 357–366.

[19] Konstantin Tretyakov, Abel Armas-Cervantes, Luciano García-Bañuelos, Jaak Vilo, and Marlon Dumas. 2011. Fast fully dynamic landmark-based estimation of shortest path distances in very large graphs. In *Proceedings of the 20th ACM international conference on Information and knowledge management*. ACM, 1785–1794.

[20] Fang Wei. 2010. TEDI: efficient shortest path query answering on graphs. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. ACM, 99–110.