Computers, Materials & Continua CMC, vol.63, no.2, pp.705-724, 2020

A Fast Method for Shortest-Path Cover Identification in Large Complex Networks

Qiang Wei[1, 2, *], Guangmin Hu[1], Chao Shen[3] and Yunfei Yin[4, 5]

Abstract: Fast identifying the amount of information that can be gained by measuring a network via shortest-paths is one of the fundamental problem for networks exploration and monitoring. However, the existing methods are time-consuming for even moderate-scale networks. In this paper, we present a method for fast shortest-path cover identification in both exact and approximate scenarios based on the relationship between the identification and the shortest distance queries. The effectiveness of the proposed method is validated through synthetic and real-world networks. The experimental results show that our method is 10^5 times faster than the existing methods and can solve the shortest-path cover identification in a few seconds for large-scale networks with millions of nodes and edges.

Keywords: Network discovery, shortest-path cover, shortest-path distance query, large complex networks.

1 Introduction

Network topology is a crucial prerequisite for understanding network performance and network security situation. In order to study complex networks such as the Internet, the World Wide Web or social networks, one has to explore or monitor them [Guillaume and Latapy (2005)]. Most of the complex networks rely on partial views obtained by using various intricate exploration method. Traceroute-like measuring is one of the main approaches and has been adopted by many topology discovery systems [McRobb, Claffy and Monk (1999); Spring, Mahajan and Wetherall (2004)]. For obtaining a view of the topology, traceroute-like approach merges routes from a given set of sources to destinations. To identify the amount of information by traceroute-like measuring, in this paper, we consider the problem of shortest-path cover identification (SPCI) which determines the proportion of edges that can be discovered via shortest-paths from given source nodes to all the other nodes. SPCI is a reasonable model for traceroute-like measuring [Ouédraogo and Magnien (2011)], meanwhile, SPCI is a precondition for optimal deploying monitor sources [Han and Xu

[1] School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu, 611731, China.

[2] National Key Laboratory of Science and Technology on Blind Signal Processing, Chengdu, 610041, China.

[3] MOE Key Lab for Intelligent Networks and Network Security, Xi'an Jiaotong University, Xi'an, 710049, China.

[4] The Electronic Engineering Department, Universidad de Sevilla, Seville, 41004, Spain.

[5] School of Astronautics, Harbin Institute of Technology, Harbin, 150001, China.

[*] Corresponding Author: Qiang Wei. Email: weiqiang@std.uestc.edu.cn.

Received: 23 May 2019; Accepted: 02 August 2019.

(2008); Zou, Qiao, Zhou et al. (2009)]; Boothe, Dvorák, Farley et al. (2007)].

The researchers have paid plenty of attention to SPCI. In theory, Blondel et al. [Blondel, Guillaume, Hendrickx et al. (2007)] quantitatively estimate the average SPCI from a single source in Erdös-Rényi networks. Boothe et al. [Boothe, Dvorák, Farley et al. (2007)] define two related deploying optimization problems based on SPCI and show the NP-complete hardness of the two problems on general networks. Dall'Asta et al. [Dall'Asta, Alvarez-Hamelin, Barrat et al. (2006)] point out that the probability of node and link detection is related to the betweenness centrality of the element and the density of sources and destinations. In practice, Guillaume et al. [Guillaume and Latapy (2005)] focus on the differences of SPCI under various source settings in Erdös-Rényi networks, and they show that the ratio amount varies and is difficult to estimate. Pignolet et al. [Pignolet, Schmid and Tredan (2017)] introduce an asymmetric model and give approximate algorithms for cactus and outer planar networks. Barford et al. [Barford, Bestavros, Byers et al. (2004)] quantify the marginal utility of SPCI, and they find that the marginal utility of adding sources quickly drops and low marginal utility which does not imply the overall coverage is high. Zou et al. [Zou, Qiao, Zhou et al. (2009)] and Han et al. [Han and Xu (2008)] develop heuristic methods to solve one of the deploying optimization problems. As a fundamental operation on networks, SPCI needs to be computed fast [Blondel, Guillaume, Hendrickx et al. (2007)], most of the previous works consider SPCI to be a known precondition, but it is still a challenge for large complex networks.

Fast SPCI is a non-trivial task in large complex networks [Sommer (2014)]. SPCI is related to all the shortest-paths between any two nodes in a network [Boothe, Dvorák, Farley et al. (2007)]. Most of the state-of-the-art algorithms for a single-source shortest-path or all-pairs shortest-path [Cormen, Leiserson, Rivest et al. (2001); Madkour, Aref, Rehman et al. (2017)] are aimed at outputting only one of the shortest-paths between any two nodes. However, it costs us impractically exponential time to pick out all the shortest-paths by using these algorithms directly, e.g., the time complexity of generating all the shortest-paths by Breadth First Search (BFS) is $O(2^{|V|})$.

To address this issue, this paper presents a method for fast SPCI in large complex networks. We prove that SPCI can be solved by shortest-path distance queries (SPDQ). After making the connection between SPCI and SPDQ, we propose a method for fast SPCI in both exact and approximate scenarios, and we get more than 10^5 times speed up than BFS. Furthermore, we develop two sampling strategies for further speed up SPCI. Combining SPDQ and network sampling, we can solve SPCI in a few seconds for networks with hundreds of millions of nodes with a small accuracy loss. Our method can not only be used to quickly evaluate the performance of the deployment schemes that is crucial for network monitoring, but also be used as a fundamental step in solving deploying optimization problems. In summary, our contributions are as follows:

- We show that SPCI can be solved by SPDQ instead of computing all the actual shortest-paths, and this conversion avoid the exponential time complexity.
- We propose a semi-approximate SPDQ method for SPCI in order to trade-off between exact SPDQ and approximate SPDQ, and the semi-approximate method speeds up the exact SPDQ with a small accuracy loss.

- Several network-sampling strategies are designed to further speed up SPCI. We show that SPCI can be estimated by using only a small portion of edges in a network. Network sampling strategies are independent with SPDQ methods, and therefore they can be combined together.
- We implement extensive experimental evaluation on both synthetic and real-world networks to verify the effectiveness of the proposed methods.

The paper is organized as follows: In Section 2, we define the SPCI problem, and show the connection between SPCI and SPDQ. This is followed in Section 3 with two speed up strategies including a semi-approximate SPDQ method and network sampling. Continuing in Section 4, we perform experiments on both synthetic and real-world networks and evaluate the effectiveness of our method. Finally, we conclude this paper in Section 5.

2 Methodology

2.1 Definition of the SPCI problem

In this section, we formally state the SPCI problem from [Boothe, Dvorák, Farley et al. (2007)]. Tab. 1 lists the notations that are frequently used in this paper. We model a network by a simple graph $G(V, E)$, where V is a set of nodes representing network sites and E is a set of edges representing links. For any two nodes $u, v \in V$, let $SP(u, v)$ be the set of all shortest-paths between u, v. For path $p \in SP(u, v)$, we denote $ES(p)$ as the edge set of p. Let R be the covering model that control which parts of edges can be discovered in $SP(u, v)$ and $C_R(u, v)$ be the covered edges under covering model R. We denote $C_R(u)$ as the edges covered by node u, that is, $C_R(u)=\cup_{v \in V\setminus\{u\}} C_R(u, v)$. And we denote $C_R(U)$ as the edges covered by node set U, that is, $C_R(U)=\cup_{u \in U} C_R(u)$. The two shortest-path cover identification problem are defined as follows:

- **NSPCI.** The node SPCI (NSPCI) is identifying $C_R(U)$ for a given $U \subseteq V$.
- **ESPCI.** The edge shortest-path cover identification (ESPCI) is determining whether $e \in C_R(U)$ or not for given $e \in E$ and $U \subseteq V$.

It is clearly that NSPCI can be solved through ESPCI.

We consider two different covering model R:

- Union $R=\cup$, $C_\cup(u, v)=\cup_{p \in SP(u, v)} ES(p)$.
- Intersection $R=\cap$, $C_\cap(u, v)=\cap_{p \in SP(u, v)} ES(p)$.

The two covering models above characterize the best and worst case of covered edges that can be learned from u, v [Boothe, Dvorák, Farley et al. (2007)]. It is clearly that $C_\cap(u, v) \subseteq C_\cup(u, v)$ and $C_\cap(U) \subseteq C_\cup(U)$ hold.

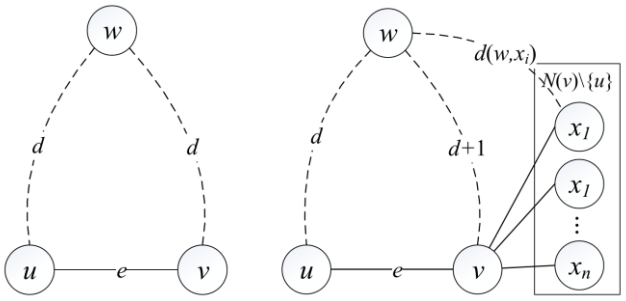
In this paper, we only consider simple networks, which are undirected and unweighted, but the proposed methods can be easily extended to directed and weighted networks.

Table 1: Frequently used notations

Notation	Description
$G(V, E)$	A graph with nodes V and edges E
$d_G(u, v)$	Shortest-path distance between u, v in G
$\widehat{d_G}(u, v)$	Approximate shortest-path distance between u, v in G
$N(v)$	Neighbors of node v
$SP(u, v)$	All the shortest-paths between u, v
$ES(p)$	Edge set constituting path p
R	Covering model
$C_R(U)$	Covered edge set from sources U to all other nodes
$r_R(U)$	Coverage rate of U, i.e., $\|C_R(U)\|/\|E\|$

2.2 Solving ESPCI by SPDQ

As is pointed out above, the time complexity of computing SPCI through the existing methods such as BFS is high, thus it is impractical for large-scale complex networks. Here we give the connection between ESPCI and SPDQ and show that ESPCI can be solved by SPDQ. SPDQ focuses on the distance between two nodes u, v in $G(V, E)$ rather than the actual shortest-paths. For example in Fig. 1, instead of concerning the specific shortest-paths between w and u, we only focus on the shortest-path distance $d_G(w, u)$.



(a) $d_G(w, u)=d_G(w, v)$ (b) $d_G(w, u)\neq d_G(w, v)$

Figure 1: Solving SPCI based on SPDQ

In the following, we come up with necessary and sufficient conditions for the two covering models.

Theorem 1. *For any edge* $e=(u, v)$, $e \notin C_{\cup}(w)$ *and* $e \notin C_{\cap}(w)$ *hold when* $d_G(w, u)=d_G(w, v)$*, otherwise* $e \in C_{\cup}(w)$ *holds*.

Proof. If $d_G(w, u)=d_G(w, v)$ holds, $\forall\, p \in SP(w, u)$, $e \notin ES(p)$, otherwise there is at least one shortest-path from w to u through v, leading to $d_G(w, v) < d$. In a similar way,

$\forall p' \in SP(w, u)$, $e \notin ES(p')$. Therefore, $e \notin C_{\cup}(w)$ and $e \notin C_{\cap}(w)$ hold by the definitions of the two routing models, as is shown in Fig. 1(a). If $d_G(w, u) \neq d_G(w, v)$ holds, we might suppose $d_G(w, u)=d+1$ and $d_G(w, v)=d$, then there is a shortest-path from w to u with e as the last edge, so $e \in C_{\cup}(w)$ holds.

Theorem 2. *Let $N(v)$ denotes the neighbors of v. If $d_G(w, u) = d$ and $d_G(w, v)=d+1$ are satisfied for $e=(u, v)$,then $e \in C_{\cap}(w)$ holds when $\forall x \in N(v)\backslash\{u\}$, $d_G(w, x)\neq d$, otherwise $e \notin C_{\cap}(w)$ holds.*

Proof. $d_G(w, u)=d$ and $d_G(w, v)=d+1$, and $\forall x \in N(v)\backslash\{u\}$, $d_G(w, u)\neq d$ leads to the fact that $\forall p \in SP(w, v)$, e is the last edge of p, therefore $e \in C_{\cap}(w)$. If $\exists x \in N(v)\backslash\{u\}$ satisfies $d_G(w, x)=d$, there is at least one path $p' \in SP(w, v)$ whose last edge is (x, v), so $e \notin C_{\cap}(w)$ holds, as is shown in Fig. 1(b).

Let $\Delta_w(u, v)$ denote the distance gap from w to u and v, i.e., $\Delta_w(u, v)=d_G(w, u)-d_G(w, v)$. Combining Theorem 1 and Theorem 2, we can conclude that both $C_{\cup}(w)$ and $C_{\cap}(w)$ can be determined by Δ_w. More precisely, as is shown in Fig. 2, we classify the neighbors of v into three sets by the distance gap from w. The three sets are $N_w^{+1}(v)$, $N_w^{0}(v)$ and $N_w^{-1}(v)$, e.g., $N_w^{+1}(v)$ denotes the subset of $N(v)$ with $\forall u \in N_w^{+1}(v)$, $d_G(w, u)=d_G(w, v)+1$ holding. Starting from the three neighbor sets, we give that:

- edges between v and $N_w^{+1}(v)$ or $N_w^{-1}(v)$ are in $C_{\cup}(w)$.
- edges between v and $N_w^{0}(v)$ are not in $C_{\cup}(w)$ or $C_{\cap}(w)$.
- edges between v and $N_w^{-1}(v)$ is in $C_{\cap}(w)$ iff $|N_w^{-1}(v)|=1$.

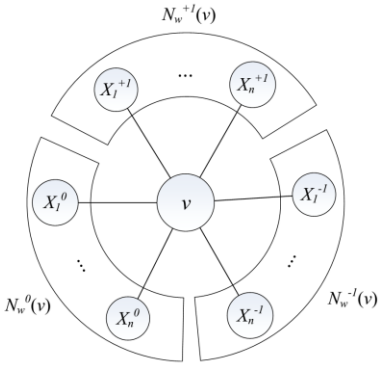


Figure 2: Three neighbor sets according to distances gap

We call a subgraph $G_T(V_T, E_T)$ as a detachable tree if it is a tree and it has at most one node b_T connect to the other subgraph G-G_T. According to Theorem 1 and Theorem 2, we can deduce:

Corollary 1. *Edges in a detachable tree of G can be covered by any nodes in V.*

Proof. Based on the definition, $\forall u_T \in V_T$ and $\forall w \in V$ there is only one shortest-path between w and u_T. So $\forall e_T=(u_T, v_T) \in E_T$ and $\forall x_T \in N(v_T)\backslash\{u_T\}$ where $\$v_T \neq b_T$, $d_G(w, u_T)\neq d_G(w, v_T)\neq d_G(w, x_T)$ holds. According to Theorem 1 and Theorem 2, $e \notin C_{\cup}(w)$ and $e \notin C_{\cap}(w)$ hold.

Corollary 1 will simplify SPCI by converting an original network G to a smaller scale one

G' by removing all the detachable trees from G. This can be done by recursively deleting the 1-degree node. The simplified results are shown in Section 4.1.

Combining Theorem 1, Theorem 2 and Corollary 1, we have shown a solution for ESPCI based on SPDQ.

SPDQ is a faster alternative to the single-source shortest-path and all-pairs shortest-path algorithms. This can be achieved by preprocessing the network and creating an auxiliary data structure to answer queries [Sommer (2014)]. Generally, a so-called preprocessing algorithm computes certain information for the next phase. After this preprocessing step, we can ask shortest-path distance, which should be answered as fast as possible. Many studies have focused on SPDQ, and they can be divided into two categories: exact methods [Akiba, Iwata and Yoshida (2013)] and approximate methods [Zhao, Sala, Wilson et al. (2010); Papadopoulos, Krioukov, Boguñá et al. (2010)]. A large portion of the exact methods are based on 2-hop cover [Abraham, Delling, Glodberg et al. (2012); Cheng and Yu (2009) ; Cohen, Halperin, Kaplan et al. (2003); Jin, Ruan, Xiang et al. (2012); Chen, Chen, Liu et al. (2018)] or tree decompositions [Akiba, Sommer and Kawarabayashi (2012)], and the main approach of the approximate methods is landmark-based [Chen, Sommer, Teng et al. (2013); Potamias, Bonchi, Castillo et al. (2009); Gubichev, Bedathur, Seufert et al. (2010); Qiao, Cheng, Chang et al. (2012); Tretyakov, Armas-Cervantes and García-Bañuelos et al. (2011)]. See [Sommer (2014); Madkour, Aref, Rehman et al. (2017)] for the review of recent progress on SPDQ.

All the SPDQ algorithms can be adopted for the SPCI problem. As we only need $d_G(u, v)$ or $\widehat{d_G}(u, v)$ according to Theorem 1, Theorem 2, therefore, both the exact methods and the approximate methods can be used to solve SPCI.

3 Algorithm

As a fundamental operation, SPCI should be computed fast and accurately, e.g., it is necessary to compute NSPCI from each node in V for the monitor deploying optimization problems [Han and Xu (2008); Boothe, Dvorák, Farley et al. (2007)]. In this section, we aim at fast SPCI. We present methods for solving SPCI based on SPDQ, and we propose the semi-approximate SPDQ algorithm as a trade-off between exact SPDQ and approximate SPDQ. Furthermore, we develop network-sampling strategies to estimate the coverage rate.

3.1 SPCI algorithms based on SPDQ

Using Theorem 1, Theorem 2 and Corollary 1, we can solve SPCI. The algorithms for ESPCI and NSPCI based on SPDQ are described as Algorithm 1, Algorithm 2 and Algorithm 3.

As mentioned above, all the SPDQ algorithms can be applied, but there are different time and space complexity between SPCI solutions by different SPDQ. Comparing all the SPDQ algorithms are out of the scope of this paper, here we focus on two representative SPDQ algorithms-PLL (Pruned Landmark Labeling) [Akiba, Iwata and Yoshida (2013)] for the exact method and LT (Landmarks and Triangulation) [Goldberg and Harrelson (2005)] for the approximate method.

Algorithm 1: ESPCI from w to $e=(u, v)$ when $R=\cup$

Input:	network G, node w, edge $e=(u, v)$.
Output:	whether $e \in C_{\cup}(w)$ or not.
1	**if** $SPDQ(w, u, G)=SPDQ(w, v, G)$ **then**
2	**return** $e \notin C_{\cup}(w)$
3	**else**
4	**return** $e \in C_{\cup}(w)$
5	**end if**

Algorithm 2: ESPCI from w to $e=(u, v)$ when $R=\cap$

Input:	network G, node w, edge $e=(u, v)$.
Output:	whether $e \in C_{\cap}(w)$ or not.
1	**if** $SPDQ(w, u, G)=SPDQ(w, v, G)$ **then**
2	**return** $e \notin C_{\cap}(w)$
3	**else**
4	**if** $SPDQ(w, u, G)+1=SPDQ(w, v, G)$ **then**
5	$u, v \leftarrow v, u$
6	**end if**
7	**if** $\forall x \in N(v)\backslash\{u\}$, $SPDQ(w, x, G)\neq SPDQ(w, v, G)$ **then**
8	**return** $e \in C_{\cup}(w)$
9	**else**
10	**return** $e \notin C_{\cup}(w)$
11	**end if**
12	**end if**

Algorithm 3: NSPCI from w

Input:	network G, node w.
Output:	$C_R(w)$.
1	$C_R(w) \leftarrow \emptyset$
2	compute $SSSPD(w)=\{SPDQ(w, u, G) \mid u \in V\}$
3	**for** $u \in V$ **do**
4	compute $N_w^{+1}(v)$, $N_w^{0}(v)$ and $N_w^{-1}(v)$ by $SSSPD(w)$
5	add the corresponding edges into $C_R(w)$ by Theorem 1 and 2
6	**end for**
7	**return** $C_R(w)$

PLL conducts a BFS from each node and adds the distance information to the labels of visited nodes. By BFS pruning and bit-parallel operations, it takes PLL microseconds to answer a distance query for networks with millions of nodes.

LT uses a set of landmarks $L \subseteq V$ and triangle inequality to estimate shortest-path distances. Each landmark $l \in L$ computes and stores distances to all the other nodes. For any distance query between w and v, the distance result is estimated as $SPDQ(w, v, G) = \min_{l \in L}\{d_G(w, l)+d_G(l, v)\}$. Furthermore, instead of minimizing over all landmarks (in time $O(|L|)$), in many schemes, nodes designate a nearest landmark for triangulation. Let $l_w(l_v)$ be a landmark that is the closest one to w (v), then the distance result is $SPDQ(w, v, G)=\min\{d_G(w, l_w)+d_G(l_w, v), d_G(w, l_v)+d_G(l_v, v)\}$. The query time complexity of nearest landmark scheme is $O(1)$. The quality of LT is highly depending on the landmark selection. Instead of random selection, several heuristics have been proposed to improve coverage [Sommer (2014)]. In this paper, we adopt nearest landmark scheme and max degree selection strategy, and then compute distance from landmarks to all nodes by PLL.

3.2 Semi-approximate SPDQ

ESPCI depends on accurately estimating shortest-path distances from w to $e=(u, v)$, as is shown in Fig. 1. PLL gives exact distances for both $w \rightarrow u$ and $w \rightarrow v$. On the contrary, LT gives approximate distances for both $w \rightarrow u$ and $w \rightarrow v$. Although LT is faster than PLL, it is inaccurate. The semi-approximate SPDQ gives the exact distance for one side, e.g., $w \rightarrow u$, and the approximate distance for the other side, e.g., $w \rightarrow v$. Therefore, the semi-approximate SPDQ is faster than PLL and more accurate than LT.

The basic starting point of the semi-approximate SPDQ is the shortest paths correlation between $w \rightarrow u$ and $w \rightarrow v$, that is, if l belongs to one of the shortest paths between w, u, it is highly probable that l also belongs to one of the shortest paths between w, v.

We improve the distance query progress from PLL to get the semi-approximate SPDQ algorithm. In PLL, for each node $u \in V$, there is a candidate node set $CA(u) \subseteq V$ and a label set $L(u)=\{(s, d_G(s, u)) \mid s \in CA(u)\}$. PLL answers the $d_G(w, u)$ as $\min\{ d_G(s, w)+d_G(s, u) \mid (s, d_G(s, u)) \in L(w), (s, d_G(s, u)) \in L(u)\}$. We first construct a cache node set $CN(w, u)=\{s \mid s \in CA(w) \cap CA(u)\}$ with at most K nodes as well as querying $d_G(w, u)$ by PLL, then we directly estimate distance between w, v as $\widehat{d_G}(w, v)=\min\{ d_G(s, w)+d_G(s, u) \mid s \in CN(w, u), (s,d_G(s, v)) \in L(v)\}$. The semi-approximate SPDQ algorithm is described as Algorithm 4.

Typically, K is much smaller than the average size of label sets. Therefore, the semi-approximate SPDQ algorithm is faster than PLL on ESPCI. For Algorithm 1, the query time complexity is $O(L(w)+L(u)+K+L(v))$ by the semi-approximate SPDQ against $O(2L(w)+L(u)+L(v))$ by PLL. In Algorithm 2, we can achieve more acceleration by estimating distances from v to $N(v)\backslash\{u\}$ when $d_G(w, v)>d_G(w, u)$ holds. See Section 4.3 for performance comparisons.

The semi-approximate SPDQ algorithm is not suitable for NSPCI. In NSPCI, we can precompute all the exact distances from the given node w, then check all edges by Algorithm 1, Algorithm 2. Therefore, it is unnecessary to estimate the distance as Algorithm 4.

Algorithm 4: Semi-Approximate SPDQ from w to e=(u, v)

Input:	network G, node w, edge e=(u, v), label sets $L(w)$, $L(u)$, K.
Output:	$d_G(w, u)$, $\widehat{d_G}(w, v)$.
1	$CN \leftarrow \emptyset$, $d_G(w, u) \leftarrow \infty$, $\widehat{d_G}(w, v) \leftarrow \infty$
2	**for** $(s, d_G(s, u)) \in L(w)$, $(s, d_G(s, u)) \in L(u)$ **do**
3	**if** $d_G(s, w)+d_G(s, u) < d_G(w, u)$ **then**
4	$d_G(w, u)=d_G(s, w)+d_G(s, u)$
5	$CN \leftarrow \emptyset$
6	**end if**
7	**if** $d_G(s, w)+d_G(s, u)=d_G(w, u)$ and $\|CN\| < K$ **then**
8	$CN \leftarrow CN \cup \{s\}$
9	**end if**
10	**end for**
11	**for** $s \in CN$, $(s, d_G(s, v)) \in L(v)$ **do**
12	**if** $d_G(s, w)+d_G(s, v) < \widehat{d_G}(w, v)$ **then**
13	$\widehat{d_G}(w, v)=d_G(s, w)+d_G(s, v)$
14	**end if**
15	**end for**
16	**return** $d_G(w, u)$, $\widehat{d_G}(w, v)$

Although the semi-approximate SPDQ algorithm is not suitable for NSPCI, it is compatible with edges sampling (see Section 3.4 for details).

It is worth mention that the semi-approximate SPDQ algorithm can be applied in all SPDQ algorithms based on 2-hop cover. All the 2-hop cover algorithms have a label set $L(u)$ for node u, and that is all the semi-approximate SPDQ algorithm needed, we just pick PLL as a typical example in this paper.

3.3 Time complexity analysis

PLL takes $O(W|E|\log|V|+W^2|V|\log^2|V|)$ time to preprocess with $O(W|V|\log|V|)$ space, and answers distance query in $O(W\log|V|)$ time [Akiba, Iwata and Yoshida (2013)], where W is the tree-width [Robertson and Seymour (1986)] of G.

The time complexity and space complexity of the semi-approximate SPDQ are the same as PLL. The difference between the semi-approximate SPDQ and PLL is that the former one has a relatively small constant factor for ESPCI.

LT takes $O(W|E|\log|V|+W^2|V|\log^2|V|+W|L||V|\log|V|)$ time to preprocess with $O(W|V|\log|V|+|L||V|)$ space, and answers distance query in $O(1)$ time.

To offset the pretreatment time, we usually require multiple queries for ESPCI, which is consistent with the actual situation.

The query time complexity of PLL and LT for NSPCI and ESPCI when $|U|=1$ are shown in Tab. 2. For ESPCI when $R=\cap$, we need to check all the neighbors of one end node from edge $e=(u, v)$ (see Algorithm 2) in the worst case, and it takes $O(MW\log|V|)$ for PLL and $O(M)$ for LT, where M is the max degree of G. In most of the real-world networks, W is small and only a fraction of nodes is degree closing to M, which makes ESPCI and NSPCI fast in practical.

Table 2: Query time complexity for ESPCI and NSPCI

		PLL algorithm	LT algorithm
ESPCI	$R=\cup$	$O(W\log\|V\|)$	$O(1)$
	$R=\cap$	$O(MW\log\|V\|)$	$O(M)$
NSPCI	$R=\cup$	$O(W\|V\|\log\|V\|+\|E\|)$	$O(\|E\|)$
	$R=\cap$	$O(W\|V\|\log\|V\|+M\|E\|)$	$O(M\|E\|)$

3.4 Network sampling for NSPCI

In most scenarios, only the coverage rate of NSPCI is concerned instead of the actual covered edges. We denote $r_R(U)$ as the coverage rate of U, that is $r_R(U)=|C_R(U)|/|E|$. It costs $O(W|U||V|\log|V|+|U||E|)$ time to compute NSPCI for each node in U, which is impractical for large networks with millions of nodes. Here we estimate $r_R(U)$ by network sampling.

Formally, the network sampling for NSPCI is to sample only a small portion of edges from E to unbiased estimate the coverage rate. For estimating $r_R(U)$, we first sample a edges list E_s, then we solve the ESPCI for each node $w \in U$ by Algorithm 1 or Algorithm 2 against E_s, and then, we merge the results to get $r_R(U)$. The algorithm for one-node NSPCI $r_R(U)$ estimation is described as Algorithm 5.

Algorithm 5: NSPCI coverage rate estimation from w

Input:	network G, node w.
Output:	estimated value of $r_R(w)$.
1	$E_s \leftarrow$ sampled edges list from E
2	$n \leftarrow 0$
3	**for** $e \in E_s$ **do**
4	**if** $e \in C_R(w)$ **then**
5	n++
6	**end if**
7	**end for**
8	**return** $n/\|E_s\|$

Two sampling strategies can be applied to unbiased estimate the coverage rate, i.e., the *N-sampling* and the *B-sampling*. The *N-sampling* uniformly samples N edges from E without replacement, and the *B-sampling* uniformly samples B edges from E with replacement.

It can be proved that Algorithm 5 using any of the two sampling strategies is an unbiased estimation [Frieze and Karoński (2015)]. The standard deviation is $\sqrt{r_R(1\text{-}r_R)(|E|\text{-}N)/N(|E|\text{-}1)}$ for *N-sampling* and $\sqrt{r_R(1\text{-}r_R)/B}$ for *B-sampling*.

Furthermore, the *N-sampling* can be approximated by *Bernoulli sampling*, that is, sampling each edge with probability $p=N/|E|$ [Kolaczyk (2009)].

3.5 Extensions

Weighted networks: To treat weighted graphs, the only necessary change is to perform pruned Dijkstra's algorithm or Bellman-Ford algorithm instead of pruned BFSs in PLL [Akiba, Iwata and Yoshida (2013)]. Dong et al. [Dong, Lakhotia, Zeng et al. (2018)] and Qiu et al. [Qiu, Zhu, Yuan et al. (2018)] have already extended PLL for large-scale weighted graphs.

Directed networks: To treat directed graphs, we first redefine $d_G(u, v)$ as the distance from u to v. Then, we store two labels L_{out} and L_{in} for each vertex for out- and in- direction. We can answer the distance from u to v by $L_{out}(u)$ and $L_{in}(v)$. To compute these labels, from each vertex, we conduct pruned BFSs twice: once in the forward direction and once in the reverse direction [Akiba, Iwata and Yoshida (2013)].

Because the semi-approximate SPDQ and LT are both depend on PLL, they can be easily extended to weighted and directed networks as well.

4 Experimental evaluation

Experimental evaluation is presented on both synthetic and real-world networks. In our research, we generate and operate networks by NetworkX [Hagberg, Swart and Schult (2008)], and we implement PLL, LT and the semi-approximate SPDQ by C++ based on [Akiba, Iwata and Yoshida (2013)]. Our experiments are executed on a 64-bit windows server with Intel Xeon E5-2667 v4 CPU, 256GB memory and 25MB cache.

4.1 Datasets

The datasets we used are listed in Tab. 3. We conduct experiments on two random networks ER [Erdös and Rényi (1960)], BA [Barabási and Albert (1999)] and three real-world networks [Leskovec and Krevl (2015)].

ER. In ER model $G(|V|, p)$, a network is constructed by connecting nodes randomly from an initial empty network of $|V|$ nodes. Each edge is included in the network with probability p independent from every other edge.

BA. In BA model $G(|V|, m)$, a network begins with an initial empty network of m nodes. New nodes are added to the network one at a time. Each new node is connected to m existing nodes with a probability that is proportional to the number of links that the existing nodes already have.

Enron. The Federal Energy Regulatory Commission originally makes Enron email network

public during its investigation. This data covers all the email communication within a dataset of around half a million emails. Nodes of the network are email addresses, and if there were at least one email between two nodes u, v, edge (u, v) is added [Klimt and Yang (2004)].

BerkStan. The web network is collected in 2002. Nodes represent pages from berkely.edu and stanford.edu domains and edges represent hyperlinks between them [Leskovec, Lang, Dasgupta et al. (2009)].

Skitter-AS. It is an internet router network from Skitter project running daily in 2005. The data is collected by traceroutes from several scattered sources to million destinations [McRobb, Claffy and Monk (1999)].

Table 3: Experimental datasets

Dataset	Network	$\|V\|$	$\|E\|$	M	W
ER	Random	1K	-	-	-
BA	Random	1K	-	-	-
Enron	Social	37K	184K	1.4K	<2.0K
BerkStan	Web	685K	6.6M	84K	<2.1K
Skitter-AS	Computer	1.7M	11M	35K	<22K

Without specification, we set $|V|$ =1000, p =0.005 and m =5 for ER and BA, and we independently generate 500 networks under the given parameters for averaging in this section.

We only consider the largest connected component $G_C(V_C, E_C)$ of networks. The reduction from $G_C(V_C, E_C)$ to $G_{CT}(V_{CT}, E_{CT})$ (by removing all the detachable trees by Corollary 1) are shown in Fig. 3. The reduce rate is defined as $|V_{CT}|/|V_C|$.

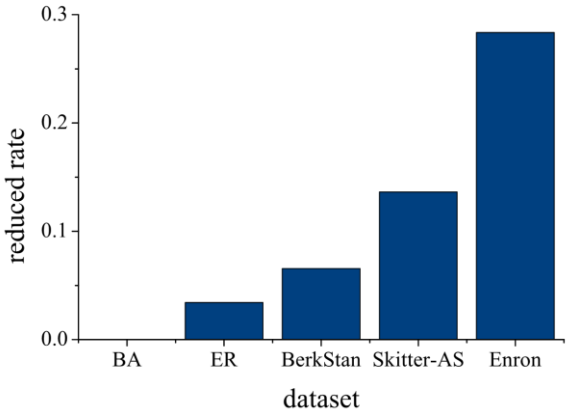


Figure 3: Reduced rate of the five networks by Corollary 1

We can see from Fig. 3 that the network simplification by Corollary 1 can reduce ~30% nodes and edges at most from real-world networks. The simplification does not always

work, e.g., for BA networks. m=5 indicates that each node in BA is 5-degree at least, so Corollary 1 can remove none of them.

4.2 Performance

The performance of PLL, LT and the semi-approximate SPDQ on the 3 real-world datasets for $|U|$=1 is shown in Tabs. 4-6, where PT denotes preprocessing time, and QT average query time, and DNF means it does not finish in one day. We evaluate the results by 10^3 random queries for NSPCI, 10^6 random queries for ESPCI, and setting K=8, $|L|$=500. We compare the three algorithms with the baseline method implemented in C/igraph library [Csardi and Nepusz (2006)], which constructs all the shortest paths between two nodes based on BFS.

All the three algorithms based on SPDQ are significant faster BFS, e.g., PLL accelerates BFS more than 10^5 times.

Tab. 4 shows the preprocessing time for the three real-world datasets. We can conclude that the preprocessing time of all the three algorithms are acceptable for large networks. The preprocessing time of the semi-approximate algorithm is the same as PLL, but the preprocessing time of LT is more than PLL, as shown in Tab. 4. The reason of the preprocessing time overhead is that LT uses PLL as a precondition for distance queries from landmarks to all the other nodes. The distance query time of PLL is generally microseconds, and the preprocessing time overhead of LT limited in hours for networks with millions of nodes. Although BFS has no preprocessing stage, it has long querying time.

Table 4: Preprocessing time for the 3 real-world datasets

Dataset	PLL/Semi-Approx.	LT	BFS
Enron	0.2 s	2.7 s	0 s
BerkStan	17 s	73 s	0 s
Skitter-AS	379 s	891 s	0 s

Table 5: ESPCI Performance for the 3 real-world datasets

Dataset	PLL		Semi-Approx.		LT		BFS
	R=∪	R=∩	R=∪	R=∩	R=∪	R=∩	
Enron	0.6 us	1.2 us	0.4 us	0.8 us	0.01 us	0.09 us	122 ms
BerkStan	0.7 us	1.4 us	0.5 us	1.0 us	0.03 us	0.08 us	204 s
Skitter-AS	1.8 us	3.7 us	1.1 us	2.2 us	0.05 us	0.12 us	92 s

Table 6: NSPCI Performance for the 3 real-world datasets

Dataset	PLL		LT		BFS
	R=∪	R=∩	R=∪	R=∩	
Enron	7.7 ms	10.3 ms	2.3 ms	6.1 ms	2939 s
BerkStan	376.4 ms	460.7 ms	85.5 ms	224.4 ms	DNF
Skitter-AS	2.0 s	2.3 s	0.38 s	0.94 s	DNF

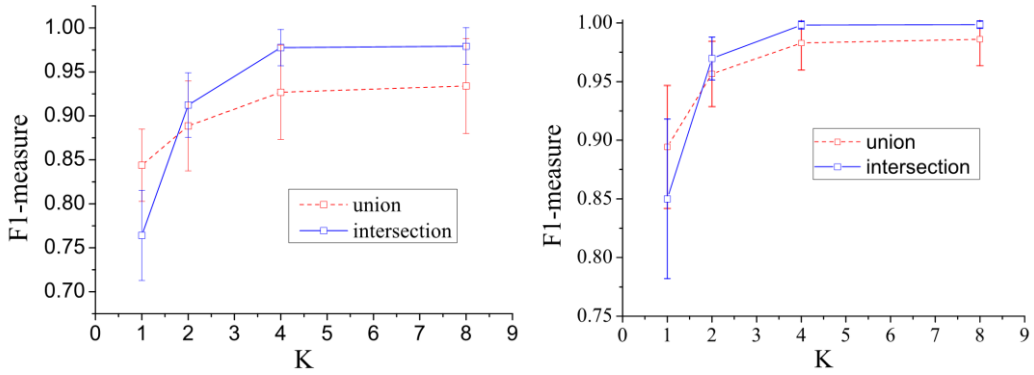
Comparing with PLL, the semi-approximate SPDQ improves the ESPCI speed more than 30% with only a small loss of accuracy (see Section 4.3). LT is much faster than PLL in query time, which gets 1.7x-38x and 13x-50x speed up respectively in NSPCI and ESPCI. As the semi-approximate SPDQ is not suitable for NSPCI (see Section 3.2), we only compare PLL and LT with BFS in Tab. 6.

4.3 Accuracy of semi-approximate and approximate SPDQ

In this section, we show the accuracy of the semi-approximate algorithm and LT under difference parameters, then we summary the application scenarios of the three SPDQ algorithms.

To quantify the accuracy, we consider $C_R(U)$ as a binary classification problem, that is, if $e \in C_R(U)$ is satisfied we treat e as a positive sample, otherwise a negative sample. The ground truth is retrieved by PLL. We use F1-measure to measure the semi-approximate SPDQ and LT performance for edge covered from a given node set, where the F1-measure is evaluated by 1000 random nodes.

For the semi-approximate SPDQ, the F1-measure over different size of the cache node set (K) on random networks is shown in Fig. 4. The F1-measure increases with K as the estimation of the shortest-path distance is more likely to be correct when K is large. The F1-measure is larger than 90% when K=8 for both ER and BA networks, so we choose K=8 for the real-world datasets. The F1-measure is beyond 99.4%, 99.9% and 92.3% on Enron, BerkStan and Skitter-AS respectively. Thus, the semi-approximate algorithm can speed up PLL more than 30% with the accuracy loss less than 10%.

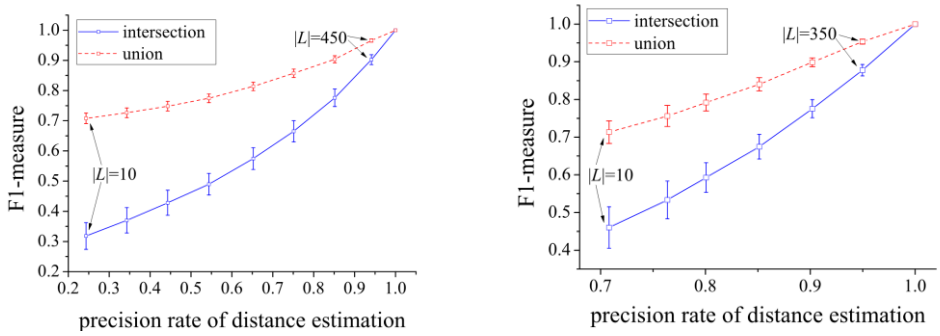


(a) ER(1000, 0.005) (b) BA(1000, 5)

Figure 4: F1-measure of the semi-approximate algorithm on random networks

For LT, the F1-measure over different landmark number $|L|$ on random networks is shown in Fig. 5. The x-axis in Fig. 5 is the accuracy of distance estimation controlled by $|L|$, and it is calculated as the percentage of correct shortest-path distance in 10^4 random pairs of nodes.

From Fig. 5, we can see that the mean of F1-measure increases while the standard deviation (error bar in Fig. 5) of F1-measure decreases as the accuracy of distance estimation. Thus, a properly $|L|$ should be chosen for estimating $C_R(w)$ with high quality.



(a) ER(1000, 0.005) (b) BA(1000, 5)

Figure 5: F1-measure of LT for random networks in different |*L*|

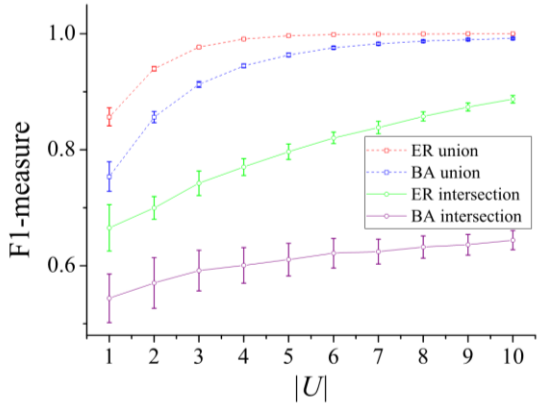
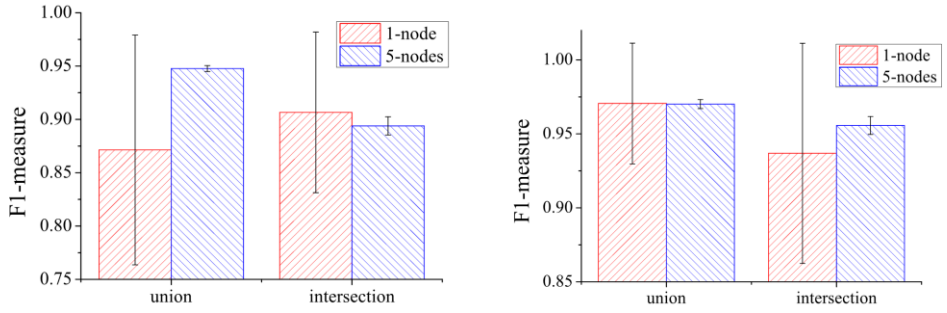
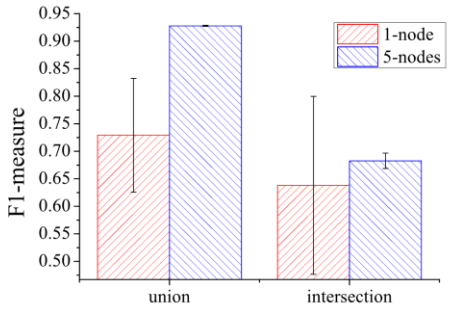


Figure 6: F1-measure of LT for random networks in different |*U*|



(a) Enron (b) BerkStan



(c) Skitter-AS

Figure 7: F1-measure of LT for real-world networks in different $|U|$

We also evaluate the F1-measure under different $|U|$, and the results are shown in Figs. 6 and 7. We fix $|L|$=140 for ER and $|L|$=20 for BA which achieves ~75% accuracy of distance estimation. For the 3 real-world networks, we fix $|L|$=200. $R=\cup$ obtains larger F1-measure margin than $R=\cap$, especially when $|L|$ is small. We get a higher F1-measure mean and a lower F1-measure standard deviation from larger $|U|$ for all networks. The F1-measure is decided by every node in U that e belongs to $C_R(U)$ or not, if there exists at least one node w in U satisfying $e \in C_R(w)$, then $e \in C_R(U)$. This fact leads to higher accuracy and recall with a higher F1-measure.

From Figs. 6 and 7, we get a better F1-measure by increasing $|U|$ when $|L|$ is fixed for a given network. On the other hand, we can approximately estimate $C_R(U)$ with high quality by a relatively small $|L|$ if $|U|$ is large.

Therefore, LT is valid for SPCI. The quality of estimating $C_R(U)$ by LT is related to $|L|$ and $|U|$. For unbiased estimation, a high accuracy of distance query is required for the small $|U|$, but this restrictive condition can be relaxed when $|U|$ is large.

In summary, the application scenarios of the three SPDQ algorithms, i.e., PLL, the semi-approximate SPDQ and LT, are different. When the accuracy is crucial, PLL should be applied, and when the speed of identification is required, LT should be picked. However, the semi-approximate SPDQ can be used as an alternative to improve the speed of PLL with a small accuracy loss.

4.4 Network sampling

We compare the two sampling strategies in this section. We compute $r_R(w)$ by PLL on all the datasets, and the mean coverage rate is 20%-66% for $R=\cup$ and 5%-22% for $R=\cap$. Here we choose BerkStan with $r_\cup(w)$=50% and Skitter-AS with $r_\cap(w)$=20% for evaluation. The results are shown in Tab. 6. The Mean and standard deviation are evaluated by 10^3 times.

If the sample size is to be determined such that the standard deviation of estimated $r_R(w)$ should not exceed a given value, e.g., 1%, we can deduct the expected sample size. For example, $\sqrt{r_R(1\text{-}r_R)/B}$<1% leads to B>2500 for *B-sampling*. As is shown in Tab. 6, we can estimate $r_R(w)$ with less than ten thousand edges, which means we can handle NSPCI

for networks with hundreds of millions of nodes and edges within a few seconds.

Table 7: Performance of two network-sampling strategies

Strategy	Parameter	BerkStan		Skitter-AS	
		Mean (%)	Std. (%)	Mean (%)	Std. (%)
B-sampling	25	51.08	10.29	20.36	7.09
	2500	49.97	0.90	19.96	0.82
	250000	49.97	0.093	20.00	0.073
N-sampling	25	49.52	10.10	19.48	8.38
	2500	49.93	0.89	20.00	0.79
	250000	50.00	0.098	20.00	0.084

5 Conclusion

In this paper, we present a method for SPCI in large complex networks. By converting NSPCI, ESPCI into SPDQ, we solve them by three algorithms, i.e., PLL, the semi-approximate SPDQ and LT. The semi-approximate SPDQ is proposed as a faster alternative of PLL with a small accuracy loss. Moreover, we develop two sampling strategies for further acceleration, and we show the effectiveness of the proposed method through synthetic and real-world networks. Our method can handle large complex networks with hundreds of millions of nodes, which can be used as a fundamental module for related problems.

Acknowledgement: This work was supported in part by the National Natural Science Foundation of China (61471101) and the National Natural Science Foundation of China (U1736205).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

Abraham, I.; Delling, D.; Glodberg, A. V.; Werneck, R. F. (2012): Hierarchical hub labelings for shortest paths. *ESA'12 Proceedings of the 20th Annual European Conference on Algorithms*, pp. 24-35.

Akiba, T.; Iwata, Y.; Yoshida, Y. (2013): Fast exact shortest-path distance queries on large networks by pruned landmark labeling. *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pp. 349-360.

Akiba, T.; Sommer, C.; Kawarabayashi, K. I. (2012): Shortest-path queries for complex networks: exploiting low tree-width outside the core. *Proceedings of the 15th International Conference on Extending Database Technology*, pp. 144-155.

Barabási, A. L.; Albert, R. (1999): Emergence of scaling in random networks. *Science*, vol. 286, no. 5439, pp. 509-512.

Barford, P.; Bestavros, A.; Byers, J.; Crovella, M. (2001): On the marginal utility of network topology measurements. *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, pp. 5-17.

Blondel, V. D.; Guillaume, J. L.; Hendrickx, J. M.; Jungers, R. M. (2007): Distance distribution in random graphs and application to network exploration. *Physical Review E*, vol. 76, no. 6, pp. 66-101.

Boothe, P.; Dvorák, Z.; Farley, A. M.; Proskurowski, A. (2007): Graph covering via shortest paths. *Congressus Numerantium*, vol. 187, no. 1, pp. 145-155.

Chen, W.; Chen, Z. Y.; Liu, J.; Yang, Q. Z. (2018): A novel shortest path query algorithm. *Cluster Computing*, vol. 21, no. 1, pp. 1-12.

Chen, W.; Sommer, C.; Teng, S. H.; Wang, Y. (2012): A compact routing scheme and approximate distance oracle for power-law graphs. *ACM Transactions on Algorithms*, vol. 9, no. 1, pp. 1-26.

Cheng, J.; Yu, J. X. (2009): On-line exact shortest distance query processing. *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, pp. 481-492.

Cohen, E.; Halperin, E.; Kaplan, H.; Zwick, U. (2003): Reachability and distance queries via 2-hop labels. *SIAM Journal on Computing*, vol. 32, no. 5, pp. 1338-1355.

Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; Stein, C. (2001): Introduction to algorithms 2nd edition. *Knuth-Morris-Pratt Algorithm*, pp. 491-527.

Csardi, G.; Nepusz, T. (2006): The igraph software package for complex network research. https://igraph.org/.

Dall'Asta, L.; Alvarez-Hamelin, I.; Barrat, A.; Vázquez, A.; Vespignani, A. (2006): Exploring networks with traceroute-like probes: theory and simulations. *Theoretical Computer Science*, vol. 355, no. 1, pp. 6-24.

Dong, Q.; Lakhotia, K.; Zeng, H.; Karman, R.; Prasanna, V. et al. (2018): A fast and efficient parallel algorithm for pruned landmark labeling. *IEEE High Performance Extreme Computing Conference*, pp. 1-7.

Erdös, P.; Rényi, A. (1960): On the evolution of random graphs. *Publication of the Mathematical Institute of the Hungarian Academy of Sciences*, vol. 5, no. 1, pp. 17-61.

Frieze, A.; Karoński, M. (2015): *Introduction to Random Graphs*, pp. 421-444. Cambridge University Press.

Goldberg, A. V.; Harrelson, C. (2005): Computing the shortest path: a search meets graph theory. *Proceedings of the Sixteenth Annual ACM-SIAM symposium on Discrete Algorithms*, pp. 156-165.

Gubichev, A.; Bedathur, S.; Seufert, S.; Weikum, G. (2010): Fast and accurate estimation of shortest paths in large graphs. *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pp. 499-508.

Guillaume, J. L.; Latapy, M. (2005): Complex network metrology. *Complex Systems*, vol. 16, no. 1, pp. 83-94.

Hagberg, A.; Swart, P.; Schult, D. (2008): Exploring network structure, dynamics, and function using NetworkX. *Proceedings of the 7th Python in Science Conference*, pp. 11-16.

Han, W.; Xu, K. (2008): A method for placing traceroute-like topology discovery instrumentation. *11th IEEE Singapore International Conference on Communication Systems*, pp. 1160-1164.

Jin, R.; Ruan, N.; Xiang, Y.; Lee, V. (2012): A highway-centric labeling approach for answering distance queries on large sparse graphs. *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pp. 445-456.

Klimt, B.; Yang, Y. (2004): The enron corpus: a new dataset for email classification research. *European Conference on Machine Learning*, pp. 217-226.

Kolaczyk, E. D. (2009): *Statistical Analysis of Network Data*, pp. 123-153. Springer Press.

Leskovec, J.; Krevl, A. (2015): SNAP datasets: stanford large network dataset collection. http://snap.stanford.edu/data

Leskovec, J.; Lang, K. J.; Dasgupta, A.; Mahoney, M. W. (2009): Community structure in large networks: natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, vol. 6, no. 1, pp. 29-123.

Madkour, A.; Aref, W. G.; Rehman, F. U.; Rahman, M. A.; Basalamah, S. (2017): A survey of shortest-path algorithms. arXiv:1705.02044.

McRobb, D.; Claffy, K.; Monk, T. (1999): Skitter: CAIDA's macroscopic internet topology discovery and tracking tool. http://www.caida.org/tools/measurement/skitter/

Ouédraogo, F.; Magnien, C. (2011): Impact of sources and destinations on the observed properties of the internet topology. *Computer Communications*, vol. 34, no. 5, pp. 670-679.

Papadopoulos, F.; Krioukov, D.; Boguñá, M.; Vahdat, A. (2010): Greedy forwarding in dynamic scale-free networks embedded in hyperbolic metric spaces. *Proceedings IEEE INFOCOM*, pp. 1-9.

Pignolet, Y. A.; Schmid, S.; Tredan, G. (2017): Tomographic node placement strategies and the impact of the routing model. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 1, no. 2, pp. 1-23.

Potamias, M.; Bonchi, F.; Castillo, C.; Gionis, A. (2009): Fast shortest path distance estimation in large networks. *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, pp. 867-876.

Qiao, M.; Cheng, H.; Chang, L.; Yu, J. X. (2012): Approximate shortest distance computing: a query-dependent local landmark scheme. *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 1, pp. 55-68.

Qiu, K.; Zhu, Y.; Yuan, J.; Zhao, J.; Wang, X. et al. (2018): ParaPLL: fast parallel shortest-path distance query on large-scale weighted graphs. *Proceedings of the 47th International Conference on Parallel Processing*, pp. 2-12.

Robertson, N.; Seymour, P. D. (1986): Graph minors. II. algorithmic aspects of tree-width. *Journal of Algorithms*, vol. 7, no. 3, pp. 309-322.

Sommer, C. (2014): Shortest-path queries in static networks. *ACM Computing Surveys*, vol. 46, no. 4, pp. 1-31.

Spring, N.; Mahajan, R.; Wetherall, D. (2002): Measuring ISP topologies with Rocketfuel. *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 4, pp. 133-145.

Tretyakov, K.; Armas-Cervantes, A.; García-Bañuelos, L.; Vilo, J.; Dumas, M. (2011): Fast fully dynamic landmark-based estimation of shortest path distances in very large graphs. *Proceedings of the 20th ACM International Conference on Information and knowledge Management*, pp. 1785-1794.

Wei, F. (2010): TEDI: efficient shortest path query answering on graphs. *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, pp. 214-238.

Zhao, X.; Sala, A.; Wilson, C.; Zheng, H.; Zhao, B. Y. (2010): Orion: shortest path estimation for large social graphs. *Networks*, vol. 1, pp. 5-14.

Zou, X.; Qiao, Z.; Zhou, G.; Xu, K. (2009): A logic distance-based method for deploying probing sources in the topology discovery. *IEEE Global Telecommunications Conference*, pp. 1-6.