Querying Shortest Path Distance with Bounded Errors in Large Graphs

Miao Qiao, Hong Cheng, and Jeffrey Xu Yu

Department of Systems Engineering and Engineering Management
The Chinese University of Hong Kong
Hong Kong, China
{mqiao, hcheng, yu}@se.cuhk.edu.hk

Abstract. Shortest paths and shortest path distances are important primary queries for users to query in a large graph. In this paper, we propose a new approach to answer shortest path and shortest path distance queries efficiently with an error bound. The error bound is controlled by a user-specified parameter, and the online query efficiency is achieved with prepossessing offline. In the offline preprocessing, we take a reference node embedding approach which computes the single-source shortest paths from each reference node to all the other nodes. To guarantee the user-specified error bound, we design a novel coverage-based reference node selection strategy, and show that selecting the optimal set of reference nodes is NP-hard. We propose a greedy selection algorithm which exploits the submodular property of the formulated objective function, and use a graph partitioning-based heuristic to further reduce the offline computational complexity of reference node embedding.

In the online query answering, we use the precomputed distances to provide a lower bound and an upper bound of the true shortest path distance based on the triangle inequality. In addition, we propose a linear algorithm which computes the approximate shortest path between two nodes within the error bound. We perform extensive experimental evaluation on a large-scale road network and a social network and demonstrate the effectiveness and efficiency of our proposed methods.

1 Introduction

Querying shortest paths or shortest path distances between vertices in a large graph has important applications in many domains including road networks, social networks, biological networks, the Internet, and so on. For example, in road networks, the goal is to find shortest routes between locations or find nearest objects such as restaurants or hospitals; in social networks, the goal is to find the closest social relationships such as common interests, collaborations, citations, *etc.*, between users; while in the Internet, the goal is to find the nearest server in order to reduce access latency for clients. Although classical algorithms like breadth-first search (BFS), Dijkstra's algorithm [1], and A^* search algorithm [2] can compute the exact shortest paths in a network, the massive size of the modern information networks and the online nature of such queries make it infeasible to apply the classical algorithms online. On the other hand, it is space inefficient to precompute the shortest paths between all pairs of vertices and store them on disk, as it requires $O(n^3)$ space to store the shortest paths and $O(n^2)$ space to store the distances for a graph with n vertices.

Recently, there have been many different methods [3,4,5,6,7,8,9,10,11,12] for estimating the shortest path distance between two vertices in a graph based on graph embedding techniques. A commonly used embedding technique is reference node embedding, where a set of graph vertices is selected as reference nodes (also called landmarks) and the shortest path distances from a reference node to all the other nodes in a graph are precomputed. Such precomputed distances can be used online to provide an estimated distance between two graph vertices. Although most of the above mentioned methods follow the same general framework of reference node embedding, they differ in the algorithmic details in the following aspects: (1) reference node selection – some (*e.g.*, [6,7,10,11,12]) select reference nodes randomly, while others (*e.g.*, [3,4,8,9]) propose heuristics to select reference nodes; (2) reference node organization – [8,10,11,12] proposed a hierarchical embedding where reference nodes are organized in multiple levels, while most of the other methods use a flat reference node embedding; and (3) an error bound on the estimated shortest path distances – [6,10] analyzed the error bound of the estimated distances with random reference node selection, while most of the other methods have no error bounds or guarantees of the estimated distances.

A theoretical error bound can guarantee the precision of the estimated distance, but the derivation of an error bound is closely related to the reference node selection strategy. Random selection [6,7] or heuristic selection strategies (*e.g.*, based on degree or centrality) [9] cannot derive an error bound to control the precision of the estimated distance. In this paper, we propose a reference node embedding method which provides a distance estimation within a user-specified error bound ϵ. Specifically, we formulate a coverage-based reference node selection strategy, *i.e.*, every node in a graph should be "covered" by some reference node within a radius $c = \epsilon/2$. The coverage property will lead to a theoretical error bound of ϵ. Importantly, allowing a user-specified error bound increases the flexibility of our method in processing queries at different error tolerance levels – when a user specifies an error bound ϵ he can tolerate, we can compute the corresponding radius c for coverage and then the number of reference nodes that are necessary to ensure the error bound. On the other hand, if a user specifies the number of reference nodes he selects, we can find the corresponding value of c and the error bound. We will also show through experimental study that by adjusting the radius c, we can achieve a tradeoff between the theoretical error bound and the offline computational time of the reference node embedding process.

Our main contributions are summarized as follows.

- We take the reference node embedding approach and formulate the optimal reference node set selection problem in a coverage-based scheme. The coverage-based strategy leads to a theoretical error bound of the estimated distance. We show that selecting the minimum set of reference nodes is an NP-hard problem and then propose a greedy solution based on the submodular property of the proposed objective function.
- The reference node embedding can be used to compute an upper bound and a lower bound of the true shortest path distance between any two vertices based on the

triangle inequality. We show that the estimated distance is within a user-specified error bound of the true distance. To further reduce the offline computational complexity of the embedding approach, we propose a graph partitioning-based heuristic for reference node embedding with a relaxed error bound.

- Based on the estimated distances, we propose a linear algorithm to compute the approximate shortest path between two vertices. This algorithm improves the shortest path query efficiency of A^* search by three to five orders of magnitude, while the distance of the approximate shortest path is very close to the exact shortest distance.
- We performed extensive experiments on two different types of networks including a road network and a social network. Although these two types of networks exhibit quite different properties on vertex degree distribution, network diameter, *etc.*, our methods can achieve high accuracy and efficiency on both types of networks.

The rest of the paper is organized as follows. Section 2 introduces preliminary concepts and formulates the distance estimation problem. Section 3 presents our proposed algorithms for reference node selection, shortest path distance estimation and approximate shortest path search. A graph partitioning-based heuristic technique for reference node embedding with a lower offline complexity is proposed in Section 4. Section 5 presents extensive experimental results. We survey related work in Section 6 and conclude in Section 7.

2 Preliminaries and Problem Statement

The input is an edge weighted graph $G = (V, E, w)$, where V is a set of vertices, E is a set of edges, and $w : E \to R^+$ is a weighting function mapping an edge $(u, v) \in E$ to a positive real number $w(u, v) > 0$, which measures the length of (u, v). We denote $n = |V|$ and $m = |E|$. For a pair of vertices $s, t \in V$, we use $D(s, t)$ to denote the true shortest path distance between s and t. If $(s, t) \in E$, $D(s, t) = w(s, t)$. In this work, we focus on undirected graphs. Our problem can be formulated as follows.

Problem 1 (Distance Estimation with a Bounded Error). Given a graph G and a user-specified error bound ϵ as input, for any pair of query vertices (s, t), we study how to efficiently provide an accurate estimation of the shortest path distance $\widehat{D}(s, t)$, so that the estimation error $|\widehat{D}(s, t) - D(s, t)| \leq \epsilon$.

To efficiently provide a distance estimation, the basic idea is to use a *reference node embedding* approach. Consider a set of vertices $\mathcal{R} = \{r_1, \ldots, r_l\}$ ($\mathcal{R} \subseteq V$), which are called *reference nodes* (also called *landmarks*). For each $r_i \in \mathcal{R}$, we compute the single-source shortest paths to all vertices in V. Then for every node $v \in V$, we can use a l-dimensional vector representation as

$$\overrightarrow{D}(v) = \langle D(r_1, v), D(r_2, v), \ldots, D(r_l, v) \rangle$$

This approach is called reference node embedding. This embedding can be used to compute an upper bound and a lower bound of the true shortest path distance between two vertices (s, t). In the rest of the paper, we will discuss the following questions:

1. Given a graph G and an error bound ϵ, how to select the minimum number of reference nodes to ensure the error bound ϵ in the distance estimation?
2. How to estimate the shortest distance with an error bound given a query (s, t)?
3. How to efficiently compute an approximate shortest path P given a query (s, t)?

3 Proposed Algorithm

The quality of the estimated shortest path distance is closely related to the reference node selection strategy. Given a graph G and an error bound ϵ, we will first formulate a coverage-based reference node selection approach to satisfy the error bound constraint. We will then define an objective function over a set of reference nodes and discuss how to select the minimum set of reference nodes according to the objective function.

3.1 Reference Node Selection

Definition 1 (Coverage). *Given a graph $G = (V, E, w)$ and a radius c, a vertex $v \in V$ is covered by a reference node r if $D(r, v) \leq c$.*

The set of vertices covered by a reference node r is denoted as C_r, *i.e.*, $C_r = \{v | v \in V, D(r, v) \leq c\}$. In particular, we consider a reference node r is covered by itself, *i.e.*, $r \in C_r$, since $D(r, r) = 0 \leq c$. Here we formulate the problem of optimal reference node selection.

Problem 2 (Coverage-based Reference Node Selection). Given a graph $G = (V, E, w)$ and a radius c, our goal is to select a minimum set of reference nodes $\mathcal{R}^* \subseteq V$, *i.e.*, $\mathcal{R}^* = \arg\min_{\mathcal{R} \subseteq V} |\mathcal{R}|$, so that $\forall v \in V - \mathcal{R}^*$, v is covered by at least one reference node from $\mathcal{R}^*$.

Given a user-specified error bound ϵ, we will show in Section 3.3, when we set $c = \epsilon/2$, the coverage-based reference nodes selection method can guarantee that the error of the estimated shortest path distance is bounded by ϵ.

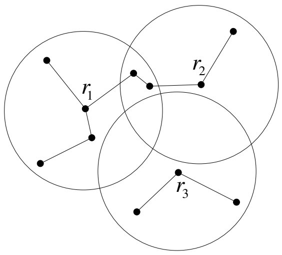

Fig. 1. Coverage-based Reference Node Selection

Example 1. Figure 1 shows a graph with three reference nodes r_1, r_2 and r_3. The three circles represent the area covered by the three reference nodes with a radius c. If a vertex lies within a circle, it means the shortest path distance between the vertex and the corresponding reference node is bounded by c. As shown in the figure, all vertices can be covered by selecting the three reference nodes.

Besides the coverage requirement, a reference node set should be as compact as possible. To evaluate the quality of a set of reference nodes $\mathcal{R}$, we define a gain function over $\mathcal{R}$.

Definition 2 (Gain Function). *The gain function over a set of reference nodes $\mathcal{R}$ is defined as*

$$g(\mathcal{R}) = |\bigcup_{r \in \mathcal{R}} C_r| - |\mathcal{R}| \tag{1}$$

In Figure 1, $g(\{r_1\}) = 5$, $g(\{r_2\}) = 3$, $g(\{r_3\}) = 2$ and $g(\{r_1, r_2, r_3\}) = 8$.
The gain function g is a submodular function, as stated in Theorem 1.

Definition 3 (Submodular Function). *Given a finite set N, a set function $f : 2^N \rightarrow R$ is submodular if and only if for all sets $A \subseteq B \subseteq N$, and $d \in N \setminus B$, we have $f(A \cup \{d\}) - f(A) \geq f(B \cup \{d\}) - f(B)$.*

Theorem 1. *For two reference node sets $A \subseteq B \subseteq V$ and $r \in V \setminus B$, the gain function g satisfies the submodular property:*

$$g(A \cup \{r\}) - g(A) \geq g(B \cup \{r\}) - g(B)$$

Proof. According to Definition 2, we have

$$\begin{aligned} g(A \cup \{r\}) - g(A) &= |C_A \cup C_r| - (|A| + 1) - |C_A| + |A| \\ &= |C_A \cup C_r| - |C_A| - 1 \\ &= |C_r - C_A| - 1 \end{aligned}$$

where $C_r - C_A$ represents the set of vertices covered by r, but not by A.

Since $A \subseteq B$, we have $C_r - C_B \subseteq C_r - C_A$, hence $|C_r - C_B| \leq |C_r - C_A|$. Therefore, the submodular property holds. □

As our goal is to find a minimum set of reference nodes $\mathcal{R}^*$ to cover all vertices in V, it is equivalent to maximizing the gain function g:

$$\max_{\mathcal{R}} g(\mathcal{R}) = \max_{\mathcal{R}}(|\bigcup_{r \in \mathcal{R}} C_r| - |\mathcal{R}|) = |V| - \min_{\mathcal{R}} |\mathcal{R}| = g(\mathcal{R}^*)$$

In general, maximizing a submodular function is NP-hard [13]. So we resort to a greedy algorithm. It starts with an empty set of reference nodes $\mathcal{R}_0 = \emptyset$ with $g(\mathcal{R}_0) = 0$.

Then it iteratively selects a new reference node which maximizes an additional gain, as specified in Eq.(2). In particular, in the k-th iteration, it selects

$$r_k = \arg \max_{r \in V \setminus \mathcal{R}_{k-1}} g(\mathcal{R}_{k-1} \cup \{r\}) - g(\mathcal{R}_{k-1}) \tag{2}$$

The algorithm stops when all vertices in V are covered by the reference nodes. The greedy algorithm returns the reference node set $\mathcal{R}$.

Continue with our example. According to the greedy selection algorithm, in the first step, we will select r_1 as it has the highest gain. Given $\mathcal{R}_1 = \{r_1\}$, we have $g(\{r_1, r_2\}) - g(\{r_1\}) = 1$ and $g(\{r_1, r_3\}) - g(\{r_1\}) = 2$. So we will select r_3 in the second step. Finally we will select r_2 to cover the remaining vertices. Note that to simplify the illustration, we only consider selecting reference nodes from r_1, r_2, r_3 in this example. Our algorithm actually considers every graph vertex as a candidate for reference nodes.

To effectively control the size of $\mathcal{R}$, we can further relax the requirement to cover all vertices in V. We observe that such a requirement may cause $|\mathcal{R}|$ unnecessarily large, in order to cover the very sparse part of a graph or the isolated vertices. So we set a parameter *Cover Ratio* (CR), which represents the percentage of vertices to be covered. The above greedy algorithm terminates when a fraction of CR vertices in V are covered by $\mathcal{R}$.

3.2 Shortest Path Distance Estimation

Given $\mathcal{R}$, we will compute the shortest path distances for the node pairs $\{(r, v) | r \in \mathcal{R},\ v \in V\}$. This is realized by computing the single-source shortest paths for every $r \in \mathcal{R}$. Given a query node pair (s, t), we have

$$|D(s, r) - D(r, t)| \leq D(s, t) \leq D(s, r) + D(r, t)$$

for any $r \in \mathcal{R}$, according to the triangle inequality. Figure 2 shows an illustration of the shortest path distance estimation between (s, t), where the circle represents the area covered by a reference node r with a radius c. In this example, s is covered by r, while t is not.

By considering all reference nodes, we have tighter bounds

$$D(s, t) \leq \min_{r \in \mathcal{R}} (D(s, r) + D(r, t)) \tag{3}$$

and

$$D(s, t) \geq \max_{r \in \mathcal{R}} |D(s, r) - D(r, t)| \tag{4}$$

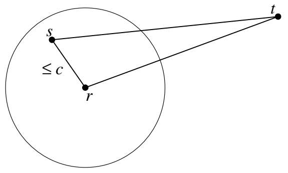

Fig. 2. Distance Estimation

Both the upper bound $\min_{r\in\mathcal{R}}(D(s,r)+D(r,t))$ and the lower bound $\max_{r\in\mathcal{R}} |D(s,r) - D(r,t)|$ can serve as an approximate estimation for $D(s,t)$. We denote them as $\widehat{D}_U$ and $\widehat{D}_L$, respectively. However, [4] reported that the upper bound achieves very good accuracy and performs far better than the lower bound in the internet network. We confirmed this observation on both a social network and a road network in our experiments. Thus we adopt the shortest path distance estimation as

$$\widehat{D}_U(s,t) = \min_{r\in\mathcal{R}}(D(s,r) + D(r,t))$$

3.3 Error Bound Analysis

In this section, we will show that, given a query (s,t), when s or t is covered within a radius c by some reference node from $\mathcal{R}$, the estimated distance $\widehat{D}_U(s,t)$ is within a bounded error of the true distance $D(s,t)$.

Theorem 2. *Given any query (s,t), the error of the estimated shortest path distance $\widehat{D}_U(s,t)$ can be bounded by $2c$ with a probability no smaller than $1-(1-CR)^2$, where c is the coverage radius and CR is the cover ratio.*

Proof. Given a query (s,t) and a reference node set $\mathcal{R}$, assume s is covered by a reference node, denoted as r^*, *i.e.*, $D(s,r^*) \le c$. Without loss of generality, we assume $D(s,r^*) \le D(r^*,t)$. Note that the following error bound still holds if $D(s,r^*) > D(r^*,t)$. The error of the estimated shortest path distance between (s,t) is bounded by

$$\begin{aligned} err(s,t) &= \widehat{D}_U(s,t) - D(s,t) \\ &= \min_{r\in\mathcal{R}}(D(s,r) + D(r,t)) - D(s,t) \\ &\le D(s,r^*) + D(r^*,t) - D(s,t) \\ &\le D(s,r^*) + D(r^*,t) - |D(s,r^*) - D(r^*,t)| \\ &= 2D(s,r^*) \\ &\le 2c \end{aligned}$$

The first inequality holds because $\min_{r\in\mathcal{R}}(D(s,r) + D(r,t)) \le D(s,r^*) + D(r^*,t)$; and the second inequality holds because we have the lower bound property $D(s,t) \ge |D(s,r^*) - D(r^*,t)|$.

The error bound holds when either s or t, or both are covered by some reference nodes. When neither s nor t is covered by some reference nodes within a radius c, $err(s,t)$ is unbounded. The probability for this case is $(1 - CR)^2$. However, in this case, if a reference node r happens to lie on the shortest path from s to t, we have the estimated distance $\widehat{D}_U(s,t) = D(s,t)$, *i.e.*, the error is still bounded by $2c$. Therefore, the probability that the error of an estimated distance is unbounded is at most $(1-CR)^2$. Thus we have $P(err(s,t) \le 2c) \ge 1 - (1 - CR)^2$. □

Given a user-specified error bound ϵ, we will have $P(err(s,t) \le \epsilon) \ge 1 - (1 - CR)^2$, when we set $c = \epsilon/2$.

3.4 Approximate Shortest Path Computation

With the shortest distance estimation, we propose a heuristic algorithm SPC to compute an approximate shortest path P for a query (s,t). The SPC algorithm works as follows: let $r = \arg\min_{v\in\mathcal{R}}(D(s,v)+D(v,t))$. We use such r to break down the path into two segments as $P(s,t) = SP(s,r)+SP(r,t)$. Here, $SP(s,r)$ represents the exact shortest path from s to r. To compute $SP(s,r)$ in linear time, we can follow the criterion

$$next(s) = \arg\min_{v\in N(s)}(D(s,v)+D(v,r))$$

where $N(s)$ denotes the the neighbor set of s and $next(s)$ denotes the successive neighbor of s that lies on the shortest path from s to r. Here we determine $next(s)$ based on the exact shortest distances $D(s,v)$ and $D(v,r)$. We iteratively apply the above criterion to find every vertex on the shortest path from s to r. Similarly, to compute $SP(r,t)$, we can follow the criterion

$$prev(t) = \arg\min_{v\in N(t)}(D(r,v)+D(v,t))$$

where $prev(t)$ is the preceding neighbor of t that lies on the shortest path from r to t.

The SPC algorithm computes an approximate shortest path whose distance equals $\widehat{D}_U(s,t)$. The time complexity is $O(|\mathcal{R}| + deg\cdot|P|)$, where $O(|\mathcal{R}|)$ is the time for finding the reference node r to break down the path, and deg is the largest vertex degree in the graph.

4 Graph Partitioning-Based Heuristic

For the reference node embedding method we propose above, the offline complexity is $O(|E| + |V|\log|V|)$ to compute the single-source shortest paths for a reference node $v\in\mathcal{R}$. It can be simplified as $O(n\log n)$ $(n = |V|)$ when the graph is sparse. Therefore, the total embedding time is $O(|\mathcal{R}|n\log n)$, which could be very expensive when $|\mathcal{R}|$ is large. In this section, we propose a graph partitioning-based heuristic for the reference node embedding to reduce the offline time complexity with a relaxed error bound. To distinguish the two methods we propose, we name the first method *RN-basic* and the partitioning-based method *RN-partition*.

4.1 Partitioning-Based Reference Node Embedding

The first step of RN-partition is reference node selection, which is the same as described in Section 3.1. In the second step, we use KMETIS [14] to partition the graph into K clusters $C_1,\ldots,C_K$. As a result, the reference node set $\mathcal{R}$ is partitioned into these K clusters. We use $\mathcal{R}_i$ to denote the set of reference nodes assigned to C_i, *i.e.*, $\mathcal{R}_i = \{r|r\in\mathcal{R}\ and\ r\in C_i\}$. It is possible that $\mathcal{R}_i=\emptyset$ for some i. For a cluster C_i with $\mathcal{R}_i=\emptyset$, we can select the vertex from C_i with the largest degree as a within-cluster reference node, to improve the local coverage within C_i. Note that the number of such within-cluster reference nodes is bounded by the number of clusters K, which is a small number compared with $|\mathcal{R}|$.

The idea of the partitioning-based reference node embedding is as follows. For the cluster C_i, we compress all reference nodes in $\mathcal{R}_i$ as a supernode SN_i and then compute the single-source shortest paths from SN_i to every vertex $v \in V$. The reference node compression operation is defined as follows.

Definition 4 (Reference Node Compression). *The reference node compression operation compresses all reference nodes in $\mathcal{R}_i$ into a supernode SN_i. After compression, for a vertex $v \in V \setminus \mathcal{R}_i$, $(SN_i, v) \in E$ iff $\exists r \in \mathcal{R}_i$, s.t. $(r, v) \in E$, and the edge weight is defined as $w(SN_i, v) = \min_{r \in \mathcal{R}_i} w(r, v)$.*

Then the shortest path between SN_i and v is actually the shortest path between a reference node $r \in \mathcal{R}_i$ and v with the smallest shortest path distance, *i.e.*,

$$D(SN_i, v) = \min_{r \in \mathcal{R}_i} D(r, v)$$

and we denote the closest reference node $r \in \mathcal{R}_i$ to v as $r_{v,i}$, which is defined as

$$r_{v,i} = \arg\min_{r \in \mathcal{R}_i} D(r, v)$$

Note $D(SN_i, v) = D(r_{v,i}, v) = \min_{r \in \mathcal{R}_i} D(r, v)$. In the following, we will use $D(SN_i, v)$ and $D(r_{v,i}, v)$ interchangeably.

The time complexity for computing shortest paths from the supernodes in each of the K clusters to all the other vertices in V is $O(Kn \log n)$. In addition, we compute the shortest path distances between every pair of reference nodes within the same cluster. The time complexity of this operation is $O(|\mathcal{R}| n/K \log n/K)$, if we assume the nodes are evenly partitioned into K clusters. We further define the diameter d for a cluster as follows.

Definition 5 (Cluster Diameter). *Given a cluster C, the diameter d is defined as the maximum shortest distance between two reference nodes in C, i.e.,*

$$d = \max_{r_i, r_j \in C} D(r_i, r_j)$$

where $D(r_i, r_j)$ is the shortest distance between r_i and r_j.

Then the diameter of the partitioning $C_1, \ldots, C_K$ is defined as the maximum of the K cluster diameters, *i.e.*,

$$d_{max} = \max_{i \in [1,K]} d_i$$

4.2 Partitioning-Based Shortest Path Distance Estimation

Given a query (s, t), for the supernode SN_i representing a cluster C_i, based on the triangle inequality we have

$$D(s, t) \leq D(s, SN_i) + D(r_{s,i}, r_{t,i}) + D(t, SN_i)$$

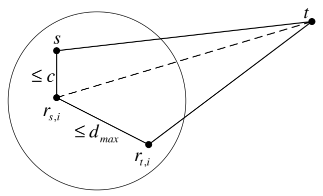

Fig. 3. Distance Estimation in RN-partition

Figure 3 shows an illustration of the shortest path distance estimation between (s, t) in RN-partition, where the circle represents a cluster C_i. Note that in general s and $r_{s,i}$ may not necessarily belong to the same cluster, and the shortest path distance $D(s, r_{s,i})$ may not necessarily be bounded by the radius c. But these factors will not affect the distance estimation strategy.

By considering all K clusters, we have a tighter upper bound

$$D(s,t) \leq \min_{i \in [1,K]} (D(s, SN_i) + D(r_{s,i}, r_{t,i}) + D(t, SN_i))$$

We denote this estimated distance upper bound as $\widehat{D}_U^P(s,t)$.

4.3 Error Bound Analysis

In the following theorem, we will show that, when s or t is covered within a radius c by some reference node from a cluster C_i for some i, the estimated distance $\widehat{D}_U^P(s,t)$ is within a bounded error of the true distance $D(s,t)$.

Theorem 3. *Given any query (s,t), the error of the estimated shortest path distance $\widehat{D}_U^P(s,t)$ by RN-partition can be bounded by $2(c + d_{max})$ with a probability no smaller than $1 - (1 - CR)^2$, where c is the coverage radius, CR is the cover ratio and d_{max} is the maximum cluster diameter.*

Proof. Given a query (s,t), assume s is covered by at least one reference node from $\mathcal{R}$ within a radius c. Without loss of generality, assume such a reference node is from the cluster C_i for some i and denote it as $r_{s,i}$. According to the triangle inequality, we have

$$D(r_{s,i}, t) - D(s,t) \leq D(s, r_{s,i}) \leq c$$

By adding $D(s, r_{s,i})$ on both sides, we have

$$D(s, r_{s,i}) + D(r_{s,i}, t) - D(s,t) \leq 2D(s, r_{s,i}) \leq 2c \tag{5}$$

Denote the closest reference node in C_i to t as $r_{t,i}$. Then we have

$$D(r_{t,i}, t) - D(r_{s,i}, t) \leq D(r_{s,i}, r_{t,i}) \leq d_{max}$$

Since $r_{s,i}, r_{t,i}$ belong to the same cluster, their distance is bounded by d_{max}. By adding $D(r_{s,i}, r_{t,i})$ on both sides, we have

$$D(r_{s,i}, r_{t,i}) + D(r_{t,i}, t) - D(r_{s,i}, t) \leq 2D(r_{s,i}, r_{t,i}) \leq 2d_{max} \tag{6}$$

By adding Eq.(5) and Eq.(6), we have

$$D(s, r_{s,i}) + D(r_{s,i}, r_{t,i}) + D(r_{t,i}, t) - D(s, t) \le 2(c + d_{max})$$

As we have defined $\widehat{D}^P_U(s,t) = \min_{i \in [1,K]}(D(s, SN_i) + D(r_{s,i}, r_{t,i}) + D(t, SN_i))$, the error of the estimated shortest path distance between (s,t) is bounded by

$$\begin{aligned} err^P(s,t) &= \widehat{D}^P_U(s,t) - D(s,t) \\ &\le D(s, r_{s,i}) + D(r_{s,i}, r_{t,i}) + D(r_{t,i}, t) - D(s,t) \\ &\le 2(c + d_{max}) \end{aligned}$$

The error bound holds when either s or t, or both are covered by some reference nodes with a radius c. When it happens that neither s nor t is covered by some reference nodes, Eq.(5) does not hold in general, thus $err^P(s,t)$ is unbounded. The probability for this case is $(1 - CR)^2$. For a similar reason as explained in Theorem 2, *i.e.*, even when neither s nor t is covered, if there are reference nodes $r_{s,i}$, $r_{t,i}$, for some i, on the shortest path from s to t, we can still have an accurate estimation which satisfies the error bound. Therefore the probability that the error of an estimated distance is unbounded is at most $(1 - CR)^2$. Thus, we have $P(err^P(s,t) \le 2(c + d_{max})) \ge 1 - (1 - CR)^2$ □

Compared with RN-basic, RN-partition reduces the offline computational complexity to $O(Kn \log n + |\mathcal{R}|n/K \log n/K)$. As long as we choose a reasonably large K such that $|\mathcal{R}|/K \le K$, the complexity of RN-partition is dominated by $O(Kn \log n)$. As a tradeoff, the error bound is relaxed from $2c$ to $2(c + d_{max})$. The cluster diameter d_{max} is determined by the size of the graph and the number of clusters K. Table 1 compares RN-basic and RN-partition on time/space complexity and the error bound. In experimental study, we will study the relationship between K, the offline computation time and the accuracy of the estimated distances.

Table 1. Comparison between RN-basic and RN-partition

	RN-basic	**RN-partition**
Offline Time Complexity	$O(\|\mathcal{R}\|n \log n)$	$O(Kn \log n + \|\mathcal{R}\|n/K \log n/K)$
Offline Space Complexity	$O(\|\mathcal{R}\|n)$	$O(Kn + \|\mathcal{R}\|^2/K)$
Distance Query Complexity	$O(\|\mathcal{R}\|)$	$O(K)$
Error Bound	$2c$	$2(c + d_{max})$

5 Experiments

We performed extensive experiments to evaluate our algorithms on two types of networks – a road network and a social network. The road network and the social network exhibit quite different properties on: (1) degree distribution, *i.e.*, the former roughly follows a uniform distribution while the latter follows a power law distribution; and (2) network diameter, *i.e.*, the social network has the shrinking diameter property [15] and the small world phenomenon, which, however, do not hold in the road network. All experiments were performed on a Dell PowerEdge R900 server with four 2.67GHz six-core CPUs and 128GB main memory running Windows Server 2008. All algorithms were implemented in Java.

5.1 Comparison Methods and Evaluation

We compare our methods RN-basic and RN-partition with two existing methods:

- **2RNE** [8] by Kriegel et al. uses a two level reference node embedding which examines K nearest reference nodes for both nodes in a query to provide a distance estimation. We select reference nodes uniformly and set $K = 3$.
- **Centrality** [9] by Potamias et al. selects reference nodes with low closeness centrality. According to [9], the approximate centrality measure is computed by selecting a sample of S random seeds, where we set $S = 10,000$ in our implementation.

For a node pair (s, t), we use the relative error to evaluate the quality of the estimated distance

$$rel\_err(s,t) = \frac{|\widehat{D}_U(s,t) - D(s,t)|}{D(s,t)}$$

As it is expensive to exhaustively evaluate all node pairs in a large network, we randomly sample a set of $10,000$ node pairs in the graph as queries and evaluate the average relative error on the sample set.

5.2 Case Study 1: Road Network

We use the New York City road network, which is an undirected planar graph with $264,346$ nodes and $733,846$ edges. A node here represents an intersection or a road endpoint while the weight of an edge represents the length of the corresponding road segment. The data set can be downloaded from http://www.dis.uniroma1.it/∼challenge9/.

The degrees of most nodes in the road network fall into the range of $[1, 4]$ and the network has no small world phenomenon. For the $10,000$ random queries we generate, we plot the histogram of the shortest path distance distribution in Figure 4. The average distance over the $10,000$ queries is $d_{avg} = 26.68$KM. So if we set the radius $c = 0.8$KM, the average relative error can be roughly bounded by $2c/d_{avg} = 0.06$.

Parameter Sensitivity Test on CR. In this experiment, we vary the cover ratio CR and compare the average error, the reference node set size and offline index time by RN-basic and RN-partition with $K = 100, 250, 500$, respectively. We fix the radius $c = 0.8$KM.

Figure 6 shows the average error of RN-basic and RN-partition with different K values. The average error of RN-basic is below 0.01 and slightly decreases as CR increases. The average error of RN-partition decreases very sharply when the number of partitions K increases and becomes very close to that of RN-basic when $K = 500$.

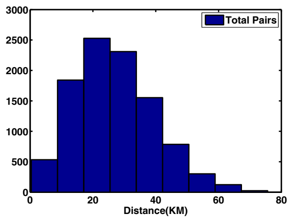

Fig. 4. Shortest Distance Distribution on Road Network

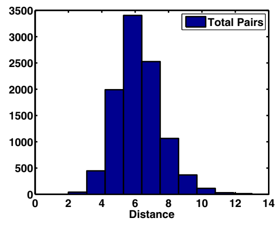

Fig. 5. Shortest Distance Distribution on Social Network

Figure 7 shows that the number of reference nodes $|\mathcal{R}|$ increases linearly with CR. As RN-basic and RN-partition have the same reference node selection process, the number is the same for both methods. When $CR = 1.0$, we need $9,000$ reference nodes to cover the road network with $264,346$ nodes.

Figure 8 shows the offline index time in logarithmic scale for RN-basic and RN-partition to compute the single-source shortest paths from every reference node. RN-partition reduces the index time of RN-basic by one order of magnitude. In addition, as the number of reference nodes $|\mathcal{R}|$ increases linearly with CR, the index time of RN-basic also increases linearly with CR, because the time complexity is $O(|\mathcal{R}|n \log n)$. On the other hand, the index time of RN-partition remains quite stable as CR increases, because RN-partition only computes the shortest paths from each of the K clusters as the source.

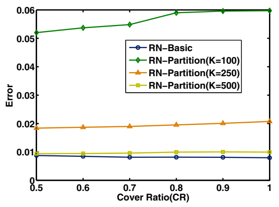

Fig. 6. Average Error vs. Cover Ratio (CR) on Road Network

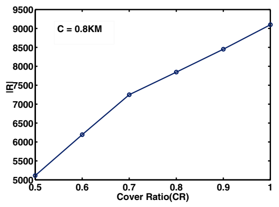

Fig. 7. Reference Node Set Size vs. Cover Ratio (CR) on Road Network

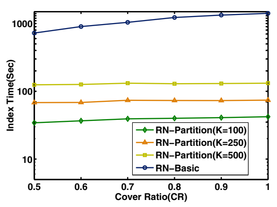

Fig. 8. Index Time vs. Cover Ratio (CR) on Road Network

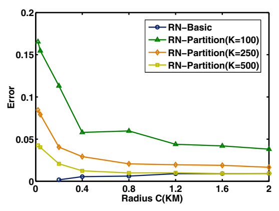

Fig. 9. Average Error vs. Radius (c) on Road Network

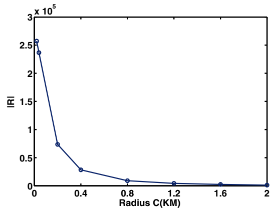

Fig. 10. Reference Node Set Size vs. Radius (c) on Road Network

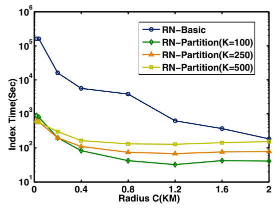

Fig. 11. Index Time vs. Radius (c) on Road Network

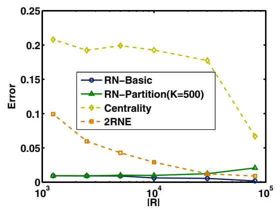

Fig. 12. Average Error vs. $|\mathcal{R}|$ on Road Network

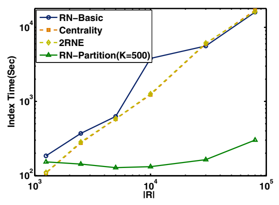

Fig. 13. Index Time vs. $|\mathcal{R}|$ on Road Network

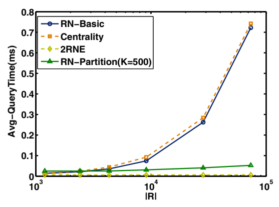

Fig. 14. Average Query Time vs. $|\mathcal{R}|$ on Road Network

Parameter Sensitivity Test on c. In this experiment, we vary the radius c and compare the average error, the reference node set size and offline index time by RN-basic and RN-partition with $K = 100, 250, 500$, respectively. We fix the cover ratio $CR = 1.0$.

Figure 9 shows the average error of RN-basic and RN-partition with different K values. We can make the following observations from the figure: (1) RN-partition ($K = 500$) achieves an average error very close to that of RN-basic when $c \geq 0.8$KM; (2) The average error of RN-basic monotonically increases with c, which is consistent with the theoretical error bound of $2c$; and (3) Different from RN-basic, the average error of RN-partition shows a decreasing trend with c. When c is very small, the number of reference nodes is very large. So RN-partition may choose suboptimal reference nodes for distance estimation, which leads to a larger error.

Figure 10 shows that the number of reference nodes $|\mathcal{R}|$ decreases with c. When $c < 0.4$KM, $|\mathcal{R}|$ decreases sharply with c. Figure 11 shows the offline index time of RN-basic and RN-partition in logarithmic scale. As $|\mathcal{R}|$ decreases with c, the index time of RN-basic also decreases with c. RN-partition reduces the index time of RN-basic by two orders of magnitude or more when $c < 0.2$KM but the difference becomes smaller as c increases. RN-basic cannot finish within 10 hours when $c \leq 0.08$KM. On the other hand, the index time of RN-partition increases moderately when c decreases to 0.2KM or below.

Comparison with 2RNE and Centrality. We compare our approaches with 2RNE [8] and Centrality [9] in terms of average error, index time and average query time, as we vary the number of reference nodes. For our methods, we set $CR = 1.0$. From Figure 12 we can see that both RN-basic and RN-partition (for most cases) outperform 2RNE and Centrality by a large margin in terms of average error. Figure 13 shows that RN-partition reduces the index time of the other three methods by up to two orders of magnitude. The index time of RN-basic, 2RNE and Centrality increases linearly with $|\mathcal{R}|$, as they all have the same time complexity of $O(|\mathcal{R}|n \log n)$, while RN-partition slightly increases the index time. Figure 14 shows that the query time of RN-partition and 2RNE remain almost constant, while that of RN-basic and Centrality increase linearly with the number of reference nodes.

Shortest Path Query Processing. In this experiment, we evaluate the efficiency and quality of the SPC procedure for computing the shortest paths. For comparison, we implemented A^* algorithm using $\widehat{D}_L$ as the h function, since it provides a lower bound

Table 2. Comparison between SPC and A^* on Road Network

	Average Error	Average Query Time (millisec)
SPC	0.012	0.19
A^*	0	141.79

distance estimation. We evaluate both methods on the $10,000$ random queries. We set $|\mathcal{R}| = 20,000$. Table 2 shows that SPC finds approximate shortest paths with an average error of 0.012 while A^* computes the exact shortest paths. But SPC is about 750 times faster than A^*, since it is a linear algorithm.

5.3 Case Study 2: Social Network

We download the DBLP dataset from http://dblp.uni-trier.de/xml/ and construct an undirected coauthor network, where a node represents an author, an edge represents a coauthorship relation between two authors, and all edge weights are set to 1. This graph has several disconnected components and we choose the largest connected one which has $629,143$ nodes and $4,763,500$ edges. The vertex degree distribution follows the power law distribution.

We randomly generate $10,000$ queries and plot the histogram of the shortest path distance distribution in Figure 5. The average distance between two nodes over the $10,000$ queries is $d_{avg} = 6.34$, which conforms with the famous social networking rule "six degrees of separation". Given $2c/d_{avg}$ as a rough estimation of the relative error bound, if we set $c = 3$, the relative error bound is $2 \times 3/6.34 = 94.64\%$. Therefore, we only test our methods given $c \in \{1, 2\}$, to control the relative error bound in a reasonably small range. Note that $c = 1$ defines the coverage of a node based on the number of its neighbors, *i.e.*, degree; while $c = 2$ measures the coverage based on the number of neighbors within two hops.

Parameter Sensitivity Test on CR. We vary the cover ratio CR and compare the average error, the reference node set size and offline index time by RN-basic and RN-partition with $K = 100, 200, 300$, respectively. We fix the radius $c = 1$.

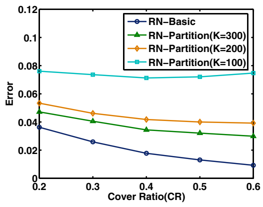

Fig. 15. Average Error vs. Cover Ratio (CR) on Social Network

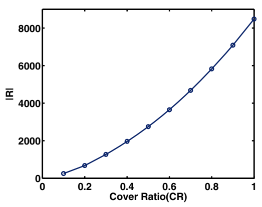

Fig. 16. Reference Node Set Size vs. Cover Ratio (CR) on Social Network

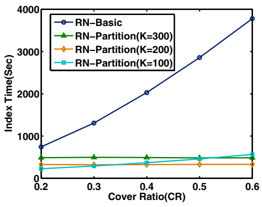

Fig. 17. Index Time vs. Cover Ratio (CR) on Social Network

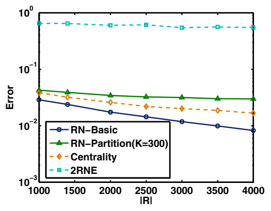

Fig. 18. Average Error vs. $|\mathcal{R}|$ on Social Network

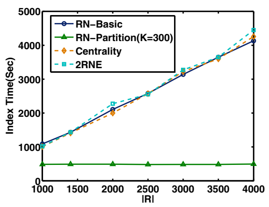

Fig. 19. Index Time vs. $|\mathcal{R}|$ on Social Network

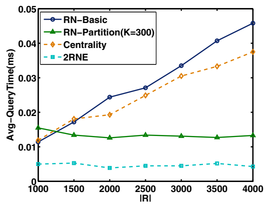

Fig. 20. Average Query Time vs. $|\mathcal{R}|$ on Social Network

Figure 15 shows that the average error of RN-basic is in the range of $[0.009, 0.04]$ and it decreases quickly as CR increases. The average error of RN-partition is slightly higher than that of RN-basic and it decreases as K increases.

Figure 16 shows the number of reference nodes $|\mathcal{R}|$ as we vary CR in the range of $[0, 1.0]$. Different from the road network which shows a linear relationship between $|\mathcal{R}|$ and CR, we observe that $|\mathcal{R}|$ increases slowly when CR is small, but much faster when CR is large. This is due to the power law degree distribution in the social network – we first select the authors with the largest number of collaborators as reference nodes; but in the later stage, with the decrease of node degrees, we need to use more reference nodes to achieve the same amount of coverage.

Figure 17 shows the offline index time for RN-basic and RN-partition. We observe that the index time of RN-basic increases quickly when CR increases. When $CR = 0.6$, RN-basic is about 10 times slower than RN-partition. We also observe that the index time of RN-partition slightly increases with CR when $K = 100$. This is because a large portion of time is spent on computing the shortest path distances between all pairs of reference nodes within the same partition. When CR increases, the number of reference nodes falling into the same partition is larger, which causes the time increase.

Parameter Sensitivity Test on *c*. In this experiment, we vary the radius $c \in \{1, 2\}$ and compare the average error, the reference node set size and offline index time by RN-basic and RN-partition ($K = 300$). We fix $CR = 0.6$. Table 3 shows that the number of reference nodes is reduced by 100 times when c is increased to 2. As a result, the offline index time for RN-basic is also reduced by 100 times with the increase of c because the time complexity is $O(|\mathcal{R}|n \log n)$. The index time for RN-partition is seven times smaller than RN-basic when $c = 1$, but slightly higher when $c = 2$ due to the within partition computational overhead. The average error of RN-partition is slightly higher than that of RN-basic, and the error of both methods increases with c, which is consistent with the theoretical error bound.

Table 3. Parameter Sensitivity Test on Radius c on Social Network

		RN-basic		RN-partition	
Radius c	$\|\mathcal{R}\|$	**Average Error**	**Index Time (sec)**	**Average Error**	**Index Time (sec)**
1	3653	0.009	3778.17	0.030	485.88
2	31	0.138	30.70	0.144	65.71

Comparison with 2RNE and Centrality. We compare our approaches with 2RNE and Centrality in terms of average error, index time and average query time, as we vary the number of reference nodes. For our methods, we set $c = 1$. Figure 18 shows that RN-basic achieves the smallest error, followed by Centrality and RN-partition. 2RNE performs the worst, because it selects reference nodes uniformly, rather than selecting reference nodes with large degrees. Figure 19 shows that the index time of RN-partition remains stable when $|\mathcal{R}|$ increases, while the time of the other three methods increases linearly with $|\mathcal{R}|$. Figure 20 shows that the query time of RN-partition and 2RNE remain almost constant, while that of RN-basic and Centrality increase linearly with the number of reference nodes.

Shortest Path Query Processing. We compare SPC with A^* on shortest path query on the $10,000$ random queries on the DBLP network. We set $|\mathcal{R}| = 4,000$. Table 4 shows that SPC finds approximate shortest paths with an average error of 0.008 while A^* computes the exact shortest paths. But SPC is about $100,000$ times faster than A^*.

Table 4. Comparison between SPC and A^* on Social Network

	Average Error	**Average Query Time (millisec)**
SPC	0.008	0.046
A^*	0	4469.28

6 Related Work

Dijkstra's algorithm [1] computes the single-source shortest path in a graph with non-negative edge weights with a time complexity of $O(|E| + |V| \log |V|)$. The Floyd-Warshall algorithm [16] computes the shortest paths between all pairs of vertices with a dynamic programming approach. Its time complexity is $O(|V|^3)$. The A^* search algorithm [2,17] uses some heuristics to direct the search direction.

In the literature, graph embedding techniques have been widely used to estimate the distance between two nodes in a graph in many applications including road networks [5,8], social networks and web graphs [7,9,11,12] and the Internet [3,4]. Kriegel et al. [8] proposes a hierarchical reference node embedding algorithm for shortest distance estimation. Potamias et al. [9] formulates the reference node selection problem to selecting vertices with high betweenness centrality. [3] proposes an architecture, called IDMaps which estimates the distance in the Internet and a related work [4] proposes a Euclidean embedding approach to model the Internet. [6] defines a notion of slack – a certain fraction of all distances that may be arbitrarily distorted as a performance guarantee based on randomly selected reference nodes. [10] and its follow up studies [11,12] provide a relative $(2k - 1)$-approximate distance estimation with $O(kn^{1+1/k})$ memory for any integer $k \geq 1$. A limitation of many existing methods is that, the estimated shortest path distance has no error bound, thus it is hard to guarantee the estimation quality. In contrast, our approach provides an absolute error bound of the distance estimation by $2c$ in RN-basic or by $2(c + d_{max})$ in RN-partition.

Computing shortest paths and processing k-nearest neighbor queries in spatial networks have also received a lot of attention. Papadias et al. [18] propose to use the Euclidean distance as a lower bound to prune the search space and guide the network expansion for refinement. [19] uses first order Voronoi diagram to answer KNN queries in spatial networks. Hu et al. [20] propose an index, called distance signature, which associates approximate distances from one object to all the other objects in the network, for distance computation and query processing. Samet et al. [21] build a shortest path quad tree to support k-nearest neighbor queries in spatial networks. [22] proposes TEDI, an indexing and query processing scheme for the shortest path query based on tree decomposition.

7 Conclusions

In this paper, we propose a novel coverage-based reference node embedding approach to answer shortest path and shortest path distance queries with a theoretical error bound. Our methods achieve very accurate distance estimation on both a road network and a social network. The RN-basic method provides very accurate distance estimation, while the RN-partition method reduces the offline embedding time of RN-basic by up to two orders of magnitude or more with a slightly higher estimation error. In addition, our methods outperform two state-of-the-art reference node embedding methods in processing shortest path distance queries.

Acknowledgment

The work was supported in part by grants of the Research Grants Council of the Hong Kong SAR, China No. 419109, No. 419008 and No. 411310, and the Chinese University of Hong Kong Direct Grants No. 2050446.

References

1. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische Mathematik 1(1), 269–271 (1959)
2. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics SSC4 4(2), 100–107 (1968)
3. Francis, P., Jamin, S., Jin, C., Jin, Y., Raz, D., Shavitt, Y., Zhang, L.: IDMaps: A global internet host distance estimation service. IEEE/ACM Trans. Networking 9(5), 525–540 (2001)
4. Ng, E., Zhang, H.: Predicting internet network distance with coordinates-based approaches. In: INFOCOM, pp. 170–179 (2001)
5. Shahabi, C., Kolahdouzan, M., Sharifzadeh, M.: A road network embedding technique for k-nearest neighbor search in moving object databases. In: GIS, pp. 94–100 (2002)
6. Kleinberg, J., Slivkins, A., Wexler, T.: Triangulation and embedding using small sets of beacons. In: FOCS, pp. 444–453 (2004)
7. Rattigan, M.J., Maier, M., Jensen, D.: Using structure indices for efficient approximation of network properties. In: KDD, pp. 357–366 (2006)

8. Kriegel, H.-P., Kröger, P., Renz, M., Schmidt, T.: Hierarchical graph embedding for efficient query processing in very large traffic networks. In: Ludäscher, B., Mamoulis, N. (eds.) SSDBM 2008. LNCS, vol. 5069, pp. 150–167. Springer, Heidelberg (2008)
9. Potamias, M., Bonchi, F., Castillo, C., Gionis, A.: Fast shortest path distance estimation in large networks. In: CIKM, pp. 867–876 (2009)
10. Thorup, M., Zwick, U.: Approximate distance oracles. Journal of the ACM 52(1), 1–24 (2005)
11. Gubichev, A., Bedathur, S., Seufert, S., Weikum, G.: Fast and accurate estimation of shortest paths in large graphs. In: CIKM, pp. 499–508 (2010)
12. Sarma, A.D., Gollapudi, S., Najork, M., Panigrahy, R.: A sketch-based distance oracle for web-scale graphs. In: WSDM, pp. 401–410 (2010)
13. Khuller, S., Moss, A., Naor, J.: The budgeted maximum coverage problem. Information Processing Letters 70, 39–45 (1999)
14. Karypis, G., Kumar, V.: A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM Journal on Scientific Computing 20(1), 359–392 (1999)
15. Leskovec, J., Kleinberg, J., Faloutsos, C.: Graph evolution: Densification and shrinking diameters. ACM Trans. Knowledge Discovery from Data 1(1) (2007)
16. Floyd, R.W.: Algorithm 97: Shortest path. Communications of the ACM 5(6), 345 (1962)
17. Goldberg, A.V., Harrelson, C.: Computing the shortest path: A* search meets graph theory. In: SODA, pp. 156–165 (2005)
18. Papadias, D., Zhang, J., Mamoulis, N., Tao, Y.: Query processing in spatial network database. In: VLDB, pp. 802–813 (2003)
19. Kolahdouzan, M., Shahabi, C.: Voronoi-based k nearest neighbor search for spatial network databases. In: VLDB, pp. 840–851 (2004)
20. Hu, H., Lee, D.L., Lee, V.C.S.: Distance indexing on road networks. In: VLDB, pp. 894–905 (2006)
21. Samet, H., Sankaranarayanan, J., Alborzi, H.: Scalable network distance browsing in spatial databases. In: SIGMOD, pp. 43–54 (2008)
22. Wei, F.: TEDI: Efficient shortest path query answering on graphs. In: SIGMOD, pp. 99–110 (2010)