Distance Oracles for Spatial Networks

Jagan Sankaranarayanan Hanan Samet[1]

Center for Automation Research
Institute for Advanced Computer Studies
Department of Computer Science
University of Maryland, College Park, MD 20742
{*jagan,hjs*}*@cs.umd.edu*

***Abstract*— The popularity of location-based services and the need to do real-time processing on them has led to an interest in performing queries on transportation networks, such as finding shortest paths and finding nearest neighbors. The challenge is that these operations involve the computation of distance along a spatial network rather than "as the crow flies." In many applications an estimate of the distance is sufficient, which can be achieved by use of an oracle. An approximate distance oracle is proposed for spatial networks that exploits the coherence between the spatial position of vertices and the network distance between them. Using this observation, a distance oracle is introduced that is able to obtain the ε-approximate network distance between two vertices of the spatial network. The network distance between every pair of vertices in the spatial network is efficiently represented by adapting the well-separated pair technique to spatial networks. Initially, use is made of an ε-approximate distance oracle of size $O(\frac{n}{\varepsilon^d})$ that is capable of retrieving the approximate network distance in $O(\log n)$ time using a B-tree. The retrieval time can be theoretically reduced further to $O(1)$ time by proposing another ε-approximate distance oracle of size $O(\frac{n \log n}{\varepsilon^d})$ that uses a hash table. Experimental results indicate that the proposed technique is scalable and can be applied to sufficiently large road networks. A 10%-approximate oracle ($\varepsilon = 0.1$) on a large network yielded an average error of 0.9% with 90% of the answers making an error of 2% or less and an average retrieval time of 68μ seconds. Finally, a strategy for the integration of the distance oracle into any relational database system as well as using it to perform a variety of spatial queries such as region search, k-nearest neighbor search, and spatial joins on spatial networks is discussed.**

I. Introduction

The popularity of web-based mapping applications such as Mapquest, Yahoo Maps, and the subsequent enhancements available in Google Maps and Microsoft Live Search, as well as the increasing pervasiveness of GPS-enabled devices such as PDAs, have led to an expectation of real-time execution for queries on transportation networks, such as computing shortest paths and finding nearest objects from a set S (e.g., restaurants, department stores, and gas stations). For example, suppose that we found the shortest distance from gas station A to the nearest restaurant B, which serves Italian food, and we wish to determine how much farther it is to go to another restaurant C, which serves Chinese food.

The challenge is that these operations involve the computation of distance along a spatial network rather than "as the crow flies." This computation is usually the byproduct of the use of shortest path algorithms to accumulate the node to node distances. The requirement that these distances be computed in real-time precludes the use of conventional graph-based algorithms (e.g., the INE and IER methods [1] and improvements on them [2], and hierarchical graph methods [3], [4]) which usually incorporate Dijkstra's shortest-path algorithm [5] in at least some parts of the solution [6]. It is well-known that the problem with Dijkstra's algorithm is that although it reports the shortest path from a starting vertex s to every other vertex v in increasing order of distance from v, it must visit every vertex that is closer to s via the shortest path from s than the vertices associated with the desired objects in S. Thus it ends up visiting a very large number of the vertices, even though the shortest paths to the objects in S do not pass through them.

A drastic alternative to the use of Dijkstra's algorithm is to precompute and store the shortest paths between all possible vertices in the spatial network. The drawback of this approach is that, for n vertices, the amount of storage could be as high as $O(n^3)$. The necessary storage can be reduced to $O(n^2)$ by taking advantage of the fact that the shortest paths from vertex u to all remaining vertices can be decomposed into subsets based on the first edges on the shortest paths to them from u [7], [8], [10]. This comes at the cost of a slower process of retrieving the shortest path which makes use of a sequence of point location operations (e.g., [9]). We have shown that the storage necessary for these subsets can be reduced substantially further to $O(n^{1.5})$ [7], [8] by noting the spatial coherence of the subsets and representing them using a shortest-path quadtree, which is a variant of the region quadtree, where the blocks are decomposed until all vertices in the block are in the same subset. Note that the use of the quadtree in that context is primarily to take advantage of its dimension-reducing property [11] to decrease the storage requirements instead of for speeding up operations such as ray tracing [12], [13].

The above algorithms exploit the spatial coherence of the destination vertices of the spatial network to reduce the storage

[1]This work was supported in part by the National Science Foundation under Grants EIA-00-91474, CCF-05-15241, and IIS-07-13501, as well as NVIDIA Corporation, Microsoft Research, the E.T.S. Walton Visitor Award of the Science Foundation of Ireland, and the National Center for Geocomputation at the National University of Ireland at Maynooth.

1084-4627/09 $25.00 © 2009 IEEE
DOI 10.1109/ICDE.2009.53

requirements of the collection of precomputed shortest paths from a specified source vertex. In this paper we continue our work [7] by showing how to also take advantage of the spatial coherence of the source vertices to further reduce the space requirements. In particular, we observe that given a set of source vertices A and a set of destination vertices B such that A and B are *sufficiently* far away from each other, while the vertices comprising them are close to one another, then the shortest paths between them may share common vertices, which in turn implies that the network distance between any source vertex in A to any destination vertex in B will more or less be the same. Figure 1 is an example of such a configuration where all the 30,000 shortest paths between vertices in A and in B have many vertices in common, while the network distances between them can be approximated by a single value.

Figure 1: The 30,000 shortest paths between all pairs of vertices in sets A and B in the spatial network of Silver Spring, MD are marked in a darker shade. These network distances can be approximated by a single value as their shortest paths have many vertices in common.

The techniques that we develop in this paper are based on our inference that given our assumptions on the proximity of the vertices that comprise A and those that comprise B, and the lack of proximity between A and B, that the network distance between the vertices in A to B will more or less be similar and can be approximated by a single value. The novelty of our approach is that in the case of the computation of the distance between two vertices, we show how to correlate the extent of this reduction of the space requirements with the approximation error in the value of the distance that is obtained. This is achieved via the introduction of a more general construct, termed an *approximate distance oracle* for spatial networks, that is capable of responding to network distance queries between any two vertices of the spatial network with a specified approximation ε—that is, given a start vertex s and a destination vertex w in spatial network G, the network distance $S_\varepsilon(s,w)$ produced by the oracle S_ε is no more or less than an ε fraction of the actual network distance $d_G(s,w)$ between s and w in G.

The utility of the oracle, as can be inferred from Figure 1, is that the network distance between all vertices of a pair of collections of spatially coherent source vertices and spatially coherent destination vertices can be approximated by a single value. This is based on our observation that the distance distortion (i.e., the ratio of the network distance to the spatial distance between two vertices in a spatial network) decreases as the separation between the vertices increases and our demonstration that it has a reasonable bound. Assuming d-dimensional data (usually $d = 2$), the latter, coupled with the path coherence property, enable us to devise an $O(\frac{n}{\varepsilon^d})$ size oracle represented using a B-tree that is capable of retrieving the ε-approximate network distance in $O(\log n)$ with a deterministic guarantee on the error. We also present a theoretical analysis of an alternative oracle that can further reduce the retrieval time further to $O(1)$ with an increase in space to $O(\frac{n \log n}{\varepsilon^d})$ using a hash table.

We achieve our results by showing how to efficiently represent the network distance between the spatially coherent collections of source vertices and the spatially coherent collections of destination vertices. This is done by adapting the notion of a well-separated decomposition of a point set, originally proposed by Callahan and Kosaraju [14] and used by others (e.g., [15], [16]) for a point set, to a spatial network.

It is important to note that one of the ways of evaluating the significance of our work lies in determining the extent to which we can improve on the storage costs that we obtained in our earlier work [7] where we represent the spatially coherent destination vertices by a shortest-path quadtree, which had a factor of $O(n^{0.5})$. Clearly, if we choose ε to be very small, then our space requirements, which, although appearing to be linear (i.e., $O(n/\varepsilon^d)$), will become sufficiently large to counteract any advantage drawn from the use of this linear-size oracle. However, the execution time of the distance computation process is also quite fast when using the linear-size oracle as the shortest-path quadtree method [7] does not explicitly store the distances between the vertices (instead, it stores distance intervals), and thus whenever it wants to compute the distance between a pair of vertices, it must compute the shortest path between them (with a possible halt once the required approximation error threshold is attained). This usually involves a large number of refinement operations, which can be slow.

At this point, we mention a few related methods, but we first present a few definitions. A spatial network can be abstracted to form an equivalent graph representation $G = (V,E)$, where V is the set of vertices, E is the set of edges, $n = |V|$, and $m = |E|$. Given $e \in E$, $w(e) \geq 0$ denotes the distance along that edge. In addition, for every $v \in V$, $p(v)$ denotes the spatial position of v with respect to S, a *spatial domain*, also referred to as an *embedding space* (*i.e.*, a reference coordinate system). We define the *network distance* $d_G(u,v)$ to be the distance along the shortest path between u and v in the spatial network. Similarly, we define the spatial distance $d_S(u,v)$ to be a function of the position of the vertices u and v on the embedding plane. For example, in the case of a road network the network distance between two vertices is the shortest distance in miles, or the time taken to travel the road network, while the spatial distance (e.g., "crow flying" distance) is a function of latitude/longitude positions of the vertices.

Furthermore, we assume that for some spatial networks (e.g., the road networks), the network distance between any two vertices is bounded from above and below by two constants γ_L, γ_H (presumably large), such that

$$\gamma_L \leq \frac{d_G(u,v)}{d_S(u,v)} \leq \gamma_H;\ \gamma_L, \gamma_H > 0.$$

The constants γ_L, γ_H are termed the minimum and maximum distortions of G. Narasimhan and Smid [17] provide a simple technique for estimating the value of γ_H for Euclidean networks which is easy to adapt to spatial networks. Our experiments show γ_H to be large for road networks.

The technique that we propose is similar to the RNE technique of Shahabi *et al.* [18] (and a recent improvement by Kriegel *et al.* [19]) who apply a Lipschitz embedding [20] to spatial networks. The RNE technique *embeds* the vertices of the spatial network in a high-dimensional vector space, such that vertices of the spatial network are now points in a high-dimensional vector space. A simpler distance measure (e.g., L_∞ metric) between these high-dimensional vector space points approximates the network distance between the corresponding vertices in the spatial network. The RNE technique uses $O(n\sqrt{n})$ storage, has a distortion of $O(\log n)$ and an approximate network distance query takes $O(\sqrt{n})$ time. On the other hand, our linear-size oracle can also be viewed as an embedding technique with the difference that the vertices are retained in their original embedding space (*i.e.*, two-dimensional for road networks), while having superior space and execution times and a distortion that lies between $(1-\varepsilon)$ and $(1+\varepsilon)$. This bounded distortion, instead of being a function of n, is what leads to the linear size of our oracle in contrast to RNE's $O(n\sqrt{n})$ storage requirements. Another difference between our proposed method and that of Shahabi *et al.* [18] and Kriegel *et al.* [19] is that our distance oracle *decouples* the spatial network from the objects that lie on it. Thus, once an approximate distance oracle of a spatial network has been computed, it can be reused for any dataset lying on the spatial network which is not the case for the other methods.

The concept of an approximate distance oracle has been proposed for a variety of graph networks. Thorup and Zwick [21] show that it is possible to construct an approximate oracle of size $O(kn^{1+\frac{1}{k}})$ for general graphs that can answer approximate distance queries in $O(1)$ time. The distortion of the approximate oracle of Thorup and Zwick lies between 1 and $(2k-1)$, where $k \geq 1$ is an integer. Gudmundsson *et al.* [16] construct an approximate oracles of size $O(n \log n)$ for *geometric t-spanner* graphs, such that the shortest path queries can be performed in $O(1)$ time with a distortion of $(1+\varepsilon)$. Gao and Zhang [15] propose an approximate oracle of size $O(n \log n)$ for unit-disk graphs that can retrieve approximate network distance in $O(1)$ time, with a distortion of $(1+\varepsilon)$. Our work goes beyond the work of Gudmundsson *et al.* [16] on geometric t-spanners and Gao and Zhang [15] on unit-disk graphs by dealing with spatial networks, while taking advantage of the spatial positions of the vertices to provide efficient search structures, such as B-trees, and hash tables, to the oracle.

The rest of this paper is organized as follows. Section II reviews the well-separated pairs technique. Sections III–V describe oracles of unit, $O(n)$, and $O(n \log n)$ sizes, respectively. Section VI contains the results of experiments. Section VII discusses strategies to integrate the distance oracle into a database system while concluding remarks are drawn in Section VIII.

II. Well-Separated Pairs

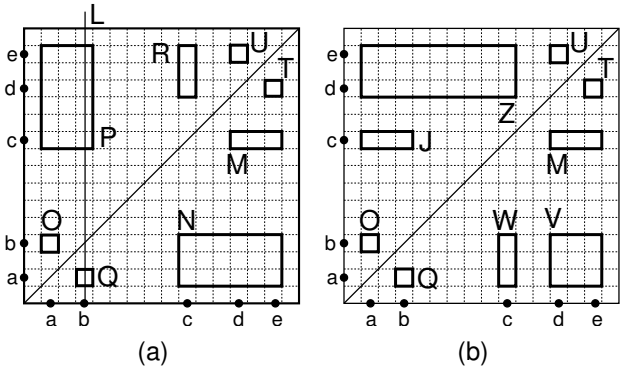

Figure 2: Example of a well-separated pair decomposition (WSPD) of a one-dimensional point set containing 5 points. The separation factors for the decompositions are (a) $s = 1$ and (b) $s = 0.25$.

Given a set of points A, the *diameter* of A is the maximum possible distance between any two points belonging to A. Similarly, given two sets of points A and B, the *minimum distance* between A and B is the minimum possible distance between a point in A and a point in B. Two sets of points A and B are said to be *well-separated* if the minimum distance between A and B is at least $s \cdot r$, where $s > 0$ is a *separation factor* and r is the larger diameter of the two sets. The pair (A,B) is termed a *well-separated pair* (WSP). A *well-separated pair decomposition* (WSPD) of a point set S, decomposes S into pairs of subsets (A,B), such that $\forall p,q \in S, p \neq q$, there exists exactly one WSP (A,B), such that $p \in A, q \in B$. The simplest WSPD of a point set S of n points contains $n \cdot (n-1)$ pairs of singleton element subsets (p,q) $\forall p,q \in S, p \neq q$. The key motivation for using WSPD is that for data of dimension d, and a separation factor s, we can always construct a WSPD containing $O(ns^d)$ pairs in $O(n \log n + ns^d)$ time [22]. Thus, the number of pairs is reduced to $O(n)$ as s is usually a fairly small constant independent of n.

As an example, consider the set of 5 one-dimensional points a, b, c, d, and e at positions 1.5. 3.5, 9.5, 12.5, and 14.5, respectively. There are a number of possible WSPDs for this dataset. Letting $s = 1$, one decomposition consists of $M = (\{\mathsf{d},\mathsf{e}\},\{\mathsf{c}\})$, $N = (\{\mathsf{c},\mathsf{d},\mathsf{e}\},\{\mathsf{a},\mathsf{b}\})$, $O = (\{\mathsf{a}\},\{\mathsf{b}\})$ $P = (\{\mathsf{a},\mathsf{b}\},\{\mathsf{c},\mathsf{d},\mathsf{e}\})$, $Q = (\{\mathsf{b}\},\{\mathsf{a}\})$, $R = (\{\mathsf{c}\},\{\mathsf{d},\mathsf{e}\})$, $T = (\{\mathsf{e}\},\{\mathsf{d}\})$, and $U = (\{\mathsf{d}\},\{\mathsf{e}\})$. This decomposition can be visualized by treating the individual pairs that make up the WSPD as rectangles in a two-dimensional space, where the axes correspond to the elements that make up the two sets involved in the WSP. For example, Figure 2a illustrates the

WSPD described above for $s = 1$, while Figure 2b illustrates another decomposition for the same points with $s = 0.25$. Notice that from the figure, we can see that any vertical (or horizontal) line L through one of the points, say p (e.g., b in Figure 2a), will cut the disjoint rectangles through which it passes so that the projection of their constituent points onto the y (or x) axis covers all of the points in S with the exception of p which means that, given any point $p', p' \neq p$, in the dataset, there is exactly one WSP A,B in the WSPD such that $p \in A$ and $p' \in B$. Also observe that just because the WSP (A,B) is a member of a WSPD does not necessarily mean that the symmetric pair (B,A) need be a member of the same WSPD. For example, consider the WSPD in Figure 2b where the symmetric pairs of $Z = (\{\mathsf{a},\mathsf{b},\mathsf{c}\},\{\mathsf{d},\mathsf{e}\})$, $M = (\{\mathsf{d},\mathsf{e}\},\{\mathsf{c}\})$, and $V = (\{\mathsf{d},\mathsf{e}\},\{\mathsf{a},\mathsf{b}\})$ are not present.

III. Oracles of Unit Size

We first assume that the ratio between the network and spatial distances is bounded both from above and below, and then show how these bounds can be attained. Note that the following is true for any finite graph.

Assumption 1: Given $s,t \in V$, $\gamma_L \leq \frac{d_G(s,t)}{d_S(s,t)} \leq \gamma_H$, where γ_L and $\gamma_H > 0$, albeit large.

Given a spatial network $G(V,E)$, Lemma 3.1 provides a simple method to determine the minimum γ_L distortion of G.

Lemma 3.1: The minimum γ_L distortion of a spatial network $G(V,E)$ containing n vertices, satisfying assumption 1, is given by: $\gamma_L = \min\{\frac{w(u,v)}{d_s(u,v)}\}$.

Proof: Let π be a shortest path of size p in G. Let π_i be the ith vertex in π, such that π_1 is the source vertex and π_{p+1} is the destination vertex. Suppose that the distortion $\gamma_L = \gamma = \frac{d_G(\pi_1,\pi_{p+1})}{d_S(\pi_1,\pi_{p+1})}$ of π is the minimum possible distortion among the $O(n^2)$ shortest paths in G. Now, let γ_i^* be the distortion of the i^{th} edge $\phi_i = (\pi_i,\pi_{i+i})$ in π comprising the shortest path π. We know that

$$d_G(\pi_1,\pi_{p+1}) = \gamma d_S(\pi_1,\pi_{p+1}) = \sum_{i=1}^{p} \gamma_i^* d_S(\pi_i,\pi_{i+1}) \quad (1)$$

$$d_S(\pi_1,\pi_{p+1}) = \sum_{i=1}^{p} \frac{\gamma_i^*}{\gamma} d_S(\pi_i,\pi_{i+1}). \quad (2)$$

Note that in Equation 2, from our initial assumption, $\frac{\gamma_i^*}{\gamma} \geq 1$. From the triangle inequality, we have that

$$d_S(\pi_1,\pi_{p+1}) \leq \sum_{i=1}^{p} d_S(\pi_i,\pi_{i+1}) \quad (3)$$

Combining Equations 2–3, we get

$$d_S(\pi_1,\pi_{p+1}) = \sum_{i=1}^{p} \frac{\gamma_i^*}{\gamma} d_S(\pi_i,\pi_{i+1}) \leq \sum_{i=1}^{p} d_S(\pi_i,\pi_{i+1}).$$

Hence, $\gamma = \gamma_1^* = \gamma_2^* = \gamma_3^* ... = \gamma_p^*$. This means that $\gamma_L = \min\{\frac{w(u,v)}{d_s(u,v)}\}$. ∎

Narasimhan and Smid [17] provide an algorithm that is based on WSPD of vertices for estimating an upper bound on the maximum distortion of a spatial network. We omit the description of the algorithm in this paper and provide a lemma below which captures their result.

Lemma 3.2: Given a spatial network G, we can compute an upper bound γ_H^* of the maximum distortion γ_H of G in $O(n \log n)$ time such that $\gamma_H^* \leq (1+\delta)\gamma_H$ and δ, $0 < \delta < 3$, is the desired approximation [17].

We define a simple approximate distance oracle that uses the values of γ_L and γ_H to provide an approximate network distance between two vertices, though with a large approximation error.

Theorem 3.3: Given a spatial network $G(V,E)$ with minimum γ_L and maximum γ_H distortions, $S_\varepsilon = \{\gamma_L,\gamma_H\}$ is an ε-approximate distance oracle of unit size such that the approximate network distance between a source vertex u and a destination vertex v is given by $S_\varepsilon(u,v) = \frac{\gamma_L+\gamma_H}{2} d_S(u,v)$, where $\varepsilon = \frac{\gamma_H-\gamma_L}{2}$, which can be computed in O(1) time.

The drawback of the above oracle is that the resulting error is dependent on the nature of the input spatial network G as the values of γ_L and γ_H may vary for different G. Moreover, γ_H can be quite large. which means that the resulting error of the distance oracle is also very large. Hence, S_ε is unsuitable for any meaningful query processing on spatial networks.

Distortion Spectrum: The behavior of the unit size oracle in Theorem 3.3 can be improved by noting that the minimum and the maximum distortion values of a spatial network depend on the spatial distance between a given source s and destination u. That is, usually for small spatial distances on G, the distortion γ values are large. However, distortion values quickly decrease as the spatial distance between the source and the destination increases. We can capture this relationship between the spatial distance and the distortion using a *distortion spectrum*, which provides the minimum and the maximum distortion values for different ranges of values of the spatial distances on G. The idea in capturing the distortion spectrum of a spatial network, instead of just computing the minimum γ_L and maximum γ_H distortion values of G, is that the resulting oracle while still taking $O(1)$ space and answering queries in $O(1)$ time, will provide better approximations, at least for large spatial distances on G.

Given a spatial network G, we first compute the spatial distance between the closest d_c and the farthest d_f pairs (diameter) of vertices in V. We then break up the distance interval $[d_c,d_f]$ into l arbitrary sub-intervals. Now for each of the l distance intervals, the minimum and maximum distortion values are computed using an algorithm similar to that of Narasimhan and Smid [17], except that we prune away WSPs that do not lie inside the specified distance interval. The distortion spectrum of G stores l spatial distance intervals and their corresponding minimum and maximum distortion values. Our experimental analysis does indeed confirm our earlier hypothesis that large distortions occur at small spatial distances. Moreover, the distortion values quickly reduce to smaller values as the sources and destinations get farther. An

approximate distance oracle S_ε defined using the distortion spectrum of G would provide better approximation, at least when s and u are far apart in G.

IV. $O(n)$ Distance Oracle

Given a source vertex s and a destination vertex w, an ε-approximate distance oracle S_ε provides an ε-approximate network distance $S_\varepsilon(s,w)$ such that:

$$(1-\varepsilon)\cdot S_\varepsilon(s,w) \leq d_G(s,w) \leq (1+\varepsilon)\cdot S_\varepsilon(s,w).$$

S_ε takes advantage of the path coherence [7], [8] in spatial networks which states that shortest paths from proximal sources to proximal destinations share common vertices. This can be seen in Figure 1 and was discussed in Section I.

We define a distance oracle S_ε of a spatial network as follows: $S_\varepsilon = \{(A,B,d_G^{AB}) | A,B \subset V, d_G^{AB} \in \mathbb{R}^+\}$. That is, we partition V into triples of the form (A,B,d_G^{AB}) such that A is a set of source vertices, B is a set of destination vertices, and d_G^{AB} is a value that approximates the network distances of all the shortest paths from A to B. The partitioning of the vertices into appropriate subsets of source and destination vertices is achieved by appealing to the well-separated pair decomposition [22], [23] and conditions under which it is satisfied for a spatial network are specified in this section. The ε-approximate network distance between a source s and a destination w is obtained by searching S_ε, which is indexed by a B-tree, for a triple (A,B,d_G^{AB}) such that A contains s and B contains w, in which case d_G^{AB} is the ε-approximate network distance of $d_G(s,w)$. In this section, we develop a distance oracle S_ε for spatial networks that is *linear* in the number of vertices in G, and that can produce an ε-approximate network distance in $O(\log n)$ time using a B-tree.

A. Preliminaries

Given a point set R in a d-dimensional space, we construct a WSPD on S by first constructing a PR quadtree [6], [24] on R. For simplicity, we assume that R is contained in a unit $[0,1]^d$ d-dimensional hypercube. This hypercube forms the root block of the PR quadtree T on R. The PR quadtree is obtained by recursively decomposing the block into 2^d congruent children blocks. The process continues until each block contains at most one point. Unfortunately, if two points in R are close to one another, it may lead to a long path of trivial blocks of which only one block would form an internal node. Callahan and Kosaraju's construction [14] did not incur this problem because they used a fair-split tree which is a data-dependent decomposition. Fischer and Har-Peled [25] remedy this problem through the use of a variant of a *path-compressed* quadtree which is obtained from the PR quadtree by compressing such trivial paths into one compressed link. The advantage of the path-compressed quadtree over the PR quadtree is that its use yields a tree with a total of $O(n)$ blocks.

Our discussion does not need to resort to the path-compressed quadtree while still using regular decomposition because of certain assumptions that we make about the distribution of the vertices in the embedding space. In particular, letting Δ be the ratio of the diameter of the set of vertices V to the distance between the closest pair of vertices in V and letting T be a PR quadtree on V, the height h of T is $O(\log\Delta)$. Consequently, given a vertex v in V, the Morton representation [26], [27] of $p(v)$, the spatial position of v, would be $O(\log\Delta)$ bits long. Assuming, without loss of generality, that the closest pair of vertices are one unit apart and that the embedding space is two-dimensional, we can pack as many as $\frac{\Delta^2}{2}$ vertices into our domain, where each vertex is in a cell of the appropriate width. Therefore, Δ is at least $O(\sqrt{n})$. In our analysis we assume that not all of the cells have vertices associated with them. In particular, we allow for Δ to be as large as $O(n)$ which is not unreasonable as demonstrated by our experiments with real road networks. From a practical standpoint, with respect to our experience with real road network data, we observe that the minimum geodesic distance between any two vertices on a road network is at least 1 meter. A PR quadtree on a sphere corresponding to the Earth with radius 6378 km and depth 24 has a 1 meter resolution at the equator. For such data, the size of the Morton code for a vertex on the road network using geographical coordinates is at most 48 bits in length.

The decomposition of R into WSPs is a *realization* on T, *i.e.*, subsets A_i, B_i of R forming a WSP (A_i,B_i) in $R\otimes R$ are pairs of blocks in R. The algorithm decomposes R into WSPs using T and s (*i.e.*, the separation factor) as inputs. The algorithm uses a list Q that is initialized to the pair (T,T) corresponding to the root of the quadtree on R. In each iteration of the algorithm, a pair (A,B) of blocks in T is retrieved from Q. If (A,B) is s-separated, it is reported as a WSP. Otherwise, new pairs are obtained by replacing A and B with their 2^d children blocks, which are inserted into Q. The algorithm terminates when Q is empty.

Suppose that a pair (u,v) is reported as a WSP by the algorithm. This would indicate that $(P(u),P(v))$ is not well-separated, where $P(b)$ is the parent block of b. Suppose further that the side length of $P(u)$ (or $P(v)$) is x and hence also its maximum possible diameter. The total number of blocks that are not well-separated from $P(u)$ is bounded by the number of blocks of diameter x that are contained within a hypersphere of diameter $(2s+1)x$ centered at $P(u)$, which contains a maximum of $O(s^d)$ blocks. Since T has $O(n)$ nodes, the algorithm creates a maximum of $O(s^d n)$ WSPs. This result and proof sketch is due to Callahan and Kosaraju [14] and we restate it below as Lemma 4.1, which is referenced in the subsequent discussion.

Lemma 4.1: Given a point set S containing n d-dimensional points, a fixed separation factor $s \geq 2$, the WSPD of S, $S\otimes S$ has $O(s^d n)$ WSPs [14].

B. Construction of the Oracle

We now describe an algorithm to construct the distance oracle S_ε of a spatial network $G(V,E)$. The algorithm takes

V as input and produces pairs of sets (u,v), such that $u,v \subset V, u \cap v = \emptyset$. The pair (u,v) has the property that the maximum and the minimum network distance between any source vertex s in u and any destination vertex w in v can be approximated by the exact network distance $d_G(p_u,p_v)$ between a source p_u vertex in u and a destination vertex p_v in v, with both p_u and p_v chosen at random. The source p_u and destination p_v vertices are termed the *representative* points of u and v, respectively. For simplicity, the discussion below assumes that G is undirected. Note, however, that the results below are equally applicable to directed spatial networks as well.

The construction of the oracle proceeds as follows. Algorithm BUILDORACLE takes the spatial network G, a PR quadtree T [6] on the spatial positions of the vertices in G, and the desired approximation ε as inputs. The output of the algorithm is a list L of Morton codes [26], such that a Morton code m in L uniquely corresponds to a pair of blocks (u,v) in T. Another equivalent interpretation of m is that it corresponds to a pair of subsets (k,l) of vertices, such that $k(l)$ is the set of vertices contained in the subtree of T with $u(v)$ as the root block. Henceforth in this paper, we assume that the three interpretations of m — block pair, Morton code, and pair of subsets of V — are all equivalent.

The algorithm uses a list Q of block pairs in T. At the start of the algorithm, Q is initialized with a block pair formed by the root block of T, as shown in line 1. The output list L is initially empty.

Each iteration of the algorithm retrieves the top block pair (u,v) in Q. If u and v are the same (as is the case with the first iteration of the algorithm, when the block pair (ROOTOF(T), ROOTOF(T)) is retrieved), then u and v are both split into their C children blocks, and the resulting C^2 block pairs are inserted into Q as shown in lines 4–7.

If u and v point to different blocks in T, then the algorithm examines if the block pair (u,v) is well-separated. We first choose two representative points $p_u \in u$, $p_v \in v$ at random (line 10). We then estimate the *network diameter* (or an over-approximation of the network diameter) of the blocks u and v, which is defined as the farthest vertex from p_u (or p_v) in u (or v) using a network distance measure (line 12). Estimating the diameter r can be done in a number of ways and is described in more details in Section IV-C. If the ratio of the network distance $d_G(p_u,p_v)$ to the diameter r is greater than or equal to $s = \frac{2}{\varepsilon}$, the block pair is well-separated and the block pair (u,v) is added to the output list L (lines 13–14). We later show in Section IV-E that if u,v are well-separated using a separation factor $s = \frac{2}{\varepsilon}$, then $d_G(p_u,p_v)$ is an ε-approximation of the network distance between any source vertex in u and destination vertex in v.

If the block pair (u,v) is not well-separated, then u (v) is split into its C children blocks if u (v) is not a leaf block, else it is not split. The resulting block pairs are inserted into L as shown in lines 16–26.

Algorithm 1
Procedure BUILDORACLE[G, T, ε]
Input: $G \leftarrow$ spatial network $G(V,E)$
Input: $T \leftarrow$ PR quadtree on the spatial positions of V
Input: $\varepsilon \leftarrow$ desired approximation; $\varepsilon > 0$
Output: $L \leftarrow$ set of Morton codes; initially empty
($* s \leftarrow \frac{2}{\varepsilon}$; separation factor $*$)
($* Q \leftarrow$ list of pairs of blocks in T; initially empty $*$)
($* b_u, b_v \leftarrow$ list of blocks; initially empty $*$)
1. INSERT(Q, ROOTOF(T), ROOTOF(T))
2. **while** (ISNOTEMPTY(Q)) **do**
3. $(u,v) \leftarrow$ TOP(Q)
4. **if** (u = v) **then**
5. ($*$ reject the pair if u (v) is a LEAF block $*$)
6. split u,v each into C children blocks
7. insert C^2 children block pairs of u,v into Q
8. **else**
9. ($*$ Choose representative points $*$)
10. $p_u \leftarrow R(u)$; $p_v \leftarrow R(v)$
11. ($*$ Estimate diameter of u and v $*$)
12. $r \leftarrow$ MAX(DIAMETER(u), DIAMETER(v))
13. **if** ($\frac{d_G(p_u,p_v)}{r} \geq s$) **then**
14. INSERT(L, MORTONCODE(u,v), $d_G(p_u,p_v)$)
15. **else**
16. **if** ISNOTLEAF(u) **then**
17. $b_u \leftarrow C$ children blocks of u
18. **else**
19. $b_u \leftarrow u$
20. **end-if**
21. **if** ISNOTLEAF(v) **then**
22. $b_v \leftarrow C$ children blocks of v
23. **else**
24. $b_v \leftarrow v$
25. **end-if**
26. INSERT all possible pairs in $b_u \times b_v$ in Q
27. **end-if**
28. **end-if**
29. **end-while**
30. **return** L

It is not difficult to see that Algorithm 1 is a WSPD of the set of vertices in G into a set of block pairs L. Hence, given a vertex pair s,w, the properties of a WSPD guarantees that there exists exactly one pair $(u,v,d_G(p_v,p_u))$ in L, such that u contains s and v contains w.

C. Estimating Network Diameter

Given a vertex p_u and a set of vertices $u \subset V$, the network diameter r of u is the farthest vertex from p_u in u using a network distance measure. One simple strategy to computing the network diameter of u is to obtain the network distance from p_u to every vertex in u and then to take the maximum value. However, this can be expensive to compute. Our strategy is to compute an over-approximation r of the network diameter of u that is easier to compute than the exact network diameter of u. However, this strategy has the unfortunate consequence

that as r is an over-approximation of the network diameter of u, Algorithm 1 would have to split the block pairs even further in order to make them well-separated. Consequently, there is a trade-off between the time spent on computing the network diameter of a block and the total storage space needed for the oracle. Below, we discuss several strategies to compute the network diameter of a set of vertices.

1) Given a block pair (u,v), we first obtain the network distance $d_G(p_u,p_v)$ between the representative points $p_u \in u$ and $p_v \in v$. We then apply an early terminating variant of Dijkstra's algorithm from p_u (p_v) that takes advantage of the *incremental* nature of Dijkstra's algorithm. That is, Dijkstra's algorithm with p_u (p_v) as a starting vertex visits vertices in G in an increasing order of their network distance from p_u (p_v). The algorithm terminates when it encounters a vertex that is farther than $\frac{d_G(p_u,p_v)}{s}$ from p_u (p_v). We now check to see if all the vertices in u (v) were visited by the Dijkstra algorithm. If yes, then u and v are well-separated.
2) If r' is the diameter of the geometric bounding box of u, the network diameter of u can be over-approximated by $\gamma_H r'$, which can be computed using Lemma 3.3, or using the distortion spectrum of the spatial network.
3) The approach of Goldberg and Harrelson [28] first selects a set of vertices, termed *landmarks*, at random. The network distance from each of the landmark vertices to all the vertices in G is precomputed. Once precomputed, the diameter of $u(v)$ can be upper bounded using the triangle inequality and the network distance to the nearest landmark.

D. Querying the Oracle

The output of Algorithm 1 is a list L of Morton codes. For each of the Morton codes in L, we associate the exact network distance between the representative points. Moreover, given a source and a destination vertex, an access structure (e.g., a B-tree or a hash table) aids efficient searching on L for a block pair containing the source and the destination vertices.

Given a source vertex s and a destination vertex w, the oracle obtains the ε-approximate network distance by first computing the Morton code corresponding to the spatial position of s and w. Using the access structure on L, we are able to obtain the ε-approximate network distance of s and w, which is the network distance between the representative points of the block pair (u,v) in L, such that u contains s and v contains w.

E. Analysis

This section provides bounds on the size of the distance oracle of G by appealing to the equivalence between the decomposition of a spatial network in Algorithm 1 and the WSPD of a point set. We now show how to extend the notion of a WSPD in terms of a spatial distance to one in terms of a network distance. This is captured by Lemma 4.2 below.

Lemma 4.2: Given an s-WSPD of the vertices V of a spatial network $G(V,E)$ based on a spatial distance also yields a s'-WSPD of V using a network distance with $s' = s \cdot \frac{\gamma_L}{\gamma_H}$.

Proof: Given an s-WSP, (A,B) in the decomposition of $V \otimes V$ using the spatial distance measure, the minimum spatial distance between A and B is at least $s \cdot r$, where r is the larger of the diameters of A and B.

Consider two vertices u,v in A (or B). We have $d_G(u,v) \leq \gamma_H \cdot d_S(u,v) \leq \gamma_H \cdot r$ as $d_S(u,v) \leq r$ by virtue of r being the diameter of A or B. r', the maximum value of $d_G(u,v)$, is the diameter of A (and B) using a network distance measure and we have that $r' \leq \gamma_H \cdot r$. Therefore, the spatial distance diameter of A (or B) is scaled by at most a factor of γ_H to obtain the network distance diameter r'.

Considering a vertex pair (a,b), such that $a \in A, b \in B$, we have from the WSP condition and Assumption 1 that:

$$s \cdot r \leq d_S(a,b) \leq \frac{d_G(a,b)}{\gamma_L} \tag{4}$$

Replacing r with $\frac{r'}{\gamma_H}$ in (4), we obtain $s \cdot \frac{r'}{\gamma_H} \leq d_S(a,b) \leq \frac{d_G(a,b)}{\gamma_L}$. The above relationship between the minimum and maximum bounds on $d_S(a,b)$ can be rewritten as $r' \cdot s \cdot \frac{\gamma_L}{\gamma_H} \leq d_G(a,b)$. Now, letting $s' = s \cdot \frac{\gamma_L}{\gamma_H}$, leads to the desired result $s' \cdot r' \leq d_G(a,b)$, which is equivalent to saying that A and B are well-separated using the network distance measure with a separation factor of s'. ■

We now show that a WSPD of the vertices of a spatial network is a realization of an approximate distance oracle.

Lemma 4.3: Let (A,B) be a WSP in the s-WSPD of G using a network distance measure, such that u^*, v^* are the representative points of A and B, respectively. The network distance $d_G(u^*,v^*)$ between the representative points is an $\varepsilon = \frac{2}{s}$ approximation of the network distance $d_G(u,v)$ between any pair of vertices (u,v), such that $u \in A$ and $v \in B$.

Proof: Given a pair of vertices (u,v), such that $u \in A, v \in B$, we know from the triangle inequality that

$$d_G(u^*,v^*) - d_G(u,u^*) - d_G(v^*,v) \leq d_G(u,v)$$
$$d_G(u,u^*) + d_G(u^*,v^*) + d_G(v^*,v) \geq d_G(u,v)$$

Without loss of generality, we assume that $d_G(v^*,v) \geq d_G(u,u^*)$. Substituting above, we get

$$d_G(u^*,v^*) - 2d_G(v^*,v) \leq d_G(u,v)$$
$$d_G(u^*,v^*) + 2d_G(v^*,v) \geq d_G(u,v)$$

$$d_G(u^*,v^*)\left(1 - \frac{2d_G(v^*,v)}{d_G(u^*,v^*)}\right) \leq d_G(u,v)$$
$$d_G(u^*,v^*)\left(1 + \frac{2d_G(v^*,v)}{d_G(u^*,v^*)}\right) \geq d_G(u,v)$$

In line 13 of Algorithm 1, we ensure that the condition $\frac{d_G(u^*,v^*)}{d_G(v^*,v)} \geq s$ is satisfied for all vertices in B. Substituting it above,

$$\left(1 - \frac{2}{s}\right) d_G(u^*,v^*) \leq d_G(u,v) \leq \left(1 + \frac{2}{s}\right) d_G(u^*,v^*)$$

Substituting, $\varepsilon = \frac{2}{s}$, we get

$$(1-\varepsilon)d_G(u^*,v^*) \leq d_G(u,v) \leq (1+\varepsilon)d_G(u^*,v^*)$$

■

Now, having established that $\varepsilon = \frac{2}{s}$, we now obtain a bound on the size of the distance oracle.

Lemma 4.4: For a given value of $\varepsilon = \frac{2}{s}$, the size of the oracle produced by Algorithm 1 is no worse than $O((\frac{\gamma_H}{\varepsilon\gamma_L})^d n)$.

Proof: Let (A,B) be a WSP, such that u^*, v^* are the representative points of A and B, respectively. We assume that A (B) is contained in a bounding hypersphere of diameter r. The network diameter of A and B is bounded by

$$\gamma_L r \leq \text{DIAMETER}(A) \leq \gamma_H r$$
$$\gamma_L r \leq \text{DIAMETER}(B) \leq \gamma_H r$$

As (A,B) is a WSP, $d_G(u^*,v^*)$ can be similarly bounded by

$$d_G(u^*,v^*) \leq \gamma_H rs.$$

The effective separation factor s' of the WSPD is $\frac{s\gamma_H}{\gamma_L}$. Hence, the worse case storage requirement of the oracle is $O((\frac{\gamma_H}{\varepsilon\gamma_L})^d n)$. ■

This leads us to the final result of this section:

Theorem 4.5: Given a spatial network $G(V,E)$, we can construct an oracle of size $O(\frac{n}{\varepsilon^d})$ to retrieve the ε-approximate network distance between any vertex pair in $O(\log n)$ time.

The real utility of the above theorem is to establish the linear size of our distance oracle. Note that the constants of proportionality estimated using an empirical analysis were found to lie, in most cases, between 1.5 and 3 which is much smaller than the worse case bound of $(\frac{\gamma_H}{\gamma_L})^d$ established in Lemma 4.4.

V. $O(n \log n)$ DISTANCE ORACLE

We now describe an oracle of size $O(\frac{n\log n}{\varepsilon^d})$ that can produce ε-approximate network distances in $O(1)$ time using a hash table. We first introduce an alternative WSPD of a point set R into pairs of the form (p,B), where p is a point in R and B is a subset of R. Such a pair is termed a *one-to-many WSP* (OM-WSP) and the resulting decomposition of R into OM-WSPs is termed an *one-to-many WSPD* (OM-WSPD) of R.

Lemma 5.1: Given a point set R containing n points and a separation factor $s > 2$, a WSPD of R can be decomposed into $O(s^d nh)$ OM-WSPs of the form $(\{a\},B)$, where $a \in R$, $B \subset R$, and h is the height of the PR quadtree on R.

Proof: Let T be a PR quadtree of height h on the spatial positions of R. Suppose that a block i in T containing s_i points is paired up with a_i other blocks in T during the WSPD of R. As a result, $s_i a_i$ OM-WSPs are created. The total number of OM-WSPs generated by $O(n)$ nodes in T is given by $\sum_{i=1}^{O(n)} a_i s_i$, which is less than $s^d \sum_{i=1}^{O(n)} s_i$, as $O(s^d)$ upper-bounds a_i. In a PR quadtree of height h, we know that $\sum_{i=1}^{O(n)} s_i = O(nh)$. Substituting the above result in $s^d \sum_{i=1}^{O(n)} s_i$, we obtain that the total number of OM-WSPs created by the algorithm is $O(s^d nh)$. ■

Recall from Section IV-A that the height h of a PR quadtree T on the position of vertices on a spatial network is given by $h = O(\log\Delta)$, where Δ is the the ratio of the diameter of the set of vertices V to the distance between the closest pair of vertices in V. Furthermore, in Section IV-A, we assumed that the value of Δ cannot be more than $O(n)$ which means that $h = O(\log n)$. Substituting, $h = O(\log n)$ and $s = \frac{2}{\varepsilon}$, we get that the WSPD of a spatial network containing $O(\frac{n}{\varepsilon^d})$ WSPs can be further decomposed in to $O(\frac{n\log n}{\varepsilon^d})$ OM-WSPs.

We now show that given a source vertex s and a destination vertex w, the OM-WSP containing the pair can be found in $O(1)$ using the properties of a WSPD. Given a OM-WSPD of a spatial network, for each vertex $u \in V$, let P_u be set of pairs of the form $\{u,B\}$ in the decomposition. Furthermore, we construct *distance classes* D_j using the OM-WSPs in P_u, such that D_j contains all the pairs (u,B) in P_u satisfying the condition $(1-\rho)j \leq d_S(u,R(B)) \leq (1+\rho)j$, where $R(B)$ is the representative point of B and $\rho > 0$. The following lemma shows that given a source s and a destination w, we can find the OM-WSP (s,B) containing s and $w \in B$ in $O(1)$, as the distance class D_j containing (s,B) only contains c other OM-WSPs, where c only depends on ρ, ε, γ_L, γ_H, and d and not on the number of vertices n on a spatial network. We omit the proof of the lemma for the sake of conciseness but the interested reader is referred to [22].

Lemma 5.2: Given a vertex pair $s,w \in V$, the number of OM-WSPs of the form $\{s,B\}$ in the canonical realization, such that $d_S(s,B) \leq d_S(s,w)$ and $d_S(s,R(B)) \in D_j$ is a constant depending only on ρ, ε, γ_L, γ_H, and d.

This leads us to the main result of this section:

Theorem 5.3: Given a spatial network $G(V,E)$, we can construct an oracle of size $O(\frac{n\log n}{\varepsilon^d})$ to retrieve the ε-approximate network distance between any vertex pair in $O(1)$ time.

Proof: We define a hash $\mathbb{H}$ data structure as follows. The WSPs in the $O(s^d n)$ distance oracle in Theorem 4.5 are first decomposed into $O(s^d n \log n)$ OM-WSPs. Next, for every vertex $u \in V$, we aggregate OM-WSPs of the form $\{u,B_i\}$ and add B_i to a set B_u. For a suitably defined value of ρ, we partition every element B_i in B_u based on the minimum a and maximum b spatial distance between u and B_i into different distance classes D_j. Note that the spatial distance interval $[a,b]$ of B_i can span more than one (but bounded by a small constant) distance classes in D_j. Lemma 5.2 provides that after all the elements in B_u have been assigned to different distance classes, the number of elements contained in each distance class is independent of n. Finally, all the vertices in V along with their distance classes are stored on disk and the resulting representation is the hash data structure $\mathbb{H}$. Now, given a source u and destination v, we first compute the spatial distance $d_S(u,v)$ which determines which distance class D'_j of u contains the OM-WSP containing $\{u,v\}$. We then examine all the elements in D'_j until we find an OM-WSP $\{u,B\}$ such

that B contains v. The approximate network distance value associated with $\{u,B\}$ approximates $d_G(u,v)$. Note that the total work performed in retrieving $\{u,B\}$ only depends on ρ, ε, γ_L, γ_H and d, but not on n. ■

VI. Experimental Results

In this section we perform an experimental evaluation of the oracles of size $O(1)$ and $O(n)$ described in this paper. We did not evaluate the $O(n\log n)$-size and $O(1)$ execution time oracle as it was presented primarily as a theoretical exercise to show the interplay between optimal execution time and space requirements, although, of course, there is really no justification for its use. The experiments were carried out on a Linux (2.4.2 kernel) quad 2.4 GHz Xeon server with one gigabyte of RAM. We implemented our algorithms using GNU C++. A number of publicly available road network datasets were used in the evaluation. These were obtained from the US Tiger Census [29] and the National Atlas [30] websites. In particular, we used a dataset containing all the major roads in the USA (*i.e.*, more than 380,000 vertices and 400,000 edges). Sample random rectangular regions were drawn from the dataset and the road network segments contained completely within them were extracted to serve as inputs to the evaluation. By taking the samples at random we were able to account for variations of road networks such as rural versus urban, and spatial network configurations that would lead to different sizes of the oracle.

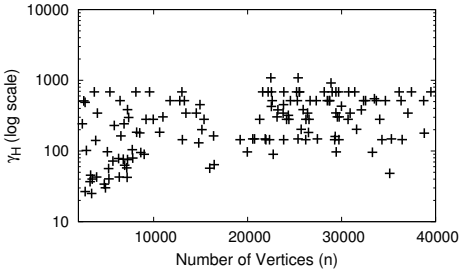

Figure 3: Maximum distortion γ_H of different road networks.

The maximum distortion γ_H of spatial networks obtained by applying the algorithm of Narasimhan and Smid [17] which is captured by Lemma 3.2. on spatial networks of varying sizes is shown in Figure 3. The value of γ_L for all the input spatial networks was one. Note that this need not always be the case. For example, if the edge weights are in terms of the time taken to travel the edge, and spatial distance is in miles, then the value of γ_L would correspond to distortion of the edge in the spatial network with the lowest speed limit. From Figure 3, we see that the value of γ_H for road networks can be very large and ranges between 10 and 1000. An $O(1)$-size oracle, described in Theorem 3.3, that uses γ_L and γ_H to provide approximate network distances cannot provide a reasonable answer for query processing as the resulting error $\varepsilon = \frac{\gamma_H - \gamma_L}{2} \approx 500$ is very large.

Next, we computed the distortion spectrum of a large road network dataset corresponding to the important roads in the

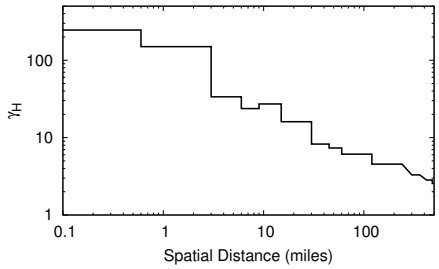

Figure 4: Distortion spectrum of the eastern seaboard road network dataset containing 91,113 vertices.

eastern seaboard states of USA, consisting of 91,113 vertices and 114,176 edges, shown in Figure 4. As we can see, the maximum distortion for small spatial distances (less than 2 miles) can be very large. However, as the spatial distance between the source and destination increases (and is greater than 50 miles), the maximum distortion quickly reduces to a low value. Note that an approximate distance oracle of size $O(1)$ that uses the distortion spectrum of a spatial network may be suitable when the spatial distance between a given source vertex and destination vertex is large.

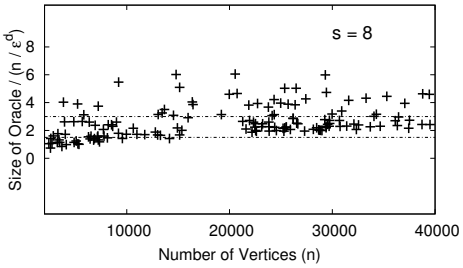

Figure 5: Size of the oracle in terms of the number of Morton codes, normalized by n/ε^d.

We now examine the characteristics of the $O(n)$-size oracle that can provide deterministic guarantees on the quality of the approximate answers it provides. We built the oracles by applying Algorithm 1 to the same road networks of different sizes used to obtain Figure 3. Figure 5 shows the effect of the size of the road networks, in terms of number of vertices n, on the size of the resultant distance oracle, which is measured in terms of the number of Morton codes normalized by n/ε^d. We chose $s = 8$ and $d = 2$ for this set of evaluations. It is easy to see that the size of the oracle does indeed follow $c \cdot n/\varepsilon^d$, where c is estimated empirically to lie between 1 and 6 and in most cases lies between 1.5 and 3 for the road networks used in our experiments. This study shows the applicability of our technique to large road networks as the size of the oracle is linear in n and that the constants involved are small, typically between 1.5 and 3. The large value of γ_H shown in Figure 3 seems to have little effect on the size of the oracle.

Next, we built $O(n)$-size oracles on the Washington, DC dataset containing 12,304 vertices for varying values of ε between 0.50 to 0.0078, *i.e.*, s = 4 to 256 which is shown in Figure 6a. In Figure 6b we recorded the size of the oracles

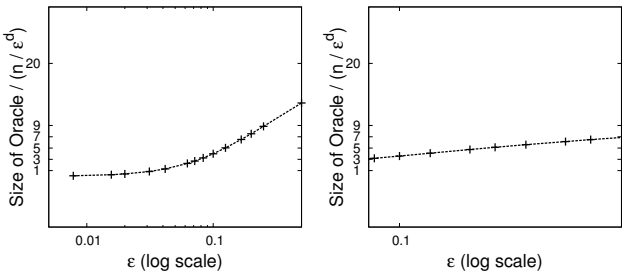

Figure 6: The size of the oracle in terms of number of Morton codes, normalized by n/ε^d applied to a) Washington, DC b) eastern seaboard datasets.

for the eastern seaboard dataset containing 91,113 vertices for values of ε between 0.50 to 0.0625, *i.e.*, s = 4 to 32. We recorded the size of the resulting distance oracle in terms of the number of Morton codes, normalized by n/ε^d. We can see that the constants of proportionality are small values that vary between 1 and 10.

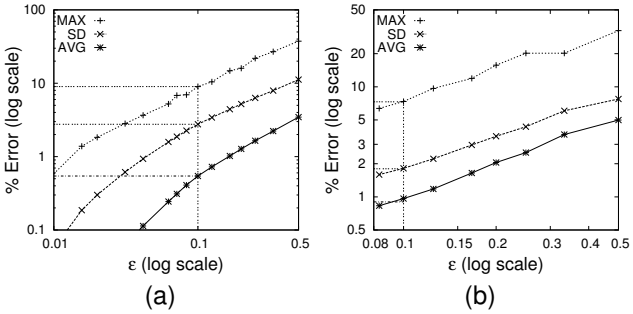

Figure 7: The maximum, average, and the standard deviation errors for 100,000 network distance queries on the various oracles in Figure 6 on a) Washington, DC, and b) eastern seaboard datasets.

For each of the distance oracles computed in Figure 6a–b, we made 100,000 ε-approximate distance queries between a vertex pair chosen at random. We computed the actual network distance between the pairs, and recorded the *maximum*, *average*, and the *standard deviation* of the error due to the approximation. The resultant error for the oracles is shown in Figure 7. We can see that while the maximum error is within the prescribed bounds, the average and the standard deviation of the error are much lower than the maximum theoretical bounds. For example, for the distance oracles on the Washington, DC dataset shown in Figure 7a, the average error, standard deviation and the maximum error (in percentage) of the answers provided by $\varepsilon = 0.1$ (10% error) oracle is 0.5%, 2.7%, and 9.0%, respectively. In the case of the eastern seaboard dataset, shown in Figure 7b for $\varepsilon = 0.1$ the corresponding average error, standard deviation and the maximum error (in percentage) values are 0.9%, 1.8%, and 7.3%, respectively. These low average errors (i.e., less than 1 %) mean that, in practice, the quality of the answers provided by this oracle is very close to the exact network distance.

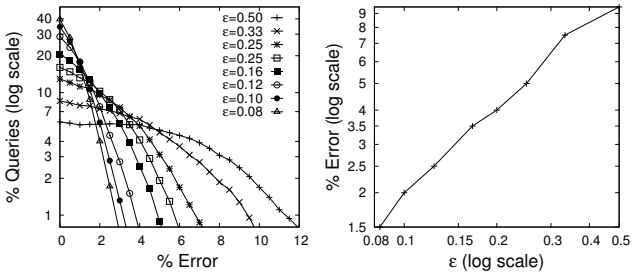

Figure 8: a) Percentages of queries along with their associated errors and b) maximum error of 90% of the queries, for the oracles in Figure 6b.

Figure 8a tabulates the resulting errors in the answers provided by the distance oracles as a percentage of the total number of queries for the 100,000 queries on the eastern seaboard dataset of Figure 6b. For example, Figure 8a shows that for $\varepsilon = 0.25$ (*i.e.*, 25% approximation), 12.9% of the queries are provided with more or less exact answers (i.e., less than 0.5% error). Moreover, 90% of the queries have errors of less than 5% as can be seen from Figure 8b, which tabulates the maximum error of 90% of the queries. We note from our data (not shown here) that while the maximum possible error of the oracle is 25%, less than 1% of the answers to queries have errors of more than 10%.

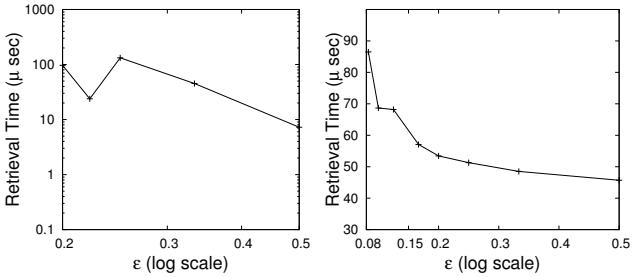

Figure 9: Average time to retrieve an ε-approximate network distance for values of ε between a) 0.1 and 0.5 for Washington, DC, and b) 0.08 and 0.5 for eastern seaboard dataset.

Next, we computed the average time taken to retrieve an ε-approximate network distance for some of the oracles in Figure 6a. Figure 9a–b shows that the average time taken to compute an ε-approximate network distance is on the order of tens of microseconds with maximums of 100 microseconds for the Washington, DC dataset in Figure 9a, and 86 microseconds for the eastern seaboard dataset in Figure 9b. Note that this time can be still improved further by using a more efficient implementation of the B-tree structure.

VII. Database Integration

A major data engineering problem in databases is that spatial networks cannot be easily represented using a relational model and furthermore, operations on it cannot be easily cast

in terms of relational operators namely *selection*, *projection*, and *joins*, etc. In this paper, we introduced an approximate distance oracle which can be represented as a *relation* in a database system indexed by a B-tree. We now show how operations on spatial networks can be cast as relational operations on the distance oracles. In particular, we demonstrate how to cast region search, k nearest neighbor queries, and spatial join queries in terms of relational operations on the oracle relation using the SQL language. Such a setup is very efficient as, now, everything can be done in the context of a database system. Our work opens up the use of a commercial database for building interactive applications (e.g., GIS applications [31]) on spatial networks.

In the rest of the Section, we restrict our discussion to a spatial network that is embedded in a two-dimensional space (e.g., road networks) where the position of an object is represented by its position in terms of, say, latitude and longitude. We first define a new object type called *Morton* which consists of a *code* and a *level*. Given a two-dimensional point object u, the *code* is the bit-interleaved representation [6] of the x and y-dimensional components of u of length $2 \cdot level$ bits. For all practical purposes, we will restrict the length of the Morton code to 58 bits (29 each for x and y-dimensions) and assign 6 bits to encode the *level* of the Morton code which gives a resolution of roughly 7.46 cms at the equator of earth. The level and the code of the Morton blocks are packed into a single 64 bit-integer thereby taking advantage of the bit-level parallelism in 64 bit machines. Let $Z : \mathbb{R}^2 \rightarrow \mathbb{N}$ be the transformation function from a two-dimensional object to a Morton object which is packed into a 64-bit integer. Such a transformation can be performed in $O(1)$ time entirely using bit operations. We also define a comparator function for the Morton objects which given two Morton objects, orders them based on their relative positions in a Morton space filling curve in $O(1)$ time. In particular, two Morton objects u, v are said to be *equal* if u contains v, or v contains u. In addition, we define another transformation function $Z_4 : (\mathbb{R}^2, \mathbb{R}^2) \rightarrow \mathbb{N}$ that transforms a pair of two-dimensional points into a 128 bit Morton block. In this case, 30 bits are assigned to each of the four dimensions with 8 bits assigned to storing the levels.

An ε-approximate distance oracle O is given by the relation $(Z_4^{AB}, d_\varepsilon)$ where $O.Z_4^{AB}$ is a four-dimensional Morton object such that A is a set of sources, B is a set of destinations, and d_ε is the ε-approximate network distance that approximates the network distance between any location in A to any location in B. The attribute $O.Z_4^{AB}$ is indexed using a B-tree. Note that we only store the four-dimensional Morton object corresponding to A, B in the relation O and there is no need to store the graph representation, or for that matter even the positions of the vertices and edges of a spatial network. This leads us to an SQL query to obtain the ε-approximate network distance by applying a "selection" operator on the oracle relation.

Network Distance Query: Given source p and destination q, find the ε-approximate network distance between them.

SELECT $O.d_\varepsilon$ FROM O WHERE $O.Z_4^{AB} = Z_4(p,q)$

An alternate representation of the oracle O is a relation of schema $(Z^A, Z^B, d_\varepsilon)$ which is similar to the one described above except that instead of storing A and B as a single four-dimensional Morton object, we store them as two separate two-dimensional Morton blocks, namely Z^A and Z^B. We now construct a two-dimensional B-tree on (Z^A, Z^B) which is a B-tree is on Z^A whose leaves are B-trees on Z^B for the tuples that have the same value of Z^A.

We now discuss how to perform spatial queries on a spatial network using the ε-approximate network distance oracle. Assume the following setup. O is an ε-approximate distance oracle of a predefined approximation. Let R be a relation of restaurants of schema (pos, type), where pos is the position of the restaurants given by a two-dimensional point object, and *type* is the type of the cuisine served by the restaurant. Furthermore, we assume that there is a B-tree on $Z(R.pos)$. We also define another relation Q of coffee shops given by the same schema as R such that $Z(Q.pos)$ is also indexed using a B-tree. We present the following queries on a spatial network.

Region Search: Given a query location q, find all restaurants in R that are within 10 miles of q.

SELECT $R.pos$, $O.d_\varepsilon$ FROM O, R WHERE
$O.Z^A = Z(q)$ and $O.Z^B = Z(R.pos)$ and $O.d_\varepsilon \leq 10$ miles

k-Nearest Neighbor Search: Given a query location q, find the k closest restaurants in R to q that serve *Italian* cuisine.

SELECT $R.pos$, $O.d_\varepsilon$ FROM O, R WHERE
$O.Z^A = Z(q)$ and $O.Z^B = Z(R.pos)$ and R.type = "Italian"
ORDER BY $O.d_\varepsilon$ LIMIT k

Distance Join Operator: Given that R, S are relations of restaurants and coffee shops, respectively. Find the k closest pairs of restaurants and coffee shops such that closest is defined in terms of the network distance [32].

SELECT $R.pos$, $Q.pos$, $O.d_\varepsilon$ FROM O, R, Q
WHERE $O.Z^A = Z(R.pos)$ and $O.Z^B = Z(Q.pos)$
ORDER BY $O.d_\varepsilon$ LIMIT k

Each of the above operations are simple relational operations on the oracle relation that uses a B-tree. A commercial database can optimize complicated query processing scenarios involving B-trees. For example, the ε-approximate network distance query and the region search are simple selection operators that uses the B-tree index. The rest of the queries involve simple join operations aided by a B-tree which can be efficiently performed by a query optimizer. In short, query processing on spatial networks can be easily integrated into a traditional database system. Finally, our strategy relies on the precomputation of a distance oracle for a pre-specified value of ε. For example, a distance oracle for the road network of USA can be precomputed and commercially distributed. This will then enable query processing on any spatial dataset residing on the road network of USA using a commercial database. Moreover, the wide use of such an oracle will justify the large cost of the precomputation.

VIII. Concluding Remarks

In this paper, we presented three approximate oracles for spatial networks that can answer approximate network distance

queries. Our first oracle took unit space and could answer approximate network distance queries in $O(1)$ time. The drawback of this oracle was that the resulting error was large and dependent on the characteristics of the given spatial network. This led us to propose an oracle of size $O(n)$ that took advantage of the path coherence in spatial networks by decomposing the spatial network into sets of coherent source vertices and coherent destination vertices such that the network distances between them are represented by a single value that approximated them. Such an oracle could answer queries in $O(\log n)$ time using a B-tree. We also presented a theoretical analysis of a third variant that took $O(n \log n)$ space, but which could retrieve approximate network distances in $O(1)$ time with the aid of a hash table. Experiments were performed on the $O(n)$-size oracle that confirmed its linear storage requirements, while enabling us to answer approximate network distance queries on the order of tens of microseconds. Our experiments also demonstrated that the average and the standard deviation of the approximation error were low and, in fact, on the average, the error was much lower than the theoretical maximum ε value. For example, in the case of an oracle with $\varepsilon = 0.1$ (10% approximation) on the eastern seaboard dataset, our average error was just around 0.9% with 90% of the queries making less than 2% errors which is negligible. This means that our oracle can be used as an efficient construct to provide *near* exact network distances in real time. Finally, we showed how to integrate our distance oracle with a relational database system and demonstrated its use in complicated query processing scenarios involving datasets residing on a spatial network.

Acknowledgments: We have benefited greatly from discussions with Houman Alborzi.

REFERENCES

[1] D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao, "Query processing in spatial network databases," in *Proc. of VLDB*, Berlin, Germany, Sep. 2003, pp. 802–813.

[2] H.-J. Cho and C.-W. Chung, "An efficient and scalable approach to CNN queries in a road network," in *Proc. of VLDB*, Trondheim, Norway, Sep. 2005, pp. 865–876.

[3] N. Jing, Y.-W. Huang, and E. A. Rundensteiner, "Hierarchical encoded path views for path query processing: an optimal model and its performance evaluation," *TKDE*, vol. 10, no. 3, pp. 409–432, May 1998.

[4] S. Jung and S. Pramanik, "An efficient path computation model for hierarchically structured topographical road maps," *TKDE* vol. 14, no. 5, pp. 1029–1046, Sep./Oct. 2002.

[5] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[6] H. Samet, *Foundations of Multidimensional and Metric Data Structures*. San Francisco: Morgan-Kaufmann, 2006.

[7] H. Samet, J. Sankaranarayanan, and H. Alborzi, "Scalable network distance browsing in spatial databases," in *Proc. of SIGMOD*, Vancouver, Canada, June 2008, pp. 43–54 (SIGMOD 2008 Best Paper Award).

[8] J. Sankaranarayanan, H. Alborzi, and H. Samet, "Efficient query processing on spatial networks," in *Proc. of the 13th ACM GIS*, Bremen, Germany, Nov. 2005, pp. 200–209.

[9] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics: Principles and Practice*, 2nd ed. Reading, MA: Addison-Wesley, 1990.

[10] D. Wagner and T. Willhalm, "Geometric speed-up techniques for finding shortest paths in large sparse graphs," in *Proc. of ESA*, LNCS 2832, Budapest, Hungary, Sep. 2003, pp. 776–787.

[11] G. M. Hunter and K. Steiglitz, "Operations on images using quad trees," *PAMI*, vol. 1, no. 2, pp. 145–153, Apr. 1979.

[12] H. Samet, "Neighbor finding in images represented by octrees," *CVGIP*, vol. 46, no. 3, pp. 367–386, June 1989.

[13] H. Samet, "Implementing ray tracing with octrees and neighbor finding," *Computers & Graphics*, vol. 13, no. 4, pp. 445–460, 1989.

[14] P. B. Callahan and S. R. Kosaraju, "Faster algorithms for some geometric graph problems in higher dimensions," in *Proc. of SODA*, Austin, TX, Jan. 1993, pp. 291–300.

[15] J. Gao and L. Zhang, "Well-separated pair decomposition for the unit-disk graph metric and its applications," in *Proc. of STOC*, San Diego, CA, July 2003, pp. 483–492.

[16] J. Gudmundsson, C. Levcopoulos, G. Narasimhan, and M. Smid, "Approximate distance oracles for geometric graphs," in *Proc. of SODA*, San Francisco, CA, Jan. 2002, pp. 828–837.

[17] G. Narasimhan and M. Smid, "Approximating the stretch factor of Euclidean graphs," *SICOMP*, vol. 30, no. 3, pp. 978–989, 2000.

[18] C. Shahabi, M. R. Kolahdouzan, and M. Sharifzadeh, "A road network embedding technique for k-nearest neighbor search in moving object databases," *GeoInformatica*, vol. 7, no. 3, pp. 255–273, Sep. 2003.

[19] H.-P. Kriegel, P. Kröger, M. Renz, and T. Schmidt, "Hierarchical graph embedding for efficient query processing in very large traffic networks," in *Proc. of SSDBM*, LNCS 5069, Hong Kong, China, July 2008, pp. 150–167.

[20] N. Linial, E. London, and Y. Rabinovich, "The geometry of graphs and some of its algorithmic applications," *Combinatorica*, vol. 15, pp. 215–245, 1995.

[21] M. Thorup and U. Zwick, "Approximate distance oracles," in *Proc. of STOC*, Hersonissos, Greece, 2001, pp. 183–192.

[22] P. Callahan, "Dealing with higher dimensions: The well-separated pair decomposition and its applications," Ph.D. dissertation, The Johns Hopkins University, Baltimore, MD, Sep. 1995.

[23] P. B. Callahan and S. R. Kosaraju, "A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields," *JACM*, vol. 42, no. 1, pp. 67–90, Jan. 1995.

[24] J. A. Orenstein, "Multidimensional tries used for associative searching," *Inf. Proc. Let.*, vol. 14, no. 4, pp. 150–157, June 1982.

[25] J. Fischer and S. Har-Peled, "Dynamic well-separated pair decomposition made easy," in *Proc. of CCCG*, Windsor, Canada, Aug. 2005, pp. 235–238.

[26] G. M. Morton, "A computer oriented geodetic data base and a new technique in file sequencing," IBM Ltd., Ottawa, Canada, Tech. Rep., 1966.

[27] I. Gargantini, "An effective way to represent quadtrees," *CACM*, vol. 25, no. 12, pp. 905–910, Dec. 1982.

[28] A. V. Goldberg and R. F. Werneck, "Computing point-to-point shortest paths from external memory," in *Proc. of ALENEX*, Vancouver, Canada, Jan. 2005.

[29] U.S. Census Bureau, "TIGER/Line Files, Census 2000," U.S. Census Bureau, Washington, DC, Oct. 2001, http://www.census.gov/geo/www/tiger/tiger2k/tiger2000.html.

[30] U.S. Geological Survey, "Major Roads of the United States," U.S. Geological Survey, Reston, VA, Nov. 1999, http://nationalatlas.gov/atlasftp.html.

[31] H. Samet, H. Alborzi, F. Brabec, C. Esperança, G. R. Hjaltason, F. Morgan, and E. Tanin, "Use of the SAND spatial browser for digital government applications," *CACM*, vol. 46, no. 1, pp. 63–66, Jan. 2003.

[32] J. Sankaranarayanan, H. Alborzi, and H. Samet, "Distance join queries on spatial networks," in *Proc. of ACM GIS*, Arlington, VA, Nov. 2006, pp. 211–218.