

Approximate Distance Oracles for Graphs with Dense Clusters

Mattias Andersson^a Joachim Gudmundsson^b
Christos Levcopoulos^a

^a*Department of Computer Science, Lund University, Box 118, 221 00 Lund,
Sweden. Email: mattias@cs.lth.se, christos@cs.lth.se*

^b*National ICT Australia Ltd, Sydney, Australia.
Email: joachim.gudmundsson@nicta.com.au*¹

Abstract

Let $\mathcal{H}_1 = (\mathcal{V}, \mathcal{E}_1)$ be a collection of N pairwise vertex disjoint $\mathcal{O}(1)$ -spanners where the weight of an edge is equal to the Euclidean distance between its endpoints. Let $\mathcal{H}_2 = (\mathcal{V}, \mathcal{E}_2)$ be the graph on \mathcal{V} with M edges of non-negative weight. The union of the two graphs is denoted $\mathcal{G} = (\mathcal{V}, \mathcal{E}_1 \cup \mathcal{E}_2)$. We present a data structure of size $\mathcal{O}(M^2 + n \log n)$ that answers $(1 + \varepsilon)$ -approximate shortest path queries in \mathcal{G} in constant time, where $\varepsilon > 0$ is constant.

1 Introduction

The *shortest-path* (SP) problem for weighted graphs with n vertices and m edges is a fundamental problem for which efficient solutions can now be found in any standard algorithms text, see also [10,14,21,23,24]. Lately the approximation version of this problem has also been studied extensively [1,9,11]. In numerous algorithms, the query version of the SP-problem frequently appears as a subroutine. In such a query, we are given two vertices and have to compute or approximate the shortest path between them. Thorup and Zwick [25] presented an algorithm for undirected weighted graphs that computes $(2k - 1)$ -approximate solutions to the query version of the SP problem in $\mathcal{O}(k)$ time, using a data structure that takes expected time $\mathcal{O}(kmn^{1/k})$ to construct and utilizes $\mathcal{O}(kn^{1+1/k})$ space. It is not an approximation scheme in the true sense

¹ Funded by the Australian Government's Backing Australia's Ability initiative, in part through the Australian Research Council.

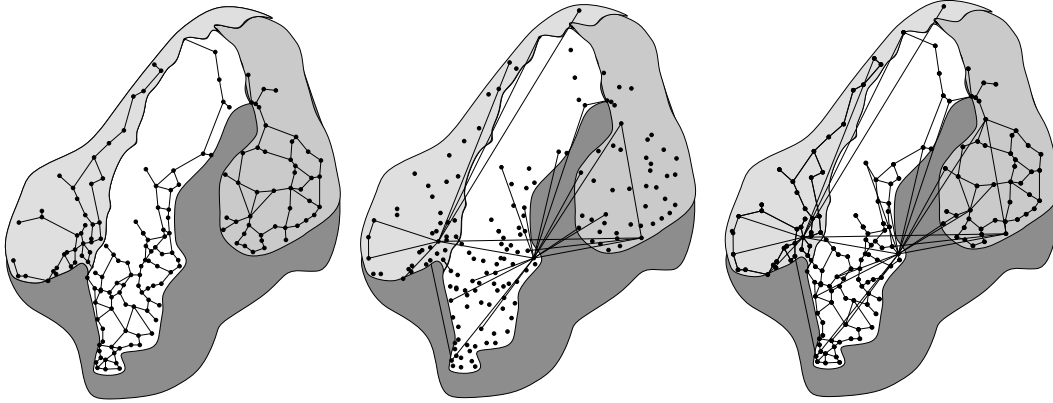


Fig. 1. The three figures models parts of the transportation network in Norway, Sweden and Finland. (a) The domestic railway network in the three countries (not complete). (b) The railway connections between the countries together with the main air and sea connections within, and between, Norway, Sweden and Finland. (c) The two networks combined into one graph \mathcal{G} .

because the value k needs to be a positive integer. Since the query time is essentially bounded by a constant, Thorup and Zwick refer to their queries as approximate *distance oracles*. The time of preprocessing was recently improved by Baswana and Sen in [4].

We focus on the geometric version of this problem. A geometric graph has vertices corresponding to points in \mathbb{R}^d and edge weights from a Euclidean metric. Throughout this paper we will assume that d is a constant. A geometric graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is said to be a t -spanner for \mathcal{V} , if for any two points p and q in \mathcal{V} , there exists a path of length at most t times the Euclidean distance between p and q . For geometric graphs, also, considerable previous work exists on the shortest path and related problems. A good survey can be found in [20], see also [2,6–8,12,13,22]. The geometric query version was recently studied by Gudmundsson et al. [15,16] and they presented the first data structure that answers approximate shortest-path queries in constant time, provided that the input graph is a t -spanner for some known constant $t > 1$. Their data structure uses $\mathcal{O}(n \log n)$ space and can be constructed in time $\mathcal{O}(m + n \log n)$.

In this paper we extend the results in [15,16] to hold also for “islands” of t -spanners, i.e., a set of N vertex disjoint t -spanners inter-connected through “airports” i.e. M edges of arbitrary non-negative weight. We construct a data structure that can answer $(1+\varepsilon)$ -approximate shortest path queries in constant time. The data structure uses $\mathcal{O}(M^2 + n \log n)$ space and can be constructed in time $\mathcal{O}(m + (M^2 + n) \log n)$, where m is the total number of edges. Hence, for $M = \mathcal{O}(\sqrt{n})$ the bound is essentially the same as in [15,16].

We claim that the generalization studied is natural in many applications. Consider for example the freight costs within Norway, Sweden and Finland, see Fig. 1. The railway network and the road network within a country are usu-

ally t -spanners for some small value t , and the weight (transport cost) of an edge is linearly dependent on the Euclidean distance. In Fig. 1a the railway networks of Norway, Sweden and Finland (although not complete) is shown. The weight of an edge is dependent on the Euclidean distance between its endpoints. Hence, each country's railway network and road network can most often be modeled as a Euclidean t -spanner for some small constant t . (Places that are not reachable by train are treated as single t -spanners containing only one point, for example Haugesund on the west coast of Norway is only reachable by boat.). Apart from these edges there are also edges that models, for example, air freight, sea freight, or inter-connecting railway transports. An example of this is shown Fig. 1b, where the main air and sea routes together with the inter-connecting railway tracks are shown. The weight of these edges can be completely independent of the Euclidean distance, as is usually the case when it comes to air fares. The reason why inter-connecting railway transports is included in the latter set of edges is because the railway networks of different countries are usually sparsely connected. For example, there are one connection between Sweden and Finland, two between Norway and Sweden and, zero between Finland and Norway. The same holds for many other adjacent countries, for example, there are three connections between the Netherlands and Germany, two between the Netherlands and Belgium, and three between France and Spain. Finally, note that in most cases M is very small compared to n .

In [15] it was shown that an approximate shortest-path distance oracle can be applied to a large number of problems, for example, finding shortest obstacle-avoiding path between two vertices in a planar polygonal domain with obstacles and interesting query versions of closest pair problems. The extension presented in this paper also generalizes the results for the above mentioned problems.

The main idea for obtaining our results is to develop a method to efficiently combine existing methods for $\mathcal{O}(1)$ -spanners with methods for general graphs. One problem, for example, may be given a starting point p and a destination q , which should be the first airport to travel to, since (in theory) there might be a non-constant number of airports on p 's island? In order to achieve this, we determine a small number, $\mathcal{O}(M)$, of representative "junction" points, so that every point p in the graph is represented by exactly one such junction point $r(p)$, located on the same island as p . All airports are also treated as such junction points. For all pairs of junction points we precompute approximate distances, using space $\mathcal{O}(M^2)$. For any two points p and q , an approximately shortest path between them is found either by only using edges of one of the $\mathcal{O}(1)$ -spanners, or by following a path from p to its representative junction point $r(p)$, then from $r(p)$ to $r(q)$, and finally from $r(q)$ to q . In order to choose such a small set of suitable representative junction points we present, in Section 3.4, a partition of space which may be useful also in other applications.

In Section 4 we show general correctness, and in Section 5 we mention some refinements and extensions of the main results.

2 Preliminaries

Our model of computation is the traditional algebraic computation model with the added power of indirect addressing. We will use the following notation. For points p and q in \mathbb{R}^d , $|p, q|$ denotes the Euclidean distance between p and q . If \mathcal{G} is a geometric graph, then $\delta_{\mathcal{G}}(p, q)$ denotes the Euclidean length of a shortest path in \mathcal{G} between p and q . If P is a path in \mathcal{G} between p and q having length Δ with $\delta_{\mathcal{G}}(p, q) \leq \Delta \leq (1 + \varepsilon) \cdot \delta_{\mathcal{G}}(p, q)$, then P is a $(1 + \varepsilon)$ -approximate shortest path for p and q .

The main result of this paper is stated in the following theorem:

Theorem 1 *Consider two graphs $\mathcal{H}_1 = (\mathcal{V}, \mathcal{F}_1)$ and $\mathcal{H}_2 = (\mathcal{V}, \mathcal{F}_2)$, where \mathcal{H}_1 is a collection of N vertex disjoint Euclidean t -spanners ($t > 1$ is a constant), and \mathcal{H}_2 is a graph with M edges of non-negative weight. The union of the two graphs is denoted $\mathcal{G} = (\mathcal{V}, \mathcal{E} = \{\mathcal{F}_1 \cup \mathcal{F}_2\})$.*

One can construct a data structure in time $\mathcal{O}((|\mathcal{E}| + M^2) \log n)$ using $\mathcal{O}(M^2 + n \log n)$ space that can answer $(1 + \varepsilon)$ -approximate shortest path queries in \mathcal{G} in constant time, where $0 < \varepsilon < 1$ is a given constant.

The set of pairwise vertex disjoint t -spanners of \mathcal{H}_1 is called the “islands” of \mathcal{G} and will be denoted $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1), \dots, \mathcal{G}_N = (\mathcal{V}_N, \mathcal{E}_N)$. An edge $(u, v) \in \mathcal{F}_2$ is said to be an *inter-connecting* edge (even though both its endpoints may belong to the same island). A vertex $v \in \mathcal{V}_i$ incident to an edge in \mathcal{H}_2 is called an *airport*, for simplicity (even though these vertices may represent any kind of junction point). The set of all airports of \mathcal{V}_i is denoted \mathcal{C}_i . Note that the total number of airports is $\mathcal{O}(M)$ since the number of inter-connecting edges is M .

3 Tools

In the construction of the distance oracle we will need several tools, among them the well-separated pair decomposition by Callahan and Kosaraju [5], a graph pruning tool by Gudmundsson et al. [15,17] and well-separated clusters by Krznaric and Levcopoulos [18]. In this section we briefly recollect these tools. In section 3.4 we also show a useful tool that clusters points with respect to a subset of representative points, as described in the introduction.

3.1 Well-separated pair decomposition

Definition 1 [5] Let $s > 0$ be a real number, and let \mathcal{A} and \mathcal{B} be two finite sets of points in \mathbb{R}^d . We say that \mathcal{A} and \mathcal{B} are well-separated with respect to s if there are two disjoint balls $C_{\mathcal{A}}$ and $C_{\mathcal{B}}$, having the same radius, such that $C_{\mathcal{A}}$ contains \mathcal{A} and, $C_{\mathcal{B}}$ contains \mathcal{B} , and the distance between $C_{\mathcal{A}}$ and $C_{\mathcal{B}}$ is at least s times the radius of $C_{\mathcal{A}}$. We refer to s as the separation ratio.

Lemma 1 [5] Let \mathcal{A} and \mathcal{B} be two sets of points that are well-separated with respect to s , let x and x' be two points of \mathcal{A} , and let y and y' be two points of \mathcal{B} . Then $|x, x'| \leq (2/s)|x', y'|$, and $|x', y'| \leq (1 + 4/s)|x, y|$.

Definition 2 [5] Let S be a set of points in \mathbb{R}^d , and let $s > 0$ be a real number. A well-separated pair decomposition (WSPD) for S with respect to s is a sequence $\{\mathcal{A}_i, \mathcal{B}_i\}, 1 \leq i \leq m$, of pairs of non-empty subsets of S , such that

- (1) $\mathcal{A}_i \cap \mathcal{B}_i = \emptyset$ for all $i = 1, \dots, m$,
- (2) for each unordered pair $\{p, q\}$ of distinct points of S , there is exactly one pair $\{\mathcal{A}_i, \mathcal{B}_i\}$ in the sequence, such that (i) $p \in \mathcal{A}_i$ and $q \in \mathcal{B}_i$, or (ii) $q \in \mathcal{A}_i$ and $p \in \mathcal{B}_i$,
- (3) \mathcal{A}_i and \mathcal{B}_i are well-separated with respect to s for all $i = 1, \dots, m$.

The integer m is called the size of the WSPD.

Callahan and Kosaraju show how such a WSPD can be computed. They start by constructing in $\mathcal{O}(n \log n)$ time, a split tree T having the points in S as leaves. Given this tree, they show how a WSPD of size $m = \mathcal{O}(s^d n)$ can be computed in time $\mathcal{O}(s^d n)$. In this WSPD, each pair $\{\mathcal{A}_i, \mathcal{B}_i\}$ is represented by two nodes u_i and v_i of T . That is, \mathcal{A}_i and \mathcal{B}_i are the sets of all points stored at the leaves of the subtrees rooted at u_i and v_i , respectively.

Theorem 2 [5] Let S be a set of points in \mathbb{R}^d , and let $s > 0$ be a real number. A WSPD for S with respect to s having size $\mathcal{O}(s^d n)$ can be computed in $\mathcal{O}(n \log n + s^d n)$ time.

3.2 Pruning a t -spanner

In [15] it was shown that a simple way of pruning an existing t -spanner with m edges into a $(t(1 + \varepsilon))$ -spanner with $\mathcal{O}(n)$ edges was to use the WSPD described in the previous section.

Assume that we are given a t -spanner $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Compute a WSPD $\{\mathcal{A}_i, \mathcal{B}_i\}$,

$1 \leq i \leq \ell$, for \mathcal{V} , with separation constant $s = 4(1 + (1 + \varepsilon)t)/\varepsilon$ and $\ell = O(n)$. Let $\mathcal{G}' = (\mathcal{V}, \mathcal{E}')$ be the graph that contains for each i , exactly one (arbitrary) edge (x_i, y_i) of E with $x_i \in \mathcal{A}_i$ and $y_i \in \mathcal{B}_i$, provided such an edge exists. It holds that \mathcal{G}' is a $(1 + \varepsilon)$ -spanner of \mathcal{G} , and hence:

Fact 1 (Corollary 1 in [17]) *Given a real constant $\varepsilon > 0$ and a t -spanner $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, for some real constant $t > 1$, with n vertices and m edges, one can compute a $(1 + \varepsilon)$ -spanner \mathcal{G}' of \mathcal{G} with $O(n)$ edges in time $O(m + n \log n)$.*

3.3 Well-Separated Clusters

Let \mathcal{S} be a set of points in the plane, and let $b \geq 1$ be a real constant. Let the *rectangular diameter* of $\mathcal{A} \in \mathcal{S}$, abbreviated $rd(\mathcal{A})$, be the diameter of smallest axis-aligned rectangle containing \mathcal{A} . We may now consider the following cluster definitions from [18]:

Definition 3 *A subset \mathcal{A} of \mathcal{S} is a b -cluster if \mathcal{A} equals \mathcal{S} or the distance between any point of \mathcal{A} and any point of $\mathcal{S} - \mathcal{A}$ is greater than $b \cdot rd(\mathcal{A})$.*

Definition 4 *The hierarchy of b -clusters of \mathcal{S} is a rooted tree whose nodes correspond to distinct b -clusters, such that the root corresponds to \mathcal{S} and leaves to single points of \mathcal{S} . Let $\nu(\mathcal{A})$ be any internal node and let \mathcal{A} be its corresponding b -cluster. The children of $\nu(\mathcal{A})$ correspond to every b -cluster \mathcal{C} such $\mathcal{C} \subset \mathcal{A}$ and there is no b -cluster \mathcal{B} such that $\mathcal{C} \subset \mathcal{B} \subset \mathcal{A}$.*

The following observation is straightforward.

Observation 1 *Let \mathcal{A} and \mathcal{B} be two distinct b -clusters, and let x and x' be two points of \mathcal{A} , and let y and y' be two points of \mathcal{B} . Then $|x', y'| \leq (1 + 2/b)|x, y|$.*

PROOF. Assume w.l.o.g that $|x', y'| \geq |x, y|$ and that $rd(\mathcal{A}) \geq rd(\mathcal{B})$. This means $|x', y'| \leq |x, y| + 2rd(\mathcal{A}) \leq |x, y| + 2|x, y|/b = (1 + 2/b)|x, y|$. \square

The cluster tree can also be computed efficiently.

Theorem 3 *Let \mathcal{S} be a set of n points in \mathbb{R}^d and a real constant $b \geq 1$, the hierarchy of b -clusters of \mathcal{S} can be computed in $O(n \log n)$ time and space.*

PROOF. The hierarchy of b -clusters can easily be computed in $O(n \log n)$ time for any constant number of dimensions d , e.g., by using a hierarchical cluster decomposition according to the complete-linkage criterion in the L_0 -metric (see Krznicaric and Levkopoulos [19]), since each such b -cluster is also a cluster in the complete-linkage hierarchy. \square

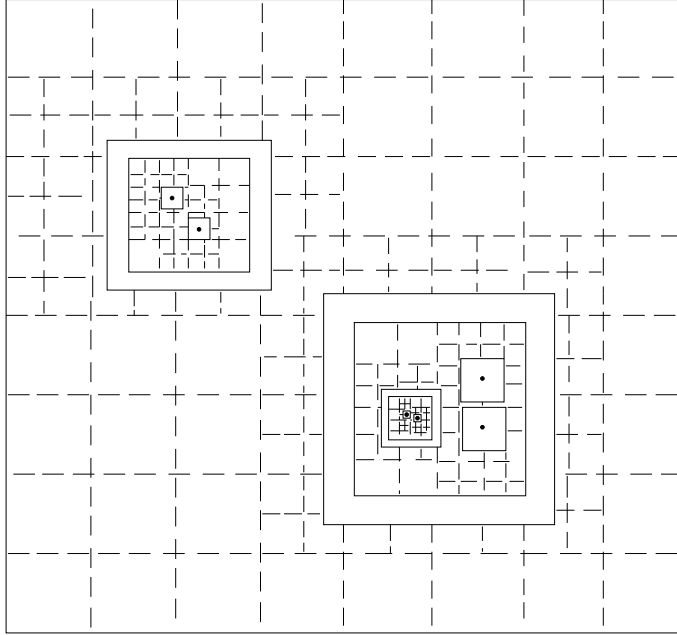


Fig. 2. An example cell partition, with respect to \mathcal{V}' , made by the algorithm. Doughnuts are drawn with solid lines, while inner cells are drawn with dashed lines.

3.4 Partitioning space into small cells

In this section, given a set \mathcal{V} of n points in \mathbb{R}^d , and a subset $\mathcal{V}' \subseteq \mathcal{V}$, we show how to associate a representative point $r \in \mathcal{V}$ to each point $p \in \mathcal{V}$, such that the distance $|p, r| + |r, q|$, for any point $q \in \mathcal{V}'$, is a good approximation of the distance $|p, q|$. The total number of representative points is $\mathcal{O}(|\mathcal{V}'|)$. The idea is to partition space into cells, such that all points included in a cell may share a common representative point.

We will use the following fact by Arya et al. [3]:

Fact 2 (Theorem 1 in [3]) Consider a set \mathcal{S} of n points in \mathbb{R}^d . There is a constant $c_{d,\varepsilon} \leq d[1 + 6d/\varepsilon]^d$, such that in $\mathcal{O}(dn \log n)$ time it is possible to construct a data structure of size $\mathcal{O}(dn)$, such that for any Minkowski metric:

- (i) Given any $\varepsilon > 0$ and $q \in \mathbb{R}^d$, a $(1 + \varepsilon)$ -approximate nearest neighbor of q in \mathcal{S} can be reported in $\mathcal{O}(c_{d,\varepsilon} \log n)$.
- (ii) More generally, given $\varepsilon_N > 0$, $q \in \mathbb{R}^d$, and any k , $1 \leq k \leq n$, a sequence of k $(1 + \varepsilon)$ -approximate nearest neighbors can be computed in $\mathcal{O}((c_{d,\varepsilon} + kd) \log n)$ time.

3.4.1 Computing representative points

As a pre-processing step we compute the b -cluster tree \mathcal{T} of \mathcal{V}' with $b = 10/\varepsilon^2$, as described in Theorem 3.

For a level i in \mathcal{T} let $\nu(\mathcal{D}_1), \dots, \nu(\mathcal{D}_{\ell_i})$ be the nodes at that level, where $\mathcal{D}_1, \dots, \mathcal{D}_{\ell_i}$ are the associated clusters. For each cluster \mathcal{D}_j pick an arbitrary vertex d_j as the center point of \mathcal{D}_j . The set of the ℓ_i center points is denoted $\mathcal{D}(i)$. Perform the following four steps for each level i of \mathcal{T} .

- (1) Compute an approximate nearest neighbor structure with $\mathcal{D}(i)$ as input, as described in Fact 2.
- (2) For each center point d_j in $\mathcal{D}(i)$ compute the $(1 + \varepsilon)$ -approximate nearest neighbor of d_j . The point returned by the structure is denoted v_j , where $v_j \neq d_j$.
- (3) For each cluster \mathcal{D}_j construct two squares; $is(\mathcal{D}_j)$ and $os(\mathcal{D}_j)$ with centers at d_j and side length $2\alpha = 2(1 + 1/\varepsilon) \cdot rd(\mathcal{D}_j)$ and $2\beta = \frac{2\varepsilon|d_j, v_j|}{(1+\varepsilon)(1+2/b)}$ respectively, where $\alpha < \beta$. The two squares are called the inner and outer shells of \mathcal{D}_j , and the set theoretical difference between the inner and the outer shell is denoted the *doughnut* of \mathcal{D}_j .
- (4) The inner shell of \mathcal{D}_j is recursively partitioned into four equally sized squares, until each square s either
 - (a) is completely included in the union of the outer shells of the children of $\nu(\mathcal{D}_j)$. In this case the square is deleted and, hence, not further partitioned. Or,
 - (b) has diameter at most $\frac{\varepsilon}{1+\varepsilon} \cdot K$, where K is the smallest distance between a point within s and a point in \mathcal{D}_j . A $(1 + \varepsilon)$ -approximation of K can be computed in time $\mathcal{O}(\log |\mathcal{D}_j|)$. This implies that the diameter of s is bounded by $\varepsilon \cdot K$.

The resulting cells are denoted *inner cells*. Note that, due to step 4a, every inner cell is empty of points from \mathcal{D}_j .

Finally, after all levels of \mathcal{T} have been processed, we assign a representative point, $r(p)$, to each point p in \mathcal{V} . Preprocess all the produced cells and perform a point-location query for each point. If p belongs to a doughnut cell then the center point of the associated cluster (see step 1) is the representative point of p . Otherwise, if p belongs to an inner cell C and p is the first point within C processed in this step then $r(C)$ is set to p . If p is not the first point then $r(p) = r(C)$. Further, note that an inner cell may overlap with the union of the outer shells of the children of $\nu(\mathcal{D}_j)$. If a point is included in both an inner cell and an outer shell, we treat it as if it belonged to the inner cell, and assign a representative point as above.

3.4.2 The analysis

Below we prove the main result of this section.

Theorem 4 *Given a set \mathcal{V} of n points in \mathbb{R}^d , a subset $\mathcal{V}' \subseteq \mathcal{V}$ and a positive real value $\tau_1 < 1$, the above algorithm associate for each point $p \in \mathcal{V}$ a representative point $r(p) \in \mathcal{V}$ such that for any point $q \in \mathcal{V}'$, it holds that*

$$\min\{|p, r(p)|, |r(p), q|\} \leq \tau_1 |p, q|.$$

The number of representative points is $\mathcal{O}(|\mathcal{V}'|)$ and they can be computed in time $\mathcal{O}(n \log n)$.

The proof of Theorem 4 is partitioned into three steps: first we show that for each vertex $v \in \mathcal{V}$ the algorithm always choose a good representative point, then it will be shown that the total number of representative points is $\mathcal{O}(|\mathcal{V}'|)$, and finally we prove the time-complexity of the algorithm.

Lemma 2 *For each point $p \in \mathcal{V}$ and for every point $q \in \mathcal{V}'$ it holds that*

$$\min\{|p, r(p)|, |r(p), q|\} \leq \tau_1 |p, q|.$$

PROOF. Consider the above algorithm and select a positive constant $\varepsilon = (1 + \sqrt{2})\tau_1/\sqrt{8}$. For each point $p \in \mathcal{V}$ we distinguish between two cases depending on the cell C in which p lies, either in an inner cell or a doughnut.

inner cell: Let $p' \in \mathcal{V}'$ be the nearest neighbor of p in \mathcal{V}' . The distance between a point in C , and its nearest neighbor, can differ at most $\text{rd}(C)$ from the distance between any other point in C and its nearest neighbor. Thus, from the way C was created it is straight-forward to see that $\text{rd}(C) \leq \varepsilon \cdot |p, p'|$ and thus

$$\min\{|p, r(p)|, |r(p), q|\} = |p, r(p)| \leq \text{rd}(C) \leq \varepsilon |p, p'| \leq \varepsilon |p, q| \leq \tau_1 |p, q|.$$

doughnut: Let $cl(C)$ denote the cluster that was processed when C was created. We distinguish between two cases, either $q \in cl(C)$, or not.

- **$q \in cl(C)$:** From the algorithm it holds that $|p, r(p)| \geq \alpha - \text{rd}(cl(C))$ and that $|r(p), q| \leq \text{rd}(cl(C))$, which means that $\min\{|p, r(p)|, |r(p), q|\} = |r(p), q| \leq \text{rd}(cl(C))$. It holds that $\text{rd}(cl(C))$ can be rewritten as $\varepsilon((1 + 1/\varepsilon) \cdot \text{rd}(cl(C)) - \text{rd}(cl(C)))$, hence since $\alpha = (1 + 1/\varepsilon) \cdot \text{rd}(cl(C))$ and since $\text{rd}(cl(C)) \leq \varepsilon |p, q|$ we get

$$\begin{aligned}
\min\{|p, r(p)|, |r(p), q|\} &= |r(p), q| \\
&\leq rd(cl(C)) \\
&\leq \varepsilon(\alpha - rd(cl(C))) \\
&\leq \varepsilon|p, r(p)| \\
&\leq \varepsilon(|p, q| + rd(cl(C))) \\
&\leq 2\varepsilon|p, q| \leq \tau_1|p, q|
\end{aligned}$$

- $\mathbf{q} \notin \mathbf{cl}(C)$: We know that from the assignment of representative points that $|p, r(p)| \leq 2\sqrt{2}\beta$. Let d_i be the center point of $cl(C)$ chosen by the algorithm, let $d'_i \in \mathcal{V}' \setminus c(C)$ be the point closest to d_i and let $p' \in \mathcal{V}' \setminus cl(C)$ be the point closest to p . From the definition of β it holds that $\beta \leq \varepsilon|d_i, d'_i| \leq \varepsilon|p, p'| - \sqrt{2}\beta \Rightarrow \beta \leq \varepsilon|p, p'|/(1 + \sqrt{2})$. As a result we get

$$\begin{aligned}
\min\{|p, r(p)|, |r(p), q|\} &\leq |p, r(p)| \\
&\leq 2\sqrt{2}\beta \\
&\leq 2\sqrt{2}\varepsilon|p, p'|/(1 + \sqrt{2}) \\
&\leq 2\sqrt{2}\varepsilon|p, q|/(1 + \sqrt{2}) \\
&= \tau_1|p, q|.
\end{aligned}$$

This concludes the proof of the lemma. \square

Lemma 3 *The number of representative points is $\mathcal{O}(|\mathcal{V}'|)$.*

PROOF. For each cluster there is (at most) one doughnut cell, thus $|\mathcal{V}'|$ in total, hence we only need to bound the number of inner cells.

Intuitively this is done as follows. Given a b -cluster \mathcal{D} let $\nu(\mathcal{D})$ be the node in \mathcal{T} associated with \mathcal{D} and let $\mathcal{D}_1, \dots, \mathcal{D}_\ell$ be the cluster associated with the children of $\nu(\mathcal{D})$. It will be shown that the number of inner cells of \mathcal{D} is $\mathcal{O}(\ell)$, which means that we have $\mathcal{O}(|\mathcal{V}'|)$ inner cells in total. For each cluster \mathcal{D}_i let d_i be the center point of \mathcal{D}_i , the set of these points is denoted \mathcal{K} and $|\mathcal{K}| = \ell$. In the analysis we will consider the WSPD of \mathcal{K} with a constant s as separation constant. For each well-separated pair $\{\mathcal{A}_i, \mathcal{B}_i\}$ we consider a disc of radius $\Theta(dist(\mathcal{A}_i, \mathcal{B}_i))$ surrounding the pair. We let each such disc “pay” for all cells of approximate size $dist(\mathcal{A}_i, \mathcal{B}_i)$ included in the circle, which, according to standard packing arguments, is a constant number of cells. We then show that each cell is paid for by at least one disc. Since the size of the WSPD according to Theorem 2 is $\mathcal{O}(\ell)$, it holds that the number of cells is $\mathcal{O}(\ell)$.

We are now ready to give a more detailed analysis. Using Observation 1 and Lemma 1 we obtain the following observation, used throughout this proof:

Observation 2 *Consider a well-separated pair $\{\mathcal{A}_i, \mathcal{B}_i\}$ and two center points $d_j \in \mathcal{A}_i$ and $d_k \in \mathcal{B}_i$, with corresponding b -clusters \mathcal{D}_j and \mathcal{D}_k . Given points $x \in \mathcal{A}_i$, $y \in \mathcal{B}_i$, $x' \in \mathcal{D}_j$ and $y' \in \mathcal{D}_k$, then $|x', y'| \leq (1 + 4/s)(1 + 2/b)|x, y|$.*

For simplicity of writing we will set $\tau = (1 + 4/s)(1 + 2/b)$. Next, for each well-separated pair $\{\mathcal{A}_i, \mathcal{B}_i\}$ choose two arbitrary points $a_i \in \mathcal{A}_i$ and $b_i \in \mathcal{B}_i$. Let \mathcal{C}_i be the disc with center at a_i and of radius $17\tau(1 + 1/\varepsilon) \cdot |a_i, b_i|$. The aim is to show that each cell is paid for, that is, given a cell c we need to show that there exists a pair $\{\mathcal{A}_i, \mathcal{B}_i\}$ such that

- (i) c intersects the disc \mathcal{C}_i , and
- (ii) $\frac{1}{\gamma} \cdot rd(c) \leq |a_i, b_i| \leq \delta \cdot rd(c)$, where $\delta = \frac{68\tau(1+\varepsilon)(1+2/b)}{\varepsilon^2}$ and $\gamma = 4\tau$.

That is, the cell must overlap the disc surrounding $\{\mathcal{A}_i, \mathcal{B}_i\}$ and its diameter must be comparable to the distance between the points in \mathcal{A}_i and the points in \mathcal{B}_i .

Consider an arbitrary inner cell c , let p be an arbitrary point of \mathcal{V} within c and let q be the nearest neighbor in \mathcal{V}' of p . Assume w.l.o.g. that $q \in \mathcal{D}_1$, and recall that d_1 is the center point of \mathcal{D}_1 . Finally, let r be the point in $\{\mathcal{V}' \setminus \mathcal{D}_1\}$ closest to d_1 . We will first show that there must exist well-separated pairs such that (i) holds. This will be shown by contradiction, where we distinguish between three cases:

Case 1: $|a_i, b_i| > \delta \cdot rd(c)$ for all pairs $\{\mathcal{A}_i, \mathcal{B}_i\}$ that contain d_1 .

Since the bound holds for every well-separated pair containing d_1 it especially holds that $|d_1, r| > \frac{\delta}{\tau} \cdot rd(c) = \frac{68 \cdot rd(c)}{\varepsilon^2}$. Considering the side length β of $os(\mathcal{D}_1)$ it holds from the algorithm, and especially from the way that c was constructed, that

$$\beta > \frac{\varepsilon \delta}{\tau(1 + \varepsilon)(1 + 2/b)} \cdot rd(c) = \frac{68}{\varepsilon} \cdot rd(c)$$

and

$$rd(c) > \frac{\varepsilon}{2} \left(\frac{|p, q|}{1 + \varepsilon} - \sqrt{2} \cdot rd(c) \right) > \frac{\varepsilon}{2} \left(\frac{1}{2} |p, q| - \sqrt{2} \cdot rd(c) \right).$$

The second inequality can be rewritten and simplified: $|p, q| + rd(c) < \frac{8}{\varepsilon} \cdot rd(c)$. Now, since $|p, q| + rd(c) < \frac{8}{\varepsilon} \cdot rd(c) < \frac{\beta}{8}$ it holds that c is completely included in the outer shell of \mathcal{D}_1 , which is a contradiction since c then would have been removed by the algorithm (step 4a).

Case 2: $|a_i, b_i| < rd(c)/\gamma$ for all pairs $\{\mathcal{A}_i, \mathcal{B}_i\}$ that contain d_1 .

This means that $rd(\mathcal{V}') \leq \frac{\tau \cdot rd(c)}{\gamma}$, which can be rewritten as $rd(c) \geq \frac{\gamma \cdot rd(\mathcal{V}')}{\tau}$. From the construction of the inner cells it holds that $|p, q| \geq rd(c)/\varepsilon$. Combining these two inequalities we obtain that

$$|p, q| \geq \frac{rd(c)}{\varepsilon} \geq \frac{\gamma \cdot rd(\mathcal{V}')}{\varepsilon \cdot \tau} = \frac{4 \cdot rd(\mathcal{V}')}{\varepsilon} > 2\alpha.$$

Thus, c must lie partly outside the inner shell, which is a contradiction since

c was created by partitioning the inner shell.

Case 3: Neither Case 1 or 2 holds.

This means we have at least one well-separated pair such that $|a_i, b_i| < rd(c)/\gamma$, and at least one pair such that $|a_i, b_i| > \delta \cdot rd(c)$ for all pairs $\{\mathcal{A}_i, \mathcal{B}_i\}$ in which d_1 is included.

First consider the union of all well-separated pairs where $|a_i, b_i| < rd(c)/\gamma$. Each point in this union is included in one cluster \mathcal{D}_i . Consider the union of all such clusters, denoted \mathcal{U} . We have, for any point $u \in \mathcal{U}$ that $|d_1, u| < \tau \cdot rd(c)/\gamma \Rightarrow rd(\mathcal{U}) \leq 2\sqrt{2}\tau \cdot rd(c)/\gamma$.

Next consider all well-separated pairs $\{\mathcal{A}_i, \mathcal{B}_i\}$ such that $|a_i, b_i| > \delta \cdot rd(c)$, and assume w.l.o.g. that $d_1 \in \mathcal{A}_i$. Each point in the union of all \mathcal{B}_i 's is included in one cluster \mathcal{D}_i , and we let \mathcal{U}' denote the union of all these clusters. Let u and u' be two points in \mathcal{U} and \mathcal{U}' , respectively. We have that

$$|u, u'| \geq |d_1, u'| - rd(\mathcal{U}) \geq \frac{\delta \cdot rd(c)}{\tau} - rd(\mathcal{U}) \geq \frac{67}{\varepsilon^2} \cdot rd(\mathcal{U}) \geq b \cdot rd(\mathcal{U}),$$

since $b = 10/\varepsilon^2$. Thus \mathcal{U} must be a b -cluster, which means there cannot exist well-separated pairs such that $rd(c)/\gamma > |a_i, b_i|$, which is a contradiction.

We have shown that for every cell c there exists a well-separated pair $\{\mathcal{A}_i, \mathcal{B}_i\}$ such that (i) holds, and such that $d_1 \in \{\mathcal{A}_i \cup \mathcal{B}_i\}$. In order to show (ii), we consider a well-separated pair $\{\mathcal{A}_i, \mathcal{B}_i\}$ that contains d_1 and such that $|a_i, b_i| \geq rd(c)/\gamma$. Recall that the radius of \mathcal{C}_i is $17\tau(1 + 1/\varepsilon) \cdot |a_i, b_i|$. Let x denote the center of \mathcal{C}_i , it holds that

$$|p, x| < |p, q| + \tau \cdot |a_i, b_i| \leq \left(\frac{2}{\varepsilon} + 1\right) \cdot rd(c) + \tau \cdot |a_i, b_i| < 8\tau \left(1 + \frac{1}{\varepsilon}\right) \cdot |a_i, b_i|.$$

Since the distance from the center \mathcal{C}_i to p is less than the radius of \mathcal{C}_i it follows that c must overlap \mathcal{C}_i and, hence, (ii) holds. In conclusion, we have $\mathcal{O}(|\mathcal{V}'|)$ cells and, since every cell has at most one representative point, the number of representative points is also $\mathcal{O}(|\mathcal{V}'|)$. \square

Lemma 4 *The representative points can be computed in time $\mathcal{O}(n \log n)$.*

PROOF. The preprocessing, building the b -cluster tree and selecting the center points of each cluster takes $\mathcal{O}(n \log n)$ time in total. An approximate nearest neighbor data structure is constructed on each level, but the total number of elements involved, summing over all levels, is at most $2|\mathcal{V}'|$. This follows since the number of leaves in \mathcal{T} is $|\mathcal{V}'|$, and hence the total number of nodes in \mathcal{T} is $2|\mathcal{V}'|$. It immediately follows that step 2 takes $2|\mathcal{V}'| \cdot \mathcal{O}(\log |\mathcal{V}'|)$ time in total and step 3 takes $\mathcal{O}(|\mathcal{V}'|)$ time.

The final step of the algorithm is done by performing n point-location queries, each taking $\mathcal{O}(\log n)$ time.

Hence, it remains to bound step 4, which is equivalent to bound the total number of squares considered during the partition. From Lemma 3, we know that the number of cells in the partition is $\mathcal{O}(|\mathcal{V}'|)$. However, the running time of the algorithm depends on the total number of squares considered during the construction of the partitioning. We will below show that the total number of squares also is bounded by $\mathcal{O}(|\mathcal{V}'|)$.

Consider an inner shell \mathcal{I} and its corresponding b -cluster \mathcal{D} . Let $v(\mathcal{D})$ be the node in \mathcal{T} associated with \mathcal{D} and let $\mathcal{D}_1, \dots, \mathcal{D}_\ell$ be the clusters associated with the children of $\nu(\mathcal{D})$. Further, consider the partition tree \mathcal{Y} of \mathcal{I} , where the nodes correspond to squares in the natural way, and the leaf nodes correspond to either:

- (i) inner cells included in the final partitioning, or
- (ii) squares which were removed because they were completely included in an outer shell of $\mathcal{D}_1, \dots, \mathcal{D}_\ell$, and thus not partitioned further.

Let f be a node in \mathcal{Y} and let c be its corresponding cell, such that the corresponding cell of one of its sibling nodes was removed due to inclusion in the outer shell $os(\mathcal{D}_i)$ of a cluster \mathcal{D}_i .

Since the cell of the sibling node was removed, and sibling cells are of equal size, it is clear that c must be smaller than $os(\mathcal{D}_i)$. Next, consider a leaf node $g \in \mathcal{Y}$ of type (i) and its corresponding cell c' resulting from the continued partitioning of c . We know that $os(\mathcal{D}_i)$ is greater than $is(\mathcal{D}_i)$, which, in turn is larger than $rd(\mathcal{D}_i)$ multiplied by some large constant factor based on ε . Further, since a part of g lies outside $os(\mathcal{D}_i)$ and is not partitioned further once it is at a sufficient distance from the points in \mathcal{D}_i , it is straight-forward to see that the size of c' is at least $rd(\mathcal{D}_i)$ multiplied by some constant factor including ε . Thus c is only a constant factor larger than c' , which means that c' was constructed by partitioning c a constant number of times.

Our goal is to show that we have $O(\ell)$ nodes in \mathcal{Y} , since this immediately implies that the total number of considered squares are $\mathcal{O}(|\mathcal{V}'|)$. From Lemma 2 we know that we have $\mathcal{O}(\ell)$ leaf nodes of type (i). Consider such a type (i) leaf node $g \in \mathcal{Y}$, and all ancestors $\mathcal{A}(g)$ up to the first node such that none of its children is a type (ii) node. We let g pay for all children of all nodes in $\mathcal{A}(g)$, which are of type (ii). From the above reasoning it is straight-forward to see that each type (i) leaf node pays for a constant number of type (ii) leaf nodes. Further each type (ii) leaf node $h \in \mathcal{Y}$ must be payed for by a type (i) leaf node. This holds, since at least one of its siblings h' is not a type (ii) leaf node (if all siblings were of type (ii) then their parents would be type (ii), which is a contradiction), and thus h must be payed for by the leaf node of type (i) resulting from the partitioning of h' .

This means that the total number of leaf nodes (type either *(i)* or *(ii)*) are $\mathcal{O}(\ell)$. Further, since the number of internal nodes in \mathcal{Y} is at most a constant factor larger than the total number of leaf nodes it follows that the total number of nodes in \mathcal{Y} is $\mathcal{O}(\ell)$, and thus, the total number of considered squares is $\mathcal{O}(|\mathcal{V}'|)$. \square

4 Constructing the Oracle

This section is divided into three subsections: first we present the construction of the structure, then how queries are answered and, finally the analysis is presented.

Consider two graphs $\mathcal{H}_1 = (\mathcal{V}, \mathcal{F}_1)$ and $\mathcal{H}_2 = (\mathcal{V}, \mathcal{F}_2)$ with the same vertex set, where \mathcal{H}_1 is a collection of N vertex disjoint Euclidean t -spanners $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$, $1 \leq i \leq N$, with m edges where $t > 1$ is a constant, and \mathcal{H}_2 is a graph with M edges of non-negative weight. The union of the two graphs is denoted $\mathcal{G} = (\mathcal{V}, \mathcal{E} = \{\mathcal{F}_1 \cup \mathcal{F}_2\})$.

4.1 Constructing the basic structures

In this section we show how to pre-process \mathcal{G} in time $\mathcal{O}(m + (M^2 + n) \log n)$ such that we obtain three structures that will help us answer $(1 + \varepsilon)$ -approximate distance queries in constant time. We will assume that the number of edges in each subgraph is linear with respect to the number of vertices in \mathcal{V}_i , if not the subgraph is pruned using Fact 1. Hence, we can from now on assume that $|\mathcal{E}_i| = \mathcal{O}(|\mathcal{V}_i|)$.

Let \mathcal{V}' be the set of vertices in \mathcal{V} incident on an inter-connecting edge. Now we can apply Theorem 4 with parameters \mathcal{V} , $\mathcal{V}' = \Gamma'$ and τ_1 to obtain a representative point for each point in \mathcal{V} .

Now we are ready to present the three structures:

Oracle A: An oracle that given points p and q returns a 3-tuple $[SI, r(p), r(q)]$, where SI is a boolean with value ‘true’ if p and q belongs to the same island, otherwise it is ‘false’, and $r(p)$ and $r(q)$ is the representative points for p and q respectively.

Oracle B: An $(1 + \varepsilon)$ -approximate distance oracle for any pair of points belonging to the same island.

Matrix D: An $\mathcal{O}(M) \times \mathcal{O}(M)$ matrix. For each pair of representative points, p and q , D contains the $(1 + \varepsilon)$ -approximate shortest distance between p and q .

The representative point of a point p is denoted $r(p)$, and the set of all representative points of \mathcal{V}_i and \mathcal{V} is denoted Γ_i and Γ , respectively. Note that $\mathcal{C}_i \subseteq \Gamma_i$. Now we turn our attention to the construction of the oracles and the matrix.

4.1.1 Oracle A:

The oracle is a 4-level tree, denoted \mathcal{T} , with the points of \mathcal{V} corresponding to the leaves of \mathcal{T} . The parents of the leaves correspond to the representative points of \mathcal{V} and their parents correspond to the islands $\mathcal{G}_1, \dots, \mathcal{G}_N$ of \mathcal{G} . Finally, the root of \mathcal{T} corresponds to \mathcal{G} . Since the representative points already are computed, the tree \mathcal{T} can be constructed in linear time. The root is at level 0 and the leaves are at level 3 in \mathcal{T} .

Assume that one is given two points p and q . Follow the paths from p and q respectively to the root of \mathcal{T} . If p and q have the same ancestor at level 1 then they lie on the same island and hence SI is set to ‘true’, otherwise to ‘false’. Finally, the ancestor of p and the ancestor of q at level 2 corresponds to the representative points of p and q . Obviously a query can be answered in constant time since the number of levels in \mathcal{T} is four.

4.1.2 Oracle B:

This oracle is the structure that is easiest to build since we can apply the following result to each of the islands. Combining Theorem 1 in [16] with Corollary 1 in [17] we get the following fact (see also [15]):

Fact 3 *Let \mathcal{V} be a set of n points in R^d , let τ_2 be a positive real constant and let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a t -spanner for \mathcal{V} , for some real constant $t > 1$, having m edges. In $\mathcal{O}(n \log n)$ time we can preprocess \mathcal{G} into a data structure of size $\mathcal{O}(n \log n)$, such that for any two points p and q in \mathcal{V} , we can in constant time compute a $(1 + \tau_2)$ -approximation to the shortest-path distance in \mathcal{G} between p and q .*

Hence, oracle B will actually be a collection of oracles, one for each island. Given two points p and q the appropriate oracle can easily be found in constant time using a similar construction as for oracle A . Thus, after $\mathcal{O}(n \log n)$ preprocessing using $\mathcal{O}(n \log n)$ space, $(1 + \tau_2)$ -approximate shortest path queries between points on the same island can be answered in constant time.

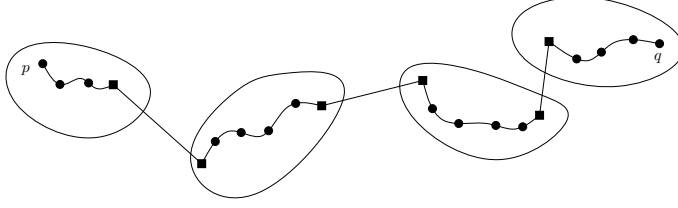


Fig. 3. Illustrating the approximate shortest path between p and q . The boxes illustrate the “airports” along the path.

4.1.3 Matrix D :

For each i , $1 \leq i \leq N$, compute the WSPD of Γ_i with separation constant $s = \left(\frac{1+\tau_2+\tau_3}{\tau_3-\tau_2}\right)$. As output we obtain a set of well-separated pairs $\{\mathcal{A}_i, \mathcal{B}_i\}_{1 \leq i \leq w_i}$, such that $w_i = \mathcal{O}(|\mathcal{C}_i|)$. Next, construct the non-Euclidean graph $\mathcal{F} = (\Gamma, \mathcal{E}')$ as follows. For each Γ_i and each well-separated pair $\{\mathcal{A}_j, \mathcal{B}_j\}$ of the WSPD of Γ_i select two (arbitrary) representative points $a_j \in \mathcal{A}_j$ and $b_j \in \mathcal{B}_j$. Add the edge (a_j, b_j) to \mathcal{E}' with weight $B_i(a_j, b_j)$, where $B_i(p, q)$ denotes a call to oracle B_i for \mathcal{G}_i with parameters p and q . Note that the graph \mathcal{F} will have $\mathcal{O}(M)$ vertices and edges.

Let D be an $\mathcal{O}(M) \times \mathcal{O}(M)$ matrix. For each representative point $p \in \Gamma$ compute the single-source shortest path in \mathcal{F} to every point q in Γ and store the distance of each path in $D[p, q]$. The total time for this step is $\mathcal{O}(M^2 \log M)$, and it can be obtained by running Dijkstra’s algorithm M times.

Lemma 5 *The oracles A and B , and the matrix D can be built in time $\mathcal{O}(m + (M^2 + n) \log n)$ and the total complexity of A , B and M is $\mathcal{O}(M^2 + n \log n)$.*

PROOF. The lemma is obtained by adding up the complexity for the pre-processing together with the cost of building each structure. Recall that as pre-processing steps we first pruned the subgraphs and then we computed the representative point for each point in \mathcal{V} . This was done in $\mathcal{O}(m + n \log n)$ time using $\mathcal{O}(n \log n)$ space, according to Fact 1 and Theorem 4. Next, oracle A was constructed in linear time using linear space, followed by the construction of oracle B which, according to Fact 3 was done in $\mathcal{O}(n \log n)$ time using $\mathcal{O}(n \log n)$ space. Finally, the matrix D was constructed by first computing the graph \mathcal{F} . Then a single-source shortest path query was performed for each vertex in \mathcal{F} . Since the complexity of \mathcal{F} is $\mathcal{O}(M)$ it follows that D was computed in time $\mathcal{O}(M^2 \log n)$ using $\mathcal{O}(M^2)$ space. Hence, adding these bounds gives the lemma. \square

4.2 Querying

Given the two oracles and the matrices presented above the query algorithm is very simple, see pseudo-code below. Let $r(p)$ denote the representative point of $p \in \mathcal{V}$. Now assume that we are given two points p and q . If p and q belong to the same islands then we query Oracle B with input p, q and return the value obtained from the oracle. If p and q does not belong to the same island we return the sum of $B(p, r(p))$, $D(r(p), r(q))$ and $B(r(q), q)$. Obviously this is done in constant time.

QUERY(p, q)

1. [$\text{SameIsland}, r(p), r(q)$] $\leftarrow A(p, q)$
2. $distance \leftarrow B(p, r(p)) + D(r(p), r(q)) + B(r(q), q)$
3. **if** SameIsland **then**
4. $distance \leftarrow \min(distance, B(p, q))$
5. return $distance$

4.3 Correctness

Let $\delta_{\mathcal{G}}(p, q)$ be a shortest path in a graph \mathcal{G} between two points p and q .

Observation 3 *Let p and q be any pair of points in \mathcal{V}_i it holds that $B(p, q) \leq (1 + \tau_2) \cdot \delta_{\mathcal{G}_i}(p, q)$.*

Observation 4 *Given a point $p \in \mathcal{V}_i$ it holds that $\delta_{\mathcal{G}_i}(p, r(p)) \leq (1 + \tau_4) \cdot \delta_{\mathcal{G}}(p, r(p))$.*

PROOF. Let h be the first point in \mathcal{C}_i along the path $\delta_{\mathcal{G}}(p, r(p))$ from p to $r(p)$. If $r(p) = h$ then we are done, otherwise we will have two cases according to Theorem 4.

(a) If $|p, r(p)| \leq \tau_1 \cdot |p, h|$ then $\delta_{\mathcal{G}_i}(p, r(p)) \leq \tau_1 t \cdot |p, h|$ which is less than $|p, h|$, hence this case cannot occur.

(b) Otherwise, $|r(p), h| \leq \tau_1 \cdot |p, h|$ and hence $\delta_{\mathcal{G}_i}(p, r(p)) \leq \delta_{\mathcal{G}}(p, h) + \delta_{\mathcal{G}}(h, r(p)) \leq (1 + \tau_1 t) \cdot \delta_{\mathcal{G}}(p, h)$. The observation follows by setting $\tau_4 = t \cdot \tau_1$. \square

Lemma 6 *Let p and q be any pair of representative points in \mathcal{V}_i it holds that $D(p, q) \leq (1 + \tau_3) \cdot \delta_{\mathcal{G}_i}(p, q)$.*

PROOF. Note that it suffices to prove that $D(p, q) \leq \frac{1+\tau_3}{1+\tau_2} \cdot B(p, q)$, according to Observation 3. The proof is done by induction on the Euclidean length of (p, q) .

Base case: Recall that $\mathcal{F} = (\Gamma, \mathcal{E}')$ was constructed to build the matrix D . Assume that (p, q) is the closest pair of Γ . In this case there exists a well-separated pair $\{\mathcal{A}_j, \mathcal{B}_j\}$ such that $\mathcal{A}_j = \{p\}$ and $\mathcal{B}_j = \{q\}$ otherwise (p, q) could not be the closest pair. Hence the claim holds since there is an edge in \mathcal{F} of weight $B(a_j, b_j)$.

Induction hypothesis: Assume that the lemma holds for all pairs in Γ closer than $|p, q|$ to each other.

Induction step: If $(p, q) \notin \mathcal{F}$ then there exists an edge (x, y) in \mathcal{F} and a well-separated pair $\{\mathcal{A}_j, \mathcal{B}_j\}$ such that $x, p \in \mathcal{A}_j$ and $y, q \in \mathcal{B}_j$. According to the induction hypothesis there is path between p and x of weight $\frac{1+\tau_3}{1+\tau_2} \cdot D(p, q)$ and a path between y and q of weight $\frac{1+\tau_3}{1+\tau_2} \cdot D(p, q)$. Also, the weight of the edge (x, y) is $B(x, y)$. Putting together the weights we obtain that

$$\begin{aligned} \delta_{\mathcal{F}}(p, q) &\leq \frac{1+\tau_3}{1+\tau_2} B(p, x) + B(x, y) + \frac{1+\tau_3}{1+\tau_2} B(y, q) \\ &\leq \left(2 \frac{1+\tau_3}{1+\tau_2} (2/s) + (1+4/s) \right) \cdot B(p, q) \\ &= \frac{1+\tau_3}{1+\tau_2} B(p, q) = (1+\tau_3) \cdot \delta_{\mathcal{G}_i}(p, q) \end{aligned}$$

In the third step we used the fact that $s = \frac{1+\tau_2+\tau_3}{\tau_3-\tau_2}$. \square

Corollary 1 *Let p and q be any representative points in \mathcal{V} it holds that $D(p, q) \leq (1+\tau_3) \cdot \delta_{\mathcal{G}}(p, q)$.*

PROOF. Since all inter-connecting edges in \mathcal{G} also is in \mathcal{F} we can apply Lemma 6 to obtain the corollary. \square

Lemma 7 *Given a pair of points $p, q \in \mathcal{V}$ it holds that*

$$\delta_{\mathcal{G}}(p, r(p)) + \delta_{\mathcal{G}}(r(p), r(q)) + \delta_{\mathcal{G}}(r(q), q) \leq (1+\tau_5) \cdot \delta_{\mathcal{G}}(p, q).$$

PROOF. Let $h(p)$ and $h(q)$ denote the points in \mathcal{C} that is first encountered when following the path $\delta_{\mathcal{G}}(p, q)$ from p to q and from q to p respectively. According to Theorem 4 there are four cases to consider.

$$(1) \quad |p, r(p)| \leq \tau_1 \cdot |p, h(p)| \text{ and } |q, r(q)| \leq \tau_1 \cdot |q, h(q)|.$$

$$\begin{aligned} \delta_{\mathcal{G}}(p, q) &\leq \delta_{\mathcal{G}}(p, r(p)) + \delta_{\mathcal{G}}(r(p), r(q)) + \delta_{\mathcal{G}}(r(q), q) \\ &\leq \delta_{\mathcal{G}}(p, r(p)) + \delta_{\mathcal{G}}(r(p), p) + \delta_{\mathcal{G}}(p, h(p)) + \delta_{\mathcal{G}}(h(p), h(q)) \\ &\quad + \delta_{\mathcal{G}}(q, h(q)) + \delta_{\mathcal{G}}(q, r(q)) + \delta_{\mathcal{G}}(r(q), q) \\ &\leq (1+2t\tau_1) \delta_{\mathcal{G}}(p, h(p)) + \delta_{\mathcal{G}}(h(p), h(q)) + (1+2t\tau_1) \delta_{\mathcal{G}}(h(q), q) \\ &< (1+2t\tau_1) \cdot \delta_{\mathcal{G}}(p, q) \end{aligned}$$

(2) $|r(p), h(p)| \leq \tau_1 \cdot |p, h(p)|$ and $|r(q), h(q)| \leq \tau_1 \cdot |q, h(q)|$.

$$\begin{aligned}
\delta_{\mathcal{G}}(p, q) &\leq \delta_{\mathcal{G}}(p, r(p)) + \delta_{\mathcal{G}}(r(p), r(q)) + \delta_{\mathcal{G}}(r(q), q) \\
&\leq \delta_{\mathcal{G}}(p, h(p)) + \delta_{\mathcal{G}}(h(p), r(p)) + \delta_{\mathcal{G}}(r(p), h(p)) + \delta_{\mathcal{G}}(h(p), h(q)) \\
&\quad + \delta_{\mathcal{G}}(h(q), r(q)) + \delta_{\mathcal{G}}(r(q), h(q)) + \delta_{\mathcal{G}}(h(q), q) \\
&\leq (1 + 2t\tau_1)\delta_{\mathcal{G}}(p, h(p)) + \delta_{\mathcal{G}}(h(p), h(q)) + (1 + 2t\tau_1)\delta_{\mathcal{G}}(h(q), q) \\
&< (1 + 2t\tau_1) \cdot \delta_{\mathcal{G}}(p, q)
\end{aligned}$$

(3) $|p, r(p)| \leq \tau_1 \cdot |p, h(p)|$ and $|r(q), h(q)| \leq \tau_1 \cdot |q, h(q)|$.

$$\begin{aligned}
\delta_{\mathcal{G}}(p, q) &\leq \delta_{\mathcal{G}}(p, r(p)) + \delta_{\mathcal{G}}(r(p), r(q)) + \delta_{\mathcal{G}}(r(q), q) \\
&\leq \delta_{\mathcal{G}}(p, r(p)) + \delta_{\mathcal{G}}(r(p), p) + \delta_{\mathcal{G}}(p, h(p)) + \delta_{\mathcal{G}}(h(p), h(q)) \\
&\quad + \delta_{\mathcal{G}}(h(q), r(q)) + \delta_{\mathcal{G}}(r(q), h(q)) + \delta_{\mathcal{G}}(h(q), q) \\
&\leq (1 + 2t\tau_1)\delta_{\mathcal{G}}(p, h(p)) + \delta_{\mathcal{G}}(h(p), h(q)) + (1 + 2t\tau_1)\delta_{\mathcal{G}}(h(q), q) \\
&< (1 + 2t\tau_1) \cdot \delta_{\mathcal{G}}(p, q)
\end{aligned}$$

(4) $|p, r(p)| \leq \tau_1 \cdot |p, h(p)|$ and $|r(q), h(q)| \leq \tau_1 \cdot |q, h(q)|$. See case 3.

The lemma follows by setting $\tau_5 = 2t\tau_1$. \square

Lemma 8 *Let p and q be any points in \mathcal{V} it holds that*

$$\delta_{\mathcal{G}}(p, q) \leq B(p, r(p)) + D(r(p), r(q)) + B(r(q), q) \leq (1 + \varepsilon) \cdot \delta_{\mathcal{G}}(p, q).$$

PROOF.

$$\begin{aligned}
\delta_{\mathcal{G}}(p, q) &\leq B(p, r(p)) + D(r(p), r(q)) + B(r(q), q) \\
&\leq (1 + \tau_2) \cdot \delta_{\mathcal{G}_i}(p, r(p)) + (1 + \tau_3) \cdot \delta_{\mathcal{G}}(r(p), r(q)) \\
&\quad + (1 + \tau_2) \cdot \delta_{\mathcal{G}_j}(r(q), q) \\
&\leq (1 + \tau_2)(1 + \tau_4) \cdot \delta_{\mathcal{G}}(p, r(p)) + (1 + \tau_3) \cdot \delta_{\mathcal{G}}(r(p), r(q)) \\
&\quad + (1 + \tau_2)(1 + \tau_4) \cdot \delta_{\mathcal{G}}(r(q), q) \\
&< (1 + \tau_4)(1 + \tau_3) \cdot (\delta_{\mathcal{G}}(p, r(p)) + \delta_{\mathcal{G}}(r(p), r(q)) + \delta_{\mathcal{G}}(r(q), q)) \\
&\leq (1 + \tau_4)(1 + \tau_3)(1 + \tau_5) \cdot \delta_{\mathcal{G}}(p, q) \\
&= (1 + \varepsilon) \cdot \delta_{\mathcal{G}}(p, q)
\end{aligned}$$

On line 1 we used Observation 3 together with Lemma 6. On the following line we used Observation 4, applied Lemma 7 and finally replaced $(1 + \tau_2)(1 + \tau_3)(1 + \tau_5)$ with $(1 + \varepsilon)$. \square

Putting together Lemma 5 and Lemma 8 gives us Theorem 1.

5 Refinements and extensions

Below is listed a number of refinements and extensions:

- (1) A refined analysis yields that the data structure of Theorem 1 only uses $\mathcal{O}(|\mathcal{C}|^2 + n \log n)$ space, where \mathcal{C} is the set of all airports.
- (2) The data structure can be modified to handle the case when each island \mathcal{G}_i is a t_i -spanner, i.e., every island has different (although constant) dilation.
- (3) If we allow $(2k - 1)$ -approximations, where k is a positive integer, we can construct a data structure in $\mathcal{O}(n \log n + kM|C|^{1/k})$ time using $\mathcal{O}(n \log n + k|C|^{1+1/k})$ space, by replacing the usage of Matrix D with the method for general graphs used by Thorup and Zwick [25].

References

- [1] D. Aingworth, C. Chekuri, P. Indyk, and R. Motwani. Fast estimation of diameter and shortest paths (without matrix multiplication). *SIAM Journal on Computing*, 28(4):1167–1181, 1999.
- [2] S. Arikati, D. Z. Chen, L. P. Chew, G. Das, M. Smid, and C. D. Zaroliagis. Planar spanners and approximate shortest path queries among obstacles in the plane. In Proc. *4th European Symposium on Algorithms*, pp. 514-528, 1996.
- [3] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching. *Journal of the ACM*, 45(6):891-923, 1998.
- [4] S. Baswana and S. Sen. Approximate Distance Oracles for Unweighted Graphs in $\mathcal{O}(n^2 \log n)$ Time. In Proc. *15th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 271-280, 2004.
- [5] P. B. Callahan and S. R. Kosaraju. A decomposition of multidimensional point sets with applications to k -nearest-neighbors and n -body potential fields. *Journal of the ACM*, 42(1):67–90, 1995.
- [6] D. Z. Chen. On the all-pairs Euclidean short path problem. In Proc. *6th ACM-SIAM Symposium on Discrete Algorithms*, pp. 292-301, 1995.
- [7] D. Z. Chen, K. S. Klenk, and H.-Y. T. Tu. Shortest path queries among weighted obstacles in the rectilinear plane. *SIAM Journal on Computing*, 29(4):1223–1246, 2000.
- [8] Y.-J. Chiang and J. S. B. Mitchell. Two-point Euclidean shortest path queries in the plane. In Proc. *10th ACM-SIAM Symposium on Discrete Algorithms*, pp. 215-224, 1999.

- [9] E. Cohen. Fast algorithms for constructing t -spanners and paths with stretch t . *SIAM Journal on Computing*, 28(1):210–236, 1998.
- [10] E. W. Dijkstra. A Note on Two Problems in Connexion with Graphs. In *Numerische Mathematik* vol. 1, 1959.
- [11] D. Dor, S. Halperin, and U. Zwick. All-pairs almost shortest paths. *SIAM Journal on Computing*, 29(5):1740–1759, 2000.
- [12] D. Eppstein and D. Hart. An Efficient Algorithm for Shortest Paths in Vertical and Horizontal Segments. In *Proc. 5th Workshop on Algorithms and Data Structures*, pp. 234-247, 1997.
- [13] D. Eppstein and D. Hart. Shortest Paths in an Arrangement with k Line Orientations. In *Proc. 10th ACM-SIAM Symposium on Discrete Algorithms*, pp. 310-316, 1999.
- [14] M. L. Fredman, R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34(3):596-615, 1987.
- [15] J. Gudmundsson, C. Levcopoulos, G. Narasimhan and M. Smid. Approximate Distance Oracles for Geometric graphs. In *Proc. 13th ACM-SIAM Symposium on Discrete Algorithms*, pp. 828-837, 2002.
- [16] J. Gudmundsson, C. Levcopoulos, G. Narasimhan and M. Smid. Approximate Distance Oracles Revisited. In *Proc. 13th International Symposium on Algorithms and Computation*, pp. 357-368, 2002
- [17] J. Gudmundsson, G. Narasimhan and M. Smid. Fast Pruning of Geometric Spanners. In *Proc. 22nd International Symposium on Theoretical Aspects of Computer Science*, pp. 508-520, 2005.
- [18] D. Krznicaric and C. Levcopoulos. Computing hierarchies of clusters from the Euclidean minimum spanning tree in linear time. In *Proc. 15th Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, pp. 443-455, 1995.
- [19] D. Krznicaric and C. Levcopoulos. Optimal algorithms for complete linkage clustering in d dimensions. *Theoretical Computer Science*, 286(1):139-149, 2002.
- [20] J. S. B. Mitchell. Shortest paths and networks. In *Handbook of Discrete and Computational Geometry*, pp. 445–466. CRC Press LLC, 1997.
- [21] R. Raman. Recent results on the single-source shortest paths problem. *SIGACT News* 28:81-87,1997.
- [22] J. A. Storer and J. H. Reif. Shortest Paths in the Plane with Polygonal Obstacles. *Journal of the ACM*, 41(5): 982-1012, 1994.
- [23] M. Thorup. Floats, Integers, and Single Source Shortest Paths. *Journal of Algorithms*, 35(2):189-201, 2000.
- [24] M. Thorup. Undirected Single-Source Shortest Paths with Positive Integer Weights in Linear Time. *Journal of the ACM*, 46(3):362-394, 1999.

- [25] M. Thorup and U. Zwick. Approximate distance oracles. In Proc. *33rd ACM Symposium on Theory of Computing*, pp. 183-192, 2001.