Approximate Distance Oracles for Graphs with Dense Clusters

Mattias Andersson [a], Joachim Gudmundsson [b,1], Christos Levcopoulos [a]

[a] *Department of Computer Science, Lund University, Box 118, 221 00 Lund, Sweden.*

[b] *Department of Mathematics and Computing Science, TU Eindhoven, 5600 MB, Eindhoven, the Netherlands.*

Abstract

Let $\mathcal{G}$ be a graph containing N disjoint t-spanners that are inter-connected with M edges. We present an algorithm that constructs 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 n is the number of vertices of $\mathcal{G}$.

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 [6,8,12–14].

Lately the approximation version of this problem has also been studied extensively [1,5,7]. 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 [15] presented an algorithm for undirected weighted graphs that computes approximate solutions using a pre-computed data structure (the time of pre-processing was recently improved by Baswana and Sen [3]). Since the query time is essentially bounded by a constant, Thorup and Zwick refer to their queries as approximate *distance oracles*.

We focus on the geometric version of this problem. A geometric graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ has vertices corresponding to points in $\mathbb{R}^d$ and edge weights from a Euclidean metric, and 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. Again considerable previous work exists on the shortest path and related problems for t-spanners. The geometric query version was recently studied by Gudmundsson et al. [9,10] 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 $\mathcal{O}(m \log n)$.

In this paper we extend this result to hold also for "islands" of t-spanners, i.e., a set of disjoint t-spanners $\mathcal{G}_1, \ldots, \mathcal{G}_N$ inter-connected by M edges. 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 $\mathcal{O}((m + M^2) \log n)$, hence for $M = \mathcal{O}(\sqrt{n})$ the bound is essentially the same as in [9] and [10].

We claim that this generalization is natural in many applications. Consider for example the railway network in Europe where each country has a railway network which usually is a t-spanner for some small value t. The railway networks of the countries are then sparsely connected. Typically the number of inter-connecting edges is very small compared to the total number of edges in the network, see Fig. 1.

In [9] 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 generalises

Email addresses: `mattias@cs.lth.se` (Mattias Andersson), `h.j.gudmundsson@tue.nl` (Joachim Gudmundsson), `christos@cs.lth.se` (Christos Levcopoulos).

[1] Supported by the Netherlands Organisation for Scientific Research (NWO)

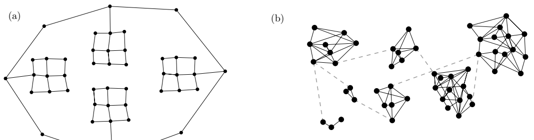

Fig. 1. (a) Many geometric networks consists of a set of "dense" graphs that are sparsely connected.(b) Example of an instance where the dashed edges are inter-connecting edges.

the results for the above mentioned problems.

We will use the following notation. For points p and q in $\mathcal{R}^d$, $|pq|$ 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 *Let $\mathcal{G}$ be a geometric graph, with n vertices and m edges, consisting of a set of disjoint t-spanners $\mathcal{G}_1=(\mathcal{V}_1,\mathcal{E}_1),\ldots,\mathcal{G}_N=(\mathcal{V}_N,\mathcal{E}_N)$ inter-connected by M edges, and let ε be a positive constant. One can construct a data structure in time $\mathcal{O}((m+M^2)\log n)$ using $\mathcal{O}(M^2+n\log n)$ space that can answer $(1+\varepsilon)$-approximate shortest path queries in constant time.*

The set of pairwise disjoint t-spanners $\mathcal{G}_1 = (\mathcal{V}_1,\mathcal{E}_1),\ldots,\mathcal{G}_N = (\mathcal{V}_N,\mathcal{E}_N)$ will be called the "islands" of $\mathcal{G}$ and, an edge $(u,v) \in \mathcal{E}$ is said to be an *inter-connecting* edge if $u \in \mathcal{V}_i$ and $v \in \mathcal{V}_j$, where $i \neq j$. A vertex $v \in \mathcal{V}_i$ incident to an inter-connecting edge is called an *harbor* and, the set of all harbors of $\mathcal{V}_i$ is denoted $\mathcal{C}_i$. Note that the total number of harbors is $\mathcal{O}(M)$ since the number of inter-connecting edges is M.

2. Tools

In the construction of the distance oracle we will need several tools, among them the well-separated pair decomposition (WSPD) by Callahan and Kosaraju [4], a graph pruning tool by Gudmundsson et al. [9] and well-separated clusters by Krznaric and Levcopoulos [11].

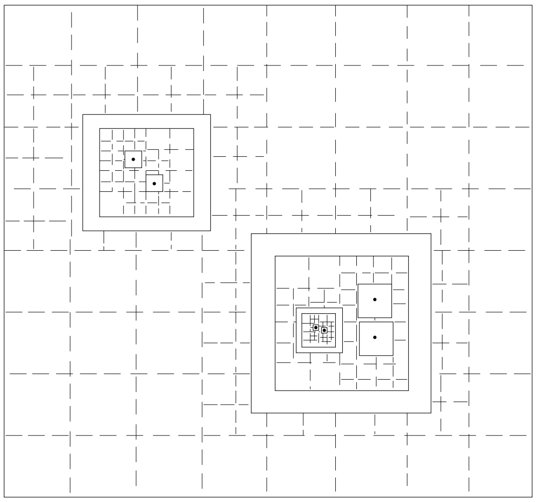

Fig. 2. An example cell partition, with respect to $\mathcal{V}'$, made by the algorithm. Doughnuts are drawn with solid lines, while inner cells are drawn with dotted ones.

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 $|pr| + |rq|$, for any point $q \in \mathcal{V}'$, is a good approximation of the distance $|pq|$. The total number of representative points is $\mathcal{O}(|V'|)$. 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 fact that, given a set $\mathcal{S}$ of n points in $\mathbb{R}^d$, an approximate nearest neighbor data structure can be efficiently computed (Mount et al. [2]).

Next the algorithm for computing representative points is presented. As a pre-processing step we compute the b-cluster tree $\mathcal{T}$ ([11]) of $\mathcal{V}'$ with $b = 10/\varepsilon^2$. For a level i in $\mathcal{T}$ let $\nu(\mathcal{D}_1),\ldots,\nu(\mathcal{D}_{\ell_i})$ be the nodes at that level, where $\mathcal{D}_1,\ldots,\mathcal{D}_{\ell_i}$ are the associated clusters. Let $\mathcal{D}_{j,1}\ldots,\mathcal{D}_{j,\ell_{i+1}}$ be the cluster associated with the children of $\nu(\mathcal{D}_j)$. 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 $\mathcal{T}$.

(i) Compute an approximate nearest neighbor structure with $\mathcal{D}(i)$ as input, as described by Mount et al. [2].

(ii) 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.

(iii) 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 $2\alpha = 2(1+1/\varepsilon)\cdot rd(cl(\mathcal{D}_j))$ and $2\beta = 2\varepsilon|d_j, v_j|/(1+\varepsilon)$ 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 $\mathcal{D}_j$.

(iv) 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 $\bigcup_{1\leq k\leq \ell_{i+1}} os(\mathcal{D}_{j,k})$ (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 $\mathcal{D}_j$. An illustration of the partition is shown in Fig. 2.

Finally, after all levels of $\mathcal{T}$ has been processed, we assign a representative point 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 $rep(C)$ is set to p. If p is not the first point then $rep(p) = rep(C)$. Further, note that an inner cell may overlap with the union of the outer shells of the children of $v(\mathcal{D}_j)$. 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.

For the above algorithm we can show the following theorem:

Theorem 2 *Given a set $\mathcal{V}$ of n points in $\mathbb{R}^d$, a subset $\mathcal{V}'\subseteq\mathcal{V}$ and a positive real value $\tau_1 < 1$, one can for each point $p\in\mathcal{V}$ associate a representative point $r(p)$ such that for any point $h\in\mathcal{V}'$, it holds that*

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

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

3. Constructing the Oracle

In this section we consider the main result of the paper, Theorem 1. The section is divided into two subsections: first we present the construction of the structure and then how queries are answered. Note that the correctness analysis has been omitted.

3.1. *Constructing the basic structures*

In this section we show how to pre-process $\mathcal{G}$ in time $\mathcal{O}((M^2+m)\log n)$ 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. This is done in time $\mathcal{O}(m\log n)$ [9]. 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 2 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 two points p and q answers 'yes' if p and q belongs to the same island, otherwise it will return the representative points (to be defined below) for p and q.

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.

The construction of Oracles A and B are rather straightforward, with construction details omitted. However, Oracle A can be constructed in linear time, using linear space, and Oracle B can be constructed in $\mathcal{O}(m\log n)$ time, using $\mathcal{O}(n\log n)$ space.

Matrix $\mathcal{D}$ is constructed as follows. For each i, $1\leq i\leq N$, compute the WSPD of Γ_i with separation constant $s = (\frac{1+\tau_2+\tau_3}{\tau_3-\tau_2})$ (the constants τ_2 and τ_3 are necessary for the correctness analy-

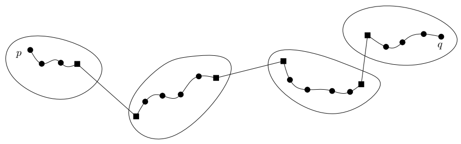

Fig. 3. Illustrating the approximate shortest path between p and q. The boxes illustrate the "harbors" along the path.

sis). As output we obtain a set of well-separated pairs $\{\{(A_1, B_1\}, \ldots, \{A_{w_i}, B_{w_i}\}\}$, 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 $\{A_j, B_j\}$ of the WSPD of Γ_i select two (arbitrary) representative points $a_j \in A_j$ and $b_j \in B_j$. 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 3 *The oracles A and B, and the matrix D can be built in time $\mathcal{O}((M^2 + m) \log n)$ and the total complexity of A, B and M is $\mathcal{O}(M^2 + n \log n)$.*

3.2. *Answer a query*

Given the two oracles and the matrices presented above the query algorithm is very simple. 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.

Using Lemma 3 and analysing the correctness of the query algorithm above, we can finally show Theorem 1.

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. 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.

[3] 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*, 2004.

[4] 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.

[5] E. Cohen. Fast algorithms for constructing t-spanners and paths with stretch t. *SIAM Journal on Computing*, 28(1):210–236, 1998.

[6] E. W. Dijkstra. A Note on Two Problems in Connexion with Graphs. In Numerische Mathmatik vol. 1, 1959.

[7] D. Dor, S. Halperin, and U. Zwick. All-pairs almost shortest paths. *SIAM Journal on Computing*, 29(5):1740–1759, 2000.

[8] 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.

[9] J. Gudmundsson, C. Levcopoulos, G. Narasimhan and M. Smid. Approximate Distance Oracles for Geometric graphs. In Proc. *13th ACM-SIAM Symposium on Discrete Algorithms*, 2002.

[10] J. Gudmundsson, C. Levcopoulos, G. Narasimhan and M. Smid. Approximate Distance Oracles Revisited. In Proc. *13th International Symposium on Algorithms and Computation*, 2002.

[11] D. Krznaric 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*, 1995.

[12] R. Raman. Recent results on the single-source shortest paths problem. SIGACT News 28:81-87,1997.

[13] M. Thorup. Floats, Integers, and Single Source Shortest Paths. *Journal of Algorithms*, 35(2):189-201, 2000.

[14] M. Thorup Undirected Single-Source Shortest Paths with Positive Integer Weights in Linear Time. *Journal of the ACM*, 46(3):362-394, 1999.

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