Approximate Distance Queries in Disk Graphs

Martin Fürer Shiva Prasad Kasiviswanathan

Computer Science and Engineering, Pennsylvania State University
e-mail: {furer,kasivisw}@cse.psu.edu

Abstract

We present efficient algorithms for approximately answering distance queries in disk graphs. Let G be a disk graph with n vertices and m edges. For any fixed $\epsilon > 0$, we show that G can be preprocessed in $O(m\sqrt{n}\epsilon^{-1} + m\epsilon^{-2}\log S)$ time, constructing a data structure of size $O(n^{3/2}\epsilon^{-1} + n\epsilon^{-2}\log S)$, such that any subsequent distance query can be answered approximately in $O(\sqrt{n}\epsilon^{-1} + \epsilon^{-2}\log S)$ time. Here S is the ratio between the largest and smallest radius. The estimate produced is within an additive error which is only ϵ times the longest edge on some shortest path.

The algorithm uses an efficient subdivision of the plane to construct a sparse graph having many of the same distance properties as the input disk graph. Additionally, the sparse graph has a small separator decomposition, which is then used to answer distance queries. The algorithm extends naturally to the higher dimensional ball graphs.

1 Introduction

In this paper we consider the problem of preprocessing a graph such that subsequent distance queries can be answered quickly within a small error. This natural extension to the all pairs shortest path problem captures practical situations, where more often than not, we are interested in estimating the distance between two vertices quickly and accurately. In this framework, the goodness of an algorithm is typically measured in terms of the preprocessing time, query time, space complexity and approximation factor (if any).

A disk graph is an intersection graph of disks in the plane. We consider weighted disk graphs where the weight of an edge is the Euclidean distance between centers. We present a new method for answering distance queries in disk graphs within an additive error which is only ϵ times the longest edge on some shortest path. The results are also extended to their higher dimensional versions, the ball graphs. The *difficulty* in answering queries for the disk graph metric when compared to the metric induced by a complete Euclidean graph (where it is trivial) is that two points that are spatially close are not necessarily close under the graph metric.

The algorithm uses a hierarchical subdivision of the plane into tiles of different sizes to replace the (possibly dense) disk graph by a sparse graph having many of the same distance properties. The sparse graph has a small geometric separator decomposition, which is then exploited for answering distance queries. The input graph might have no small separator, it could even be complete.

Disk graphs have been used widely to model the communication between objects in VLSI [9] and recently in the context of wireless ad-hoc networks [6, 7]. For wireless networks they model the fact that two wireless nodes can directly communicate with each other only if they are within a certain distance.

Distance queries are important in disk graphs as they are widely used to determine coverage in wireless sensor networks, and for routing protocols [5, 8, 12]. In most potential applications

(like military) one would not only desire high accuracy of these estimates but also the actual path producing this estimate. Our approximation algorithms are designed keeping this in mind.

1.1 Related Work

Let $G = (V, E)$ be a weighted graph, and let $d_G(u, v)$ denote the length of a shortest path between vertices u and v in G. An estimate $\delta(u, v)$ of the path length $d_G(u, v)$ is said to be a c-stretch if it satisfies $d_G(u, v) \leq \delta(u, v) \leq c d_G(u, v)$. For general undirected graphs, Thorup and Zwick [13] show that for any $c \geq 1$, a graph with n vertices and m edges can be preprocessed in $O(cmn^{1/c})$ expected time, constructing a data structure of size $O(cn^{1+1/c})$, such that a $(2c-1)$-stretch answer to any distance query can be produced in $O(c)$ time. Many other time-space trade off results are also known (See Zwick's [14] survey on this subject).

For unit disk graphs, Gao and Zhang [5] gave a construction of a c-well-separated pair decomposition (introduced by Callahan and Kosaraju in [3]) with $O(n \log n)$ pairs for any constant $c \geq 1$. Using the well-separated pair decomposition and $O(n\sqrt{n \log n}\epsilon^{-3})$ time preprocessing, they show that an $(1 + \epsilon)$-stretch answer to any distance query can be produced in $O(1)$ time. They also show that for unit ball graphs in $\mathbb{R}^k$ at least $\Omega(n^{2-2/k})$ pairs are needed for the well-separated pair decomposition. However, one cannot hope to extend these results to general disks graphs, as general disk graphs do not have a sub-quadratic well-separated pair-decomposition. One such example is the star graph, formed by a big disk and $n-1$ pairwise disjoint small disks intersecting the big disk.

2 Preliminaries

Let $\mathcal{P}$ be a set of points in $\mathbb{R}^k$ for any fixed dimension k. Let $\mathcal{D}$ be a set of n balls such that (i) $D_u \in \mathcal{D}$ is centered at $u \in \mathcal{P}$, and (ii) D_u has radius of r_u. Balls D_u and D_v intersect if $d(u, v) \leq (r_u + r_v)$, where $d(.,.)$ denotes the Euclidean metric. The ball graph G is a weighted graph where an edge between u and v with weight $d(u, v)$ exists if D_u and D_v intersect. Let d_G denote the metric induced by the connected graph G on its vertices by shortest paths.

We use m to denote the number of edges in graph G. We require that the input to the algorithms is the set of balls, not only the corresponding intersection graph. We re-scale to ensure that in $\mathcal{D}$, for every ball D_u there exists at least one ball center outside D_u. We then re-scale the balls such that the largest radius equals one. The global scale factor (ratio between largest and smallest radius) of $\mathcal{D}$ is then defined as

$$\rho(\mathcal{D}) = 1/\min\{r_u \mid D_u \in \mathcal{D}\}.$$

For disk graphs our algorithms use a variant of quadtrees. For a node s, denote by $P(s)$ the parent of s in the tree. We use d_s to denote the depth of node s in the tree. A point (x, y) is *contained* in a node s representing a square with center (x_s, y_s) and length l_s iff $x_s - l_s/2 \leq x < x_s + l_s/2$ and $y_s - l_s/2 \leq y < y_s + l_s/2$. For a set of squares $\mathcal{S}$ in the quadtree a point is contained in $\mathcal{S}$ iff there exists $s \in \mathcal{S}$, such that point is contained in s. For two squares s and s', the distance $\mathtt{dist}(s, s')$ is the Euclidean distance between their centers.

To avoid ambiguities, throughout the paper we refer to the vertices of a graph as *vertices* and vertices of a tree as *nodes*. We assume w.l.o.g. that ϵ^{-1} is a power of 2. Floors and ceilings are omitted throughout the paper, unless needed.

2.1 Separators and Separator Decomposition

A subset of vertices S of a graph G with n vertices is an $f(n)$-separator that α-splits ($\alpha < 1$) if $|S| \leq f(n)$ and the vertices of $G - S$ can be partitioned into two sets V_1 and V_2 such that there are no edges from V_1 to V_2, $\max\{|V_1|, |V_2|\} \leq \alpha n$, where f is a function. An $f(K)$-separator decomposition of G is a recursive decomposition of G using separators, where subgraphs of size K have separators of size $O(f(K))$.

We use a rooted binary tree T_G to represent a separator decomposition of a graph $G = (V, E)$. For a set V' of vertices in G, we use $Ne(V')$ to denote the neighborhood of V'. Each node $t \in T_G$ is labeled by two subsets of vertices $V(t) \subseteq V$ and $S(t) \subseteq V(t)$. Let $G(t) = (V(t), E(t))$ denote the subgraph induced by $V(t)$. Then $S(t)$ is the separator in $G(t)$. The root $r \in T_G$ has $V(r) = V$ and $S(r)$ is a separator in G. For any $t \in T_G$, the labels of its children t_0, t_1 are defined as follows: let $V_1 \subset V(t)$ and $V_2 \subset V(t)$ be the components separated by $S(t)$ in $G(t)$. Then $V(t_0) = V_1 \cup (S(t) \cap Ne(V_1))$, $V(t_1) = V_2 \cup (S(t) \cap Ne(V_2))$.

2.2 Our Contributions

In this paper we use a quadtree like partitioning scheme to construct a new sparse graph. Given an input ball (disk) graph $G = (V, E)$ a weighted graph $G' = (V', E')$ is constructed such that: (i) V' is a subset of V, (ii) every vertex in G is close to some vertex in G' under the d_G metric, and (iii) the distance between any two vertices present in G' is not much larger than the distance between them in G. We refer to the elements of V' as the *representative vertices*. We call G' the *cluster graph* of G.

For any fixed $\epsilon > 0$, an estimate $\delta(u, v)$ of the distance between u and v is said to be $(1+\epsilon)$-approximate if $d_G(u, v) \leq \delta(u, v) \leq d_G(u, v) + \epsilon d_G(u, v)$. We define a stronger notion called *strong* $(1+\epsilon)$*-approximation*. An $(1+\epsilon)$-approximate estimate $\delta(u, v)$ is said to be strong if $d_G(u, v) \leq \delta(u, v) \leq d_G(u, v) + \epsilon \ell(u, v)$, where

$$\ell(u, v) = \max\{\ell \mid \exists \text{ a shortest path in } G \text{ between } u \text{ and } v \text{ of edge length } \ell\}.$$

Let G be a ball graph defined on $\mathcal{D}$, we show that after $O(mn^{1-1/k}\epsilon^{-k+1} + m\epsilon^{-k}\log\rho(\mathcal{D}))$ time and $O(n^{2-1/k}\epsilon^{-k+1} + n\epsilon^{-k}\log\rho(\mathcal{D}))$ space preprocessing, a strong $(1+\epsilon)$-approximate estimate for the distance between any two vertices can be obtained in $O(n^{1-1/k}\epsilon^{-k+1} + \epsilon^{-k}\log\rho(\mathcal{D}))$ time. We can also output a corresponding short path between the query vertices in $O(L)$ time, where L is the number of edges of the reported path. In all our cases $\ell(u, v)$ is strictly less than $d_G(u, v)$. Therefore our approximation is strictly better than the standard $(1+\epsilon)$-approximation.

To illustrate the main ideas in our algorithm we start by considering the easier case where all the disks have *almost* the same radius, i.e., when every radius is $\Theta(1)$.

3 Distance Queries in Almost Unit Disk Graphs

We first describe the construction of the cluster graph G' and then the algorithm for finding a separator decomposition.

Imposing the Grid: The input to our algorithm is a set of disks in $\mathbb{R}^2$. Let $\mathcal{P}$ be the set of their centers. The *bounding box* of $\mathcal{P}$ is the smallest rectangle enclosing $\mathcal{P}$. We assume the left bottom corner as the origin. An ϵ-grid is defined by horizontal and vertical line segments drawn at $y \in \epsilon\mathbb{Z}$ and $x \in \epsilon\mathbb{Z}$ within the bounding box.

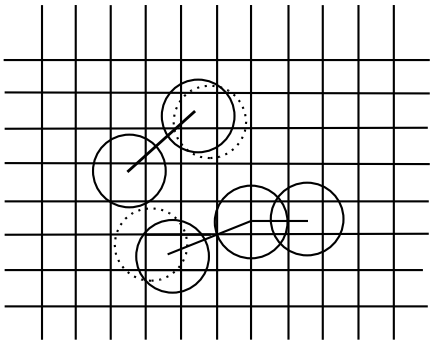


Figure 1: The grid with $\epsilon = 1/2$. The centers of solid disks are the representative vertices used in G'. The centers of dotted disks are ignored. The edges of G' are also shown.

Constructing G′: Let Roots represent the non-empty squares of the ϵ-grid. For every square $s \in \mathsf{Roots}$, the vertices contained in s form a clique in G. One such vertex is chosen as the representative, R_s. Define the *neighborhood* $N(s)$, as the set of all $s' \in \mathsf{Roots}$ which are within a distance of 2 from s. Note that the choice of distance 2 here is for the case where all the disks have the same radius. In general one can upper bound this distance as twice the radius of largest disk in $\mathcal{D}$. The vertices of G' are the representatives of the squares in Roots and the edge $(R_s, R_{s'})$ is added to G' if $s' \in N(s)$ (see Figure 1). For every square s, $|N(s)| = O(\epsilon^{-2})$. Since the graph G' has at most n vertices, it can be constructed in $O(n\epsilon^{-2})$ time.
Separator in G′: A $\sqrt{n}\epsilon^{-1}$-vertex separator in G' with 2/3-split can be found in $O(n \log n)$ by noting that G' is an $O(\epsilon^{-1})$-overlap graph as defined by Miller *et al.* [10] and by using their results for geometric separators on these graphs. In Appendix A, we provide a simpler algorithm having the same running time, but guaranteeing a superior split ratio.

Theorem 1. *Let G be an (almost) unit disk graph and G' be the cluster graph constructed from G as described above. An $O(\sqrt{n}\epsilon^{-1})$-separator decomposition with $1/2$-split of G' can be found in $O(n \log n)$ time.*

Strong $(1+\epsilon)$-approximate answers to distance queries: The query algorithm is similar to the one used by Arikati *et al.* [1]. We discuss the procedure for a single node $t \in T_{G'}$. The preprocessing phase involves the following steps: (i) Compute $T_{G'}$ the separator decomposition tree for G'. (ii) Let $H(t)$ denote the graph induced by the set of vertices

$$\{v \mid \exists s \in \mathsf{Roots}\, \exists u \in V(t) \text{ such that } u = R_s \text{ and } v \text{ is contained in } s\}$$

on G. Intuitively the graph $H(t)$ is the induced graph of vertices belonging to either $V(t)$ or contained in the same square as a vertex in $V(t)$. From each node in $S(t)$ do a single source shortest path (SSSP) computation on $H(t)$.

The query procedure for finding an approximately shortest path between the vertices u and v of G consists of the following steps: (i) If there is an edge between u and v, set $\delta(u,v)$ to $d(u,v)$. (ii) Otherwise, (a) Initialize $\delta(u,v)$ to ∞. (b) Compute $s(u)$ and $s(v)$ as nodes in Roots with u contained in $s(u)$ and v in $s(v)$. (c) Find the least common ancestor of $R_{s(u)}$ and $R_{s(v)}$ in $T_{G'}$, say t'. (d) Estimate $\delta(u,v)$ as $\min\{\delta(u,v), \min_{z \in S(t')}\{d_G(u,z) + d_G(z,v)\}\}$. (e) If t' is not the root of $T_{G'}$, set $t' = P(t')$ and repeat step (d).

Using the algorithm of Schieber and Vishkin [11], the least common ancestor queries can be answered in $O(1)$ time after linear time preprocessing. The proof of correctness of the algorithm follows as in [1] and is omitted in this extended abstract.

Theorem 2. *Let G be an (almost) unit disk graph with $m = \Omega(n \log n)$ edges. The graph G can be preprocessed in $O(m\sqrt{n}\epsilon^{-1})$ time, producing a data structure of size $O(n^{3/2}\epsilon^{-1})$, such that subsequent distance queries can be answered approximately, in $O(\sqrt{n}\epsilon^{-1})$ time. The outputs produced are strong $(1+\epsilon)$-approximate distance estimates.*

Proof. If there is an edge between u and v in G, then the actual distance is the estimate. Otherwise, we know $d_G(u, v) > 2$. Consider a shortest path $P_s = (u_1, u_2, \ldots, u_k)$ between $u_1 = u$ and $u_k = v$ with no shortcuts, i.e., no edge from u_{j-1} to u_{j+1} for any j between 2 and $k-1$. Such a path P_s exists due to the triangle inequality. Since $d(u_{j-1}, u_{j+1}) > 2$, $max\{d(u_{j-1}, u_j), d(u_j, u_{j+1})\} \geq 1$. Let $(s(u_1), s(u_2), \ldots, s(u_k))$ be the sequence of squares in Roots such that $s(u_i)$ contains u_i. We know the path $(R_{s(u_1)}, \ldots, R_{s(u_k)})$ exists in G'. Let t be a minimal depth node in $T_{G'}$ where this path gets separated. This implies that among the vertices $\{R_{s(u_1)}, \ldots, R_{s(u_k)}\}$ at least one is in $S(t)$, say $R_{s(u_i)}$.

We compute single source shortest path from the vertex $R_{s(u_i)}$, which is at most $\sqrt{2}\epsilon$ distance away from u_i. There is also an edge between u_i and $R_{s(u_i)}$ in G. This proves that the estimate is only $O(\epsilon)$ greater than actual shortest path. It is also a strong $(1+\epsilon)$-approximate estimate, because there exists an edge in the path of P_s with length at least 1.

The running time of the preprocessing is dominated by the time for running Sssp from all the separator nodes. Using Dijkstra for Sssp gives a running time of $O(m\sqrt{n}\epsilon^{-1})$ for the preprocessing phase. The storage needed for all the results of Sssp computations is $O(n^{3/2}\epsilon^{-1})$. The query time is dominated by Steps (d) and (e) of the query algorithm and can be bounded by $O(\sqrt{n}\epsilon^{-1})$. ❑

4 Distance Queries in Ball Graphs

In this section we extend the results to arbitrary ball graphs. Again we describe the algorithm for the case of disk graphs and then state the extensions to higher dimensions. Let G denote the input disk graph. Each disk is associated with a level, a disk D_u is of level l if: $2^{-l} \leq r_u < 2^{-l+1}$. Let l_{min} denote the level of the smallest disk in $\mathcal{D}$, i.e., $l_{min} = \lceil \log \rho(\mathcal{D}) \rceil$. The level l_{min} is the deepest level in Γ.

Imposing the Grid: We impose an ϵ-grid as in the case of unit disk graphs. Additionally, we recursively subdivide each non empty square in the ϵ-grid using a simple variant of quadtrees. We view this subdivision as a 4-ary forest with the root nodes as the non-empty squares in the ϵ-grid. Each square is partitioned into four equal squares, which form its children. We continue partitioning the non-empty squares until the size of the squares becomes $\epsilon 2^{-l_{min}}$. This construction differs from the standard quadtrees in: (a) when we stop partitioning it is not necessary that each square contains only one point, and (b) unlike in quadtrees we don't stop the partitioning as soon as the square has only one point inside it. We call this procedure *dissection* of the ϵ-grid.

Constructing G′: Let Γ denote the forest from the dissection of the ϵ-grid. Let Roots initially be the collection of non-empty squares of the ϵ-grid. Γ is a collection of disjoint trees, each of which is rooted at a node of Roots. We introduce a disk of level l only at depth l in Γ. For a node $s \in \Gamma$, the set $C(s)$ only consists of disks which are of level d_s or less, and whose centers are contained in s. We also add to Roots any node $s \in \Gamma$ satisfying, $C(s) \neq \emptyset$ whereas $C(P(s)) = \emptyset$.

For a leaf node s in Γ we pick one of the vertices in $C(s)$ as its representative R_s. For an internal node s, we pick one of its children s' satisfying $R_{s'} \in C(s)$ and set R_s to $R_{s'}$. For every node $s \notin \mathsf{Roots}$, we define its neighborhood $N(s)$ as the set of all nodes at depth d_s which are within a distance of 2^{-d_s} from s. To define the neighborhood of a node $s \in \mathsf{Roots}$, we introduce some new

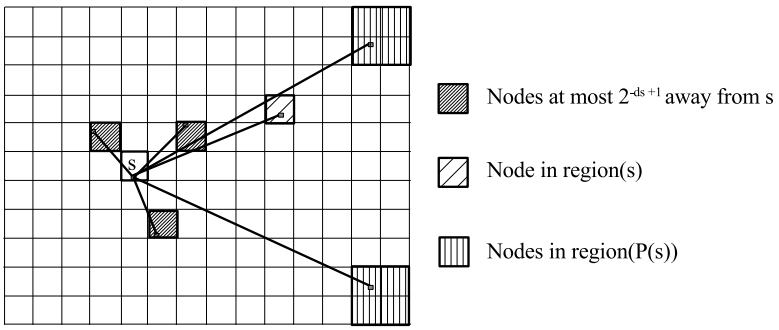


Figure 2: Neighborhood of a node s belonging to Roots.

definitions. For a node s, define

$$\begin{aligned} \mathsf{region}(s) &= \{s' \in \Gamma \mid \mathsf{dist}(s, s') \text{ lies in } [2^{-d_s+1}, 2^{-d_s+2}] \text{ and } d_s = d_{s'}\} \\ \mathsf{Region}(s) &= \bigcup_{s'=s \text{ or } s' \text{ is an ancestor of } s \text{ in } \Gamma} \mathsf{region}(s'). \end{aligned}$$

Empty squares can be ignored. The neighborhood of a node $s \in \mathsf{Roots}$ is now defined as

$$N(s) = \mathsf{Region}(s) \cup \{s' \in \Gamma \mid \text{ s' is at most } 2^{-d_s+1} \text{ from } s \text{ and } d_s = d_{s'}\}.$$

The idea behind creating the regions is to ensure that disks that intersect any disk centered in s, have their centers either close to s or inside a node of $\mathsf{Region}(s)$ (Lemma 1).

Lemma 1. *Let u be a disk center contained in node $s \in \mathsf{Roots}$. Then for every edge (u, v) in G, v is contained in $N(s)$.*

Proof. Trivially if $d(u, v) \leq 2^{-d_s+1}$ then $v \in N(s)$. If $2^{-d_s+1} < d(u, v) \leq 2^{-d_s+2}$ then v is contained in $\mathsf{region}(s)$. If $2^{-d_s+2} < d(u, v) \leq 2^{-d_s+3}$ then v is contained in $\mathsf{region}(P(s))$. Similarly for every increase in the distance by a factor of 2, we move one position up in Γ to finish the proof. ❑

The vertices of G' are the representatives of the nodes in Γ and the edge $(R_s, R_{s'})$ is added to G' if $s' \in N(s)$. The forest Γ can be constructed in $O(n \log \rho(\mathcal{D}))$ time. The graph G' can be constructed in $O(n\epsilon^{-2} \log \rho(\mathcal{D}))$ time.

Representative path of an edge: We now define a *representative path* in G' for every edge of G. For a vertex w in G, let $s(w)$ denote the deepest level node in Γ containing w. Consider an edge (u, v) of the graph G. In the rest of the discussion we assume w.l.o.g. that $r_u \leq r_v$. The representative path $P(u, v)$ starts at $R_{s(u)}$ and ends at $R_{s(v)}$. The following lemma is useful for defining the path.

Lemma 2. *Let u and v be two vertices in G' such that there exists a node $s \in \Gamma$ containing both u and v, with $v \in C(s)$. Then there exists a path from u to v in G'.*

Proof. The proof is by induction over the number of nodes contained in s. The base case is when s contains only one vertex. Now in the subtree of Γ rooted at s, consider the node s_1 at which u and v split, i.e., children of s_1 containing u and v are different. The splitting is guaranteed as u and v are representatives for different nodes at the deepest level in Γ. Let s_2 be the closest descendant of s_1 containing u with $C(s_2) \neq \emptyset$. Let s_3 be the child of s_1 containing v. In G' there exists an edge (R_{s_2}, R_{s_3}). By the inductive hypothesis we know there exists a path between u and R_{s_2} and between v and R_{s_3}. Thus in G' there exists a path $(u, \ldots, R_{s_2}, R_{s_3}, \ldots, v)$. ❑

Let a be the deepest node in $\mathtt{Roots}$ containing u. Let b be a node in Γ containing v with $d_a = d_b$. Since u and $R_{s(u)}$ (similarly v and $R_{s(v)}$) are always contained in the same node in Γ, we get that $R_{s(u)}$ is contained in a and $R_{s(v)}$ is contained in b. The representative path $P(u,v)$ for an edge (u,v) is defined using the following case distinction:
Case 1: If the distance between a and b is greater than 2^{-d_a+1}, then by Lemma 1, we know that there exists some $c \in \mathtt{Region}(a)$ containing v (and thus also $R_{s(v)}$). Since $\mathtt{dist}(a,c) \geq 2^{-d_c+1}$, we also know that $r_v \geq 2^{-d_c}$ and $v \in C(c)$. In G' there is an edge between R_a and R_c. From Lemma 2, we know there exists a path in G' between $R_{s(u)}$ and R_a and between $R_{s(v)}$ and R_c. Define the representative path $P(u,v)$ as $(R_{s(u)}, \ldots, R_a, R_c, \ldots, R_{s(v)})$.
Case 2: Otherwise, consider the deepest nodes f, g in Γ, such that (i) $R_{s(u)}$ is contained in f and $R_{s(v)}$ is contained in g, and (ii) there exists an edge (R_f, R_g) in G'. From Lemma 2, we know there exists a path between $R_{s(u)}$ to R_f and between $R_{s(v)}$ and R_g in G'. Define the representative path $P(u,v)$ as $(R_{s(u)}, \ldots, R_f, R_g, \ldots, R_{s(v)})$.

For every representative path, we also define a pair of nodes in Γ as its *covering nodes*. If $P(u,v)$ is defined using Case 1, then the nodes a and c are the covering nodes. If $P(u,v)$ is defined using Case 2, then f and g are the covering nodes. In both cases, all vertices in $P(u,v)$ are contained in one of the covering nodes with u and v contained in different covering nodes.

Lemma 3. *Let (u,v) be an edge in G of length greater than $2^{-l_{min}}$. Let $P(u,v)$ be its representative path in G' with p and q as the covering nodes. Then $\mathtt{dist}(p,q) \geq \max\{c_1 2^{-d_p}, c_1 2^{-d_q}\}$, for some constant c_1.*

Proof. We again use the same case distinction as in defining the path $P(u,v)$. Assume p contains u, and q contains v. If $P(u,v)$ is defined using Case 1, then $d_p \geq d_q$. Let r be the ancestor of p which is at the same depth as q. Since q is at least 2^{-d_q+1} away from r, we get that that $\mathtt{dist}(p,q) \geq c_1 2^{-d_q}$ (p is a square inside r).

If $P(u,v)$ is defined using Case 2, then $d_p = d_q$. Let p' be the child of p containing u. Let q' be child of q containing v. Since there is no edge between $R_{p'}$ and $R_{q'}$, we conclude $\mathtt{dist}(p',q') \geq 2^{-d_q-1}$. This implies that $\mathtt{dist}(p,q) \geq c_1 2^{-d_q}$. The existence of nodes p' and q' is guaranteed if $d(u,v) \geq (1+\epsilon/\sqrt{2})2^{-l_{min}}$. Otherwise, $\mathtt{dist}(p,q) = \Omega(2^{l_{min}})$ and again $\mathtt{dist}(p,q) \geq c_1 2^{-d_q}$. ❑

Separator in G′: We now show that a $\sqrt{n}\epsilon^{-1} + \epsilon^{-2}\log\rho(\mathcal{D})$-vertex separator in G' with 7/9-split[1] can be found in $O(n \log n)$. We use recursive partitions of rectangles. Let $\mathcal{D}(l)$ be the set of all vertices (disk centers) which were chosen as representatives at level l of Γ. For a rectangle $\mathcal{R}$, let $\mathcal{X}(\mathcal{R})$ be the sorted list of x-coordinates of vertices of G' which are contained in $\mathcal{R}$ (similarly define $\mathcal{Y}(\mathcal{R})$ for y-coordinates). We say a vertex crosses a given line if any edge incident on it in G' crosses the line.

At every step, the algorithm focuses on one rectangle, which we call the *active rectangle*. An active rectangle $\mathcal{R}$ has at least 2/3 of the vertices of G' inside it and there exists a set of $O(\sqrt{n}\epsilon^{-1} + \epsilon^{-2}\log\rho(\mathcal{D}))$ vertices which when removed ensures that no remaining vertex of G' has an edge that crosses the boundary of $\mathcal{R}$.

A *vertical double line separator* of an active rectangle is a set of at most two vertical line segments that partitions the active rectangle, such that there exists a set of $O(\sqrt{n}\epsilon^{-1} + \epsilon^{-2}\log\rho(\mathcal{D}))$ vertices which when removed ensures that no remaining vertex crosses the vertical line segments (similarly define *horizontal double line separator*). Our algorithm recursively partitions an active rectangle alternatively with a vertical or a horizontal double line separator and stops when none of the new rectangles created contains enough vertices to become active.

[1] A superior 2/3-split can be achieved by a more careful analysis.

Starting from the topmost level in Γ, we do the following step at every level l of Γ. The initial rectangle is the bounding box (and is active).

Constructing double line separators at level l: Let $\mathcal{R}$ be the currently active rectangle. Let $\mathcal{X}(\mathcal{R}) = \{x_1, x_2, \ldots\}$. We maintain sorted doubly linked lists for both the x- and y- coordinates with pointers between elements representing the same point in the two lists. Therefore in going to $\mathcal{R}$ from the previous active rectangle $\mathcal{R}_p$, the lists $\mathcal{X}(\mathcal{R})$ and $\mathcal{Y}(\mathcal{R})$ can be constructed in time proportional to the number of points removed from $\mathcal{R}_p$ to $\mathcal{R}$.

Let x_m denote the median of $\mathcal{X}(\mathcal{R})$. Starting at x_m we scan over the lower half list of $\mathcal{X}$ till we encounter the first x_l, such that $x_l - x_{l-\sqrt{n}\epsilon^{-1}} \geq (1+\epsilon/\sqrt{2})2^{-l+1}$. We define a vertical line segment (L_1) at $x = x_l$ with its end-points at boundaries of $\mathcal{R}$. Similarly starting from x_m, we scan over the upper half list of $\mathcal{X}(\mathcal{R})$ to find the first x_r such that $x_{r+\sqrt{n}\epsilon^{-1}} - x_r \geq (1+\epsilon/\sqrt{2})2^{-l+1}$. Again we define a similar vertical line segment (L_2) at $x = x_r$.

L_1 and L_2 divide $\mathcal{R}$ into at most 3 rectangles, of which only the rectangle ($\mathcal{R}_c$) between L_1 and L_2 could be active. If $\mathcal{R}_c$ is active, we repeat the same procedure with $\mathcal{Y}(\mathcal{R}_c)$ to find horizontal line segments L_3 and L_4. The active rectangle (if any) created is associated with the level $l+1$.

Lemma 4. *The distance between vertical line separators L_1 and L_2 defined at depth l is less than $(1+\epsilon/\sqrt{2})2^{-l+1}\sqrt{n}\epsilon$.*

Proof. We move in the lower (or upper list) only if there are $\sqrt{n}\epsilon^{-1}$ vertices to the left (or right) within a distance of $(1+\epsilon/\sqrt{2})2^{-l+1}$. Since there are only n disks, the result follows. ❑

Remark. The same distance bound holds also for the distance between L_3 and L_4. As a consequence of the above lemma we also get that the lengths of L_1 and L_2 for $l \geq 1$ is at most $(1+\epsilon/\sqrt{2})2^{-l+2}\sqrt{n}\epsilon$ (consider separation between line separators at level $l-1$). We use the upper bound of $2^{-l+3}\sqrt{n}\epsilon$ for the lengths of L_1 and L_2 in the proof.

Lemma 5. *Let $\mathcal{R}$ be an active rectangle at depth l. L_1 and L_2 define a vertical double line separator for $\mathcal{R}$.*

Proof. We work with L_1. The case for L_2 is similar. Every edge in G' is attributed to the disk of larger radius, i.e., an edge (u, v) in G' is attributed to u if $r_u \geq r_v$. First consider the edges attributed to vertices in $\bigcup_{l' \geq l} \mathcal{D}(l')$. These edges have length at most $(1+\epsilon/\sqrt{2})2^{-l+1}$. Any such edge crossing L_1 must have both its end points within a distance of $(1+\epsilon/\sqrt{2})2^{-l+1}$ from L_1. By construction we ensure that this number is $O(\sqrt{n}\epsilon^{-1})$. Therefore there exists a set of $O(\sqrt{n}\epsilon^{-1})$ vertices which when removed ensures that no edge of length less than or equal $(1+\epsilon/\sqrt{2})2^{-l+1}$ crosses L_1. So if $l = 0$ we are done.

We now consider edges attributed to vertices in $\bigcup_{l'<l} \mathcal{D}(l')$. Fix any level $l' < l$. From Lemma 4, and the remark above we know that the length of L_1 is less than $2^{-l+3}\sqrt{n}\epsilon$. Consider four lines, two vertical lines drawn at distance $2^{-l'+2}$ on both sides of L_1 and two horizontal lines drawn above and below L_1 at a distance of $2^{-l'+2}$ from the end points. The rectangular region $\mathcal{R}_{l'}$ formed by these four lines has an area of at most $2^{-l'+3}(2^{-l+3}\sqrt{n}\epsilon + 2^{-l'+3})$. Since each node (square) in level l' has an area of $(\epsilon 2^{-l'})^2$ and at most one vertex from each square can be in $\mathcal{D}(l')$, we get that the number of vertices of $\mathcal{D}(l')$ present in this area is at most $2^{-l+l'+6}\sqrt{n}\epsilon^{-1} + 128\epsilon^{-2}$.

Each edge attributed to vertices in $\mathcal{D}(l')$ and crossing L_1 should have both its end points in $\mathcal{R}_{l'}$. Summing over all $l' < l$ we get that the total number of vertices in $\bigcup_{l'<l} \mathcal{D}(l')$ that could have edges crossing L_1 is at most

$$\sum_{0 \leq l' < l} (2^{l'-l+6}\sqrt{n}\epsilon^{-1} + 128\epsilon^{-2}) = O(\sqrt{n}\epsilon^{-1} + \epsilon^{-2}l).$$

Since there are only $\log \rho(\mathcal{D})$ levels in Γ, we can upper bound l by $\log \rho(\mathcal{D})$. Therefore there exist $O(\sqrt{n}\epsilon^{-1} + \epsilon^{-2} \log \rho(\mathcal{D}))$ vertices such that every edge crossing L_l is incident on one of them. ❑

Final Shape of the Separator: We stop the algorithm when no rectangle is still active. Among the rectangles formed by partitioning the last active rectangle, the rectangle $\mathcal{R}_f$ with the largest number of disk centers forms one component of the separator. Since the last active rectangle had at least $2/3$ of the vertices of G' and it gets divided into at most 3 rectangles, $\mathcal{R}_f$ contains at least $2/9$ of the vertices of G'. The vertices which are outside of $\mathcal{R}_f$ and are connected to some vertices inside form the vertex separator. The proof of termination follows from Lemma 5, because when the algorithm considers level l_{min}, the horizontal line separator divides the currently active rectangle into exactly two non-active rectangles.

A naive implementation of this algorithm runs in $O(n \log n + n \log \rho(\mathcal{D}))$ time. Again we have that,

Theorem 3. *Let G be a disk graph on $\mathcal{D}$ and G' be the cluster graph constructed from G as described above. An $O(\sqrt{n}\epsilon^{-1} + \epsilon^{-2} \log \rho(\mathcal{D}))$-separator decomposition with $7/9$-splits of G' can be found in $O(n \log n)$ time.*

Strong $(1+\epsilon)$-approximate answer to distance queries: Once we have the separator in G' the preprocessing and query procedure for estimating the distance is the same as with unit disk graphs. The following theorem shows that we get a strongly $(1+\epsilon)$-approximation.

Theorem 4. *Let G be a disk graph on $\mathcal{D}$ with $m = \Omega(n \log n)$ edges. The graph G can be preprocessed in $O(m\sqrt{n}\epsilon^{-1} + m\epsilon^{-2} \log \rho(\mathcal{D}))$ time, producing a data structure of size $O(n^{3/2}\epsilon^{-1} + n\epsilon^{-2} \log \rho(\mathcal{D}))$, such that subsequent distance queries can be answered approximately, in $O(\sqrt{n}\epsilon^{-1} + \epsilon^{-2} \log \rho(\mathcal{D}))$ time. The outputs produced are strong $(1+\epsilon)$-approximate distance estimates.*

Proof. If there is an edge between u and v in G, then the actual distance is the estimate. As in Theorem 2, consider a shortest path $P_s = (u_1, u_2 \ldots, u_k)$ between $u_1 (= u)$ and $u_k (= v)$ with no edge shortcuts. Since there is always an edge if the distance between two vertices is less than $2^{-l_{min}+1}$, we get $d(u_{j-1}, u_{j+1}) \geq 2^{-l_{min}+1}$ and $max\{d(u_{j-1}, u_j), d(u_j, u_{j+1})\} \geq 2^{-l_{min}}$ for any j between 2 and $k-1$. Let $(s(u_1), s(u_2), \ldots, s(u_k))$ be the sequence of squares at the deepest level in Γ such that $s(u_i)$ contains u_i. Let t be the deepest node in $T_{G'}$ with $R_{s(u_1)}, R_{s(u_k)} \in V(t)$.

If any vertex $R_{s(u_i)}$ from $\{R_{s(u_1)}, \ldots, R_{s(u_k)}\}$ is in the separator $S(t)$, then we do a single source shortest path from $R_{s(u_i)}$ on $H(t)$. Now $R_{s(u_i)}$ is at most $\epsilon 2^{-l_{min}+1/2}$ distance away from u_i and there exists an edge between $R_{s(u_i)}$ and u_i in G. This proves the estimate produced is only $O(\epsilon)$ greater than the shortest path. The strong $(1+\epsilon)$-approximation follows as there is an edge in P_s which is of length at least $2^{-l_{min}}$.

Otherwise, consider two vertices $R_{s(u_a)}$ and $R_{s(u_b)}$ from $\{R_{s(u_1)}, \ldots, R_{s(u_k)}\}$ which are separated into different components by $S(t)$ and there exists an edge (u_a, u_b) in G (by the assumption about the path P_s we get $|a - b| = 1$). Since there is no edge between $R_{s(u_a)}$ and $R_{s(u_b)}$, we get that $d(u_a, u_b) \geq 2^{-l_{min}}$. Consider the representative path $P(u_a, u_b)$ in G'. There exists at least one vertex (say z) on the path $P(u_a, u_b)$ in the separator $S(t)$. Let p and q be the two covering nodes for $P(u_a, u_b)$. By definition all the vertices in $P(u_a, u_b)$ are contained in either p or q. Assume w.l.o.g. that z and u_a are contained in p.

In G there exists edges (z, R_p) and (R_p, u_a). Since z, R_p and u_a are all contained in p we get $d(z, R_p), d(R_p, u_a) \leq \epsilon 2^{-d_p+1/2}$. On the other hand, $dist(p, q) \geq max\{c_1 2^{-d_p}, c_1 2^{-d_q}\}$ by Lemma 3. This also implies that $d(u_a, u_b) \geq c_2 2^{-d_p}$ for some constant c_2. Putting all together we get $d(z, R_p) \leq c_3 \epsilon d(u_a, u_b)$ and $d(R_p, u_a) \leq c_3 \epsilon d(u_a, u_b)$ for some constant c_3, implying in G there is a path from z to u_a of length at most $2c_3\epsilon d(u_a, u_b)$. Therefore when we do the single source

shortest path computation from z on $H(t)$, the error we make in taking the detour can be bounded by some constant times $\epsilon d(u_a, u_b)$. Implying that our estimate is a strong $(1+\epsilon)$-approximation.

The running time analysis follows as in Theorem 2. ❑

4.1 Extension to Higher Dimensions

In $\mathbb{R}^k$, squares become k-dimensional hypercubes and the lines used in the separator algorithm become $(k-1)$-dimensional hyperplanes. Given a ball graph G in $\mathbb{R}^k$, the corresponding cluster graph G' can be constructed in $O(n\epsilon^{-k}\log\rho(\mathcal{D}))$ time. Using the same algorithm for answering distance queries we have the following.

Theorem 5. *Let G be a k-dimensional ball graph on $\mathcal{D}$ with $m = \Omega(n\log n)$ edges. The graph G can be preprocessed in $O(mn^{1-1/k}\epsilon^{-k+1} + m\epsilon^{-k}\log\rho(\mathcal{D}))$ time, producing a data structure of size $O(n^{2-1/k}\ \epsilon^{-k+1} + n\epsilon^{-k}\log\rho(\mathcal{D}))$, such that subsequent distance queries can be answered approximately, in $O(n^{1-1/k}\ \epsilon^{-k+1} + \epsilon^{-k}\log\rho(\mathcal{D}))$ time. The outputs produced are strong $(1+\epsilon)$-approximate distance estimates.*

5 Concluding Remarks

Our results are also applicable in the case where the disk graphs are directed (such models are used in wireless networks to capture radio interferences [2]). All the results presented in this paper can also be extended to cases when intersections are between squares, or regular polygons, or other disk-like objects, as well as their higher dimensional versions. This follows as our algorithms don't use any special properties of balls (or disks). However, the partitioning scheme only works if the objects have almost the same aspect ratio. An open problem is to extend the results to intersection graphs between objects not having this property.

Finally in a very recent result the authors have extended these techniques to obtain a faster algorithm for constructing a spanner of disk graphs with support for distance queries [4]. it is shown that a spanner of a general ball graph can be constructed in sub-quadratic time.

References

[1] Srinivasa R. Arikati, Danny Z. Chen, L. Paul Chew, Gautam Das, Michiel H. M. Smid, and Christos D. Zaroliagis, *Planar spanners and approximate shortest path queries among obstacles in the plane*, ESA '96, vol. 1136, Springer, 1996, pp. 514–528.

[2] Hari Balakrishnan, Christopher L. Barrett, V. S. Anil Kumar, Madhav V. Marathe, and Shripad Thite, *The distance-2 matching problem and its relationship to the mac-layer capacity of ad hoc wireless networks*, IEEE Journal on Selected Areas in Communications **22** (2004), 1069–1079.

[3] Paul B. Callahan and S. Rao Kosaraju, *A decomposition of multidimensional point sets with applications to K-nearest-neighbors and N-body potential fields*, Journal of ACM **42** (1995), no. 1, 67–90.

[4] Martin Fürer and Shiva Prasad Kasiviswanathan, *Spanners for geometric intersection graphs*, Available at: `http://www.cse.psu.edu/~kasivisw/research.html`.

[5] Jie Gao and Li Zhang, *Well-separated pair decomposition for the unit-disk graph metric and its applications*, SIAM Journal on Computing **35** (2005), no. 1, 151–169.

[6] S. O. Krumke, M. V. Marathe, and S. S. Ravi, *Models and approximation algorithms for channel assignment in radio networks*, Wireless Networks **7** (2001), no. 6, 575–584.

[7] Xiang-Yang Li, *Algorithmic, geometric and graphs issues in wireless networks*, Wireless Communications and Mobile Computing **3** (2003), no. 2, 119–140.

[8] Xiang-Yang Li, Peng-Jun Wan, and Ophir Frieder, *Coverage in wireless ad hoc sensor networks*, IEEE Transactions on Computers **52** (2003), no. 6, 753–763.

[9] Carver Mead and Lynn Conway, *Introduction to VLSI system*, Addison-Wesley, Reading, 1980.

[10] Gary L. Miller, Shang-Hua Teng, and Stephen A. Vavasis, *A unified geometric approach to graph separators*, FOCS '01, IEEE, 1991, pp. 538–547.

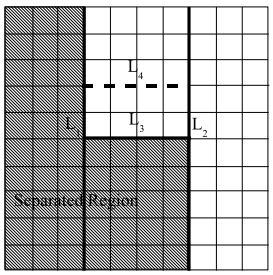
[11] Baruch Schieber and Uzi Vishkin, *On finding lowest common ancestors: Simplification and parallelization*, SIAM Journal on Computing **17** (1988), no. 6, 1253–1262.

[12] Anand Srinivas and Eytan Modiano, *Minimum energy disjoint path routing in wireless ad-hoc networks*, MOBICOM '03, ACM, 2003, pp. 122–133.

[13] Mikkel Thorup and Uri Zwick, *Approximate distance oracles*, Journal of ACM **52** (2005), no. 1, 1–24.

[14] Uri Zwick, *Exact and approximate distances in graphs - A survey*, ESA '01, vol. 2161, Springer, 2001, pp. 33–48.

Appendix A: Separators in Cluster Graph for the Unit Disk Case



We describe the algorithm for the case when all the disks have the same radius. It can easily be modified for the case where the disk radius differ only by a constant factor.

Let $\mathcal{X} = \{x_1, x_2, \ldots\}$ be the sorted list of x-coordinates of vertices in G'. Similarly define $\mathcal{Y}$ for y-coordinates. Let x_m denote the median of $\mathcal{X}$. Starting at x_m we scan over the lower half list of $\mathcal{X}$ till we encounter the first x_l, such that $x_l - x_{l-\sqrt{n}\epsilon^{-1}} \geq 2 + \sqrt{2}\epsilon$. Define a vertical line (L_1) at $x = x_l$. We add all the vertices to the left of x_l and crossing L_1 into the separator. Similarly starting from x_m, we scan over the upper half list of $\mathcal{X}$ to find the first x_r such that $x_{r+\sqrt{n}\epsilon^{-1}} - x_r \geq 2 + \sqrt{2}\epsilon$. Define a vertical line (L_2) at $x = x_r$. Again we add all the vertices to the right of x_r crossing L_2 into the separator. This entire procedure can be done in linear time if the coordinates are sorted.

Final shape of the separator: Let $\mathcal{P}_b$ be the set of all vertices of G' lying between $x = x_l$ and $x = x_r$. Let $\mathcal{Y}_b$ be the sorted list of y-coordinates of vertices in $\mathcal{P}_b$. Let $\mathcal{P}_l$ be the set of vertices to the left of L_1. Choose the element in $\mathcal{Y}_b$ with rank $|\mathcal{Y}|/2 - |\mathcal{P}_l|$, say y_d. Define the horizontal line segment (L_3) at $y = y_d$ with (x_l, y_d) and (x_r, y_d) as its endpoints. We add all the vertices crossing this horizontal line segment into the separator. After removing the separator, the union of vertices in $\mathcal{P}_l$ and the vertices in $\mathcal{P}_b$ whose y-coordinates are less than y_d forms one component.

Lemma 6. *The number of vertices of* G' *crossing the horizontal line segment* L_3 *is* $O(\sqrt{n}\epsilon^{-1})$.

Proof. Since the maximum edge length in G' is $2+\sqrt{2}\epsilon$, from the construction we know that there are at most $\sqrt{n}\epsilon^{-1}$ vertices to the left of L_1 crossing L_1 and potentially L_3 (similarly from the right of L_2). Consider the vertical strip between L_1 and L_2. By construction we also get that distance between L_1 and L_2 is $O(\sqrt{n}\epsilon)$ and hence the length of L_3 is $O(\sqrt{n}\epsilon)$. We now only consider the area within the strip.

Define another horizontal line segment L_4 at $y = y_d + 2 + \sqrt{2}\epsilon$ with x_l and x_r defining its endpoints (the case where L_4 is defined as $y = y_d - 2 - \sqrt{2}\epsilon$ is symmetric). Consider the rectangular region formed by L_1, L_2, L_3 and L_4. Any edge crossing the line segment L_3 would have an endpoint in this rectangular region. The area of this rectangular region is at most $O(\sqrt{n}\epsilon)$. Since there is

only one vertex of G' within each square of Roots, there exist at most $O(\sqrt{n}\epsilon^{-1})$ vertices within this rectangular region. Thus the total number of vertices of G' crossing L_3 is $O(\sqrt{n}\epsilon^{-1})$. ❑

The 1/2-split is guaranteed by the construction. The running time for finding the separator decomposition is dominated by the time for sorting.