Compact and Low Delay Routing Labeling Scheme for Unit Disk Graphs[⋆]

Chenyu Yan, Yang Xiang and Feodor F. Dragan[⋆⋆]

Algorithmic Research Laboratory, Department of Computer Science

Kent State University, Kent, Ohio, U.S.A.

{*cyan,yxiang,dragan*}*@cs.kent.edu*

[⋆] Part of these results will be presented at WADS 2009 Conference, 21-23 August 2009, Banff Conference Centre, Banff, Alberta, Canada.

[⋆⋆] Feodor F. Dragan is the corresponding author, Phone: +1-330-672-9058, Fax: +1-330-672-0737

Abstract. In this paper, we propose a *new compact and low delay routing labeling scheme* for *Unit Disk Graphs (UDGs)* which often model wireless ad hoc networks. We show that one can assign each vertex of an n-vertex UDG G a compact $O(\log^2 n)$-bit label such that, given the label of a source vertex and the label of a destination, it is possible to compute efficiently, based solely on these two labels, a neighbor of the source vertex that heads in the direction of the destination. We prove that this *routing labeling scheme* has a constant *hop route-stretch* (= *hop delay*), i.e., for each two vertices x and y of G, it produces a routing path with $h(x,y)$ hops (edges) such that $h(x,y) \leq 3 \cdot d_G(x,y) + 12$, where $d_G(x,y)$ is the hop distance between x and y in G. To the best of our knowledge, this is the first compact routing scheme for UDGs which not only guaranties delivery but has a low hop delay. Furthermore, our routing labeling scheme has a constant *length route-stretch* and a constant *power route-stretch*.

To obtain this result, we establish a novel *balanced separator* theorem for UDGs, which mimics the well-known Lipton and Tarjan's planar balanced shortest paths separator theorem. We prove that, in any n-vertex UDG G, one can find two hop-shortest paths $P(s,x)$ and $P(s,y)$ such that the removal of the 3-hop-neighborhood of these paths (i.e., $N_G^3[P(s,x) \cup P(s,y)]$) from G leaves no connected component with more than $2/3n$ vertices. This new *balanced shortest-paths—3-hop-neighborhood separator* theorem allows us to build, for any n-vertex UDG G, a system $\mathcal{T}(G)$ of at most $2\log_{\frac{3}{2}} n + 2$ spanning trees of G such that, for any two vertices x and y of G, there exists a tree T in $\mathcal{T}(G)$ with $d_T(x,y) \leq 3 \cdot d_G(x,y) + 12$. That is, the distances in any UDG can be approximately represented by the distances in at most $2\log_{\frac{3}{2}} n + 2$ of its spanning trees.

Keywords: unit disk graphs, collective tree spanners, routing and distance labeling schemes, balanced separators, efficient geometric graph algorithms.

1 Introduction

A common assumption for wireless ad hoc networks is that all nodes have the same maximum transmission range. By proper scaling, one can model these networks with *Unit Disk Graphs* (*UDGs*), which are defined as the intersection graphs of equal sized circles in the plane [3]. In other words, there is an edge between two vertices in an UDG if and only if their Euclidean distance is no more than one.

Communications in networks are performed using *routing schemes*, i.e., mechanisms that can deliver packets of information from any vertex of a network to any other vertex. In most strategies, each vertex v of a graph has full knowledge of its neighborhood and uses a piece of global information available to it about the graph topology – some "sense of direction" to each destination – stored locally at v. Based only on this information and the address of a destination vertex, vertex v needs to decide whether the packet has reached its destination, and if not, to which neighbor of v to forward the packet. The *efficiency* of a routing scheme is measured in terms of its *multiplicative route-stretch* (or *additive route-stretch*), namely, the maximum ratio (or surplus) between the cost (which could be the *hop-count*, the *length* or the *power-consumption*) of a route, produced by the scheme for a pair of vertices, and the cost of an optimal route available in graph for that pair. Here, the *hop-count* of a route is defined as the number of edges on it, the *length* of a route is defined as the sum of the Euclidean length of its edges, the *power-consumption* of a route is defined as the sum of the β-powers of the Euclidean length of its edges (for some $\beta \in [2, 5]$ depending on the routing environment). Using different cost functions, for a given graph G and a given routing scheme on G, one can define three different notions of route-stretch: *hop route-stretch*, *length route-stretch*, and *power route-stretch*.

The most popular strategy in wireless networks is the *geographic routing* (sometimes called also the *greedy geographic routing*), where each vertex forwards the packet to the neighbor geographically closest to the destination (see survey [12] for this and many other strategies). Each vertex of the network knows its position (e.g., Euclidean coordinates) in the underlying physical space and forwards messages according to the coordinates of the destination and the coordinates of neighbors. Although this greedy method is effective in many cases, packets may get routed to where no neighbor is closer to the destination than the current vertex. Many recovery schemes have been proposed to route around such voids for guaranteed packet delivery as long as a path exists [4, 14, 16]. These techniques typically exploit planar subgraphs (e.g., Gabriel graph, Relative Neighborhood graph), and packets traverse faces on such graphs using the well-known right-hand rule. Although these techniques guarantee packet delivery, none of them give any guaranties on how the routing path traveled is "close" to an optimal path; the worst-case route-stretch can be linear in the network size.

All earlier papers assumed that vertices are aware of their physical location, an assumption which is often violated in practice for various of reasons (see [7, 15, 24]). In addition, implementations of recovery schemes are either based on non-rigorous heuristics or on non-trivial planarization procedures. To overcome these shortcomings, recent papers [7, 15, 24] propose routing algorithms which assign virtual coordinates to vertices in a metric space X and forward messages using geographic routing in X. In [24], the metric space is the Euclidean plane, and virtual coordinates are assigned using a distributed version of Tutte's "rubber band" algorithm for finding convex embeddings of graphs. In [7], the graph is embedded in R^d for some value of d much smaller than the network size, by identifying d beacon vertices and representing each vertex by the vector of distances to those beacons. The distance function on R^d used in [7] is a modification of the ℓ_1 norm. Both [7] and [24] provide substantial experimental support for the efficacy of their proposed embedding techniques – both algorithms are successful in finding a route from the source to the destination more than 95% of the time – but neither of them has a provable guarantee. Unlike embeddings of [7] and [24], the embedding of [15] guarantees that the geographic routing will always be successful in finding a route to the destination, if such a route exists. Algorithm of [15] assigns to each vertex of the network a virtual coordinate in the hyperbolic plane, and performs greedy geographic routing with respect to these virtual coordinates. However, although the experimental results of [15] confirm that the greedy hyperbolic embedding yields routes with low route-stretch when applied to typical unit-disk graphs, the worst-case route-stretch is still linear in the network size.

In this paper, we propose a new compact and low delay routing labeling scheme for Unit Disk Graphs. We show that one can assign each vertex of an n-vertex UDG G a compact $O(\log^2 n)$-bit label such that, given the label of a source vertex and the label of a destination, it is possible to compute efficiently, based solely on these two labels, a neighbor of the source vertex that heads in the direction of the destination. We prove that this *routing labeling scheme* has a constant *hop route-stretch* (= *hop delay*), i.e., for each two vertices x and y of G, it produces a routing path with $h(x, y)$ hops such that $h(x, y) \leq 3 \cdot d_G(x, y) + 12$, where $d_G(x, y)$ is the hop distance between x and y in G. To the best of our knowledge, this is the first compact routing scheme for UDGs which not only guaranties delivery but has a low hop delay. Furthermore, our routing labeling scheme has a constant *length route-stretch* and a constant *power route-stretch*. Note also that, unlike geographic routing or any other strategies discussed in [4, 7, 12, 14–16, 24], our routing scheme is *degree-independent*. That is, each current vertex makes routing decision based only on its label and the label of destination, does not involve any labels of neighbors. The label assigned to a vertex in our scheme can be interpreted as its virtual coordinates. To assign those labels to vertices, we need to know only the topology of the input unit disk graph and relative Euclidean lengths of its edges.

To obtain our routing scheme, we establish a novel *balanced separator* theorem for UDGs, which mimics the well-known Lipton and Tarjan's planar balanced shortest paths separator theorem. We prove that, in any n-vertex UDG G, one can find two hop-shortest paths $P(s,x)$ and $P(s,y)$ such that the removal of the 3-hop-neighborhood of these paths (i.e., $N_G^3[P(s,x) \cup P(s,y)]$) from G leaves no connected component with more than $2/3n$ vertices. The famous Lipton and Tarjan's planar balanced separator theorem has two variants (see [22]). One variant (called *planar balanced* $\sqrt{n}$*-separator theorem*) states that any n-vertex planar graph G has a set S of vertices such that $|S| = O(\sqrt{n})$ and the removal of S from G leaves no connected component with more than $2/3n$ vertices. Another variant (called *planar balanced shortest-paths separator theorem*) states that any n-vertex planar graph G has two shortest paths removal of which from G leaves no connected component with more than $2/3n$ vertices. Although the first variant of the planar balanced separator theorem has an extension to the class of disk graphs (which includes UDGs) (see [1]), the second variant of the theorem proved to be more useful in designing compact routing (and distance) labeling schemes for planar graphs (see [13, 25]). To the date, there was not known any extension of the planar balanced shortest-paths separator theorem to unit disk graphs. The paper [11] notes that

"Unfortunately, Thorup's algorithm uses balanced shortest-path separators in planar graphs which do not obviously extend to the unit-disk graphs."

and uses the well-separated pair decomposition to get fast approximate distance computations in UDGs. We do not know how to use the well-separated pair decomposition of an UDG G to design a compact and low delay routing labeling scheme for G. Application of the balanced $\sqrt{\cdot}$-separator theorem of [1] to UDGs can result only in routing (and distance) labeling schemes with labels of size no less than $O(\sqrt{n}\log n)$-bits per vertex. Our separator theorem allows us to get $O(\log^2 n)$-bit labels which is more suitable for the wireless ad hoc and sensor networks where the issues of memory size and power-conservation are critical.

Our new *balanced shortest-paths—3-hop-neighborhood separator* theorem allows us to build, for any n-vertex UDG $G = (V, E)$, a system $\mathcal{T}(G)$ of at most $2\log_{\frac{3}{2}} n + 2$ spanning trees of G such that, for any two vertices x and y of G, there exists a tree T in $\mathcal{T}(G)$ with $d_T(x,y) \leq 3 \cdot d_G(x,y) + 12$. That is, the distances in any UDG can be approximately represented by the distances in at most $2\log_{\frac{3}{2}} n + 2$ of its spanning trees. An earlier version of these results has appeared in [27] (see Section 3.4 and pages 124 and 125 of Section 3.5.5). Taking the union of all these spanning trees of G, we obtain a *hop* $(3,12)$*-spanner* H of G (i.e., a spanning subgraph H of G with $d_H(x,y) \leq 3 \cdot d_G(x,y) + 12$ for any $x, y \in V$) with at most $O(n \log n)$ edges. There is a number of papers describing different types of *power-spanners*, *length-spanners* and *hop-spanners* for UDGs (see [2, 8, 10, 17–19, 21] and literature cited therein). Many of those spanners have nice properties of being planar or sparse, or having bounded maximum degree or bounded length (or power or hop) *spanner-stretch*, or having localized construction. Unfortunately, neither of those papers develops or discusses any routing schemes which could translate the constant spanner-stretch bounds into some constant route-stretch bounds.

Finally, we would like to note that since the construction of our compact and low delay routing labeling scheme is centralized and time consuming (its complexity in worst case is $O(m^2 \log n)$ for a n-vertex m-edge UDG), it is best suited for static or less mobile wireless ad-hoc or sensor networks.

2 Notions and Notations

Let V be a set of $n = |V|$ nodes on the Euclidean plane and let $G = (V, E)$ be the unit disk graph (UDG) induced by those nodes. Let also $m = |E|$. For each edge (a, b) of G, by (a, b) we denote also the open straightline segment representing it, and by $|ab|$ the Euclidean length of the edge/segment (a, b). For simplicity, in what follows, we will assume that any two edges in G can intersect at no more than one point (i.e., no two intersecting edges are on the same straight line), and no three edges intersect at the same point.

For a path P of G, the *hop-count* of P is defined as the number of edges on P, the *length* of P is defined as the sum of the Euclidean length of its edges and the *power-consumption* of P is defined as the sum of the β-powers of the Euclidean length of its edges. For any two vertices x and y of G, we denote by

– $d_G(x, y)$, the *hop-distance* (or simply *distance*) in G between x and y, i.e., the minimum hop-count of any path connecting x and y in G,

– $l_G(x, y)$, the *length-distance* in G between x and y, i.e., the minimum length of any path connecting x and y in G,

– $p_G(x, y)$, the *power-distance* in G between x and y, i.e., the minimum power-consumption of any path connecting x and y in G.

A graph family $\varGamma$ is said (see [23]) to have an $l(n)$ *bit* (s, r)*-approximate distance labeling scheme* if there is a function L labeling the vertices of each n-vertex graph in $\varGamma$ with distinct labels of up to $l(n)$ bits, and there exists an algorithm/function f, called *distance decoder*, that given two labels $L(v), L(u)$ of two vertices v, u in a graph G from $\varGamma$, computes, in time polynomial in the length of the given labels, a value $f(L(v), L(u))$ such that $d_G(v, u) \leq f(L(v), L(u)) \leq s \cdot d_G(v, u) + r$. Note that the algorithm is not given any additional information, other that the two labels, regarding the graph from which the vertices were taken. Similarly, a family $\varGamma$ of graphs is said (see [23]) to have an $l(n)$ *bit routing labeling scheme* if there exist a function L, labeling the vertices of each n-vertex graph in $\varGamma$ with distinct labels of up to $l(n)$ bits, and an efficient algorithm/function, called the *routing decision* or *routing protocol*, that given the label $L(v)$ of a current vertex v and the label $L(u)$ of the destination vertex u (the header of the packet), decides in time polynomial in the length of the given labels and using only those two labels, whether this packet has already reached its destination, and if not, to which neighbor of v to forward the packet.

Let $\mathcal{R}$ be a routing scheme and $R(x,y)$ be a route (path) produced by $\mathcal{R}$ for a pair of vertices x and y in a graph G. We say that $\mathcal{R}$ has

– *hop* (α,β)*-route-stretch* if hop-count of $R(x,y)$ is at most $\alpha \cdot d_G(x,y)+\beta$, for any $x,y \in V$,

– *length* (α,β)*-route-stretch* if length of $R(x,y)$ is at most $\alpha \cdot l_G(x,y)+\beta$, for any $x,y \in V$,

– *power* (α,β)*-route-stretch* if power-consumption of $R(x,y)$ is at most $\alpha \cdot p_G(x,y)+\beta$, for any $x,y \in V$.

Let $H=(V,E')$ be a spanning subgraph of a graph $G=(V,E)$. We say that H is

– *hop* (α,β)*-spanner* of G if $d_H(x,y) \le \alpha \cdot d_G(x,y)+\beta$, for any $x,y \in V$,

– *length* (α,β)*-spanner* of G if $l_H(x,y) \le \alpha \cdot l_G(x,y)+\beta$, for any $x,y \in V$,

– *power* (α,β)*-spanner* of G if $p_H(x,y) \le \alpha \cdot p_G(x,y)+\beta$, for any $x,y \in V$.

In Section 6, we will need also the notion of collective tree spanners from [6]. It is said that a graph G admits a system of μ collective tree (α,β)-spanners if there is a system $\mathcal{T}(G)$ of at most μ spanning trees of G such that for any two vertices x,y of G a spanning tree $T \in \mathcal{T}(G)$ exists such that $d_T(x,y) \le \alpha \cdot d_G(x,y)+\beta$.

For a vertex v of G, the kth *neighborhood* of v in G is the set $N_G^k[v]=\{u \in V : d_G(v,u) \le k\}$. For a vertex v of G, the sets $N_G[v]=N_G^1[v]$ and $N_G(v)=N_G[v]\setminus\{v\}$ are called the *neighborhood* and the *open neighborhood* of v, respectively. For a set $S \subseteq V$, by $N_G^k[S]=\bigcup_{v\in S} N_G^k[v]$ we denote the kth *neighborhood* of S in G.

3 Intersection Lemmas

In this section we present few auxiliary lemmas. From the definition of unit disk graphs, we immediately conclude the following.

Lemma 1. *In an UDG $G=(V,E)$, if edges $(a,b),(c,d) \in E$ intersect, then G must have at least one of $(a,c),(b,d)$ and at least one of $(a,d),(c,b)$ in E.*

Proof. Let o be the intersection point of (a,b) and (c,d). We know that $|ab| \le 1$ and $|cd| \le 1$. According to the triangle inequality, $|ao|+|co| > |ac|$ and $|bo|+|do| > |bd|$. Combining these inequalities, we get $2 \ge |ab|+|cd| = |ao|+|ob|+|co|+|od| > |ac|+|bd|$. The latter implies that $|ac| \le 1$ or $|bd| \le 1$, i.e., (a,c) or (b,d) must be in E. Similarly, one can show that (a,d) or (c,b) must be in E. □

Let r be an arbitrary but fixed vertex of an UDG $G=(V,E)$, and $L_0, L_1, \dots L_q$ be the *layering* of G with respect to r, where $L_i=\{u \in V : d_G(r,u)=i\}$. For G, using this layering, we construct a *layering*

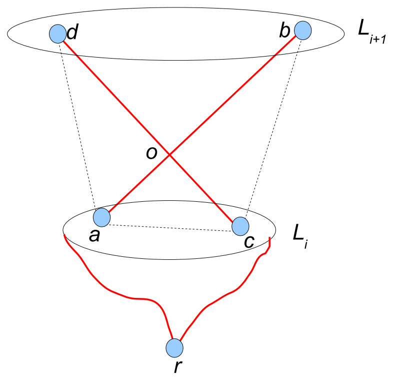


Fig. 1. A crossing of two edges (a, b) and (c, d), where a, c belong to layer L_i and b, d belong to layer L_{i+1}.

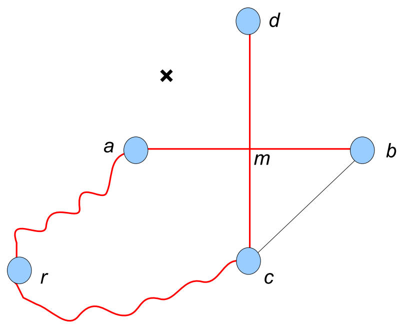


Fig. 2. A crossing of two tree-edges (a, b) and (c, d) implies that (c, b) is an edge in G and (a, d) is not an edge in G.

tree T_{orig} rooted at r as follows: each vertex $v \in L_i$ ($i \in \{1, \dots, q\}$) chooses a neighbor u in L_{i-1} such that $|vu|$ is minimum (closest neighbor in L_{i-1}) to be its father in T_{orig} (breaking ties arbitrarily). Let $E(T_{orig})$ be the edge set of T_{orig}. This tree T_{orig} will help us to construct a balanced separator for G. It will be convenient, for each vertex $v \in V$, by $L(v)$ to denote the layer index of v, i.e., $L(v) = d_G(r, v)$. In what follows, we will also adopt the following agreements (unless otherwise is specified). When we refer to any edge (a, b) of T_{orig}, we assume $L(a) = L(b) - 1$. When we refer to any two intersecting edges (a, b) and (c, d) of T_{orig} (in that order), we assume that $L(a) \leq L(c)$.

Lemma 2. *In T_{orig}, no two edges (a, b) and (c, d) with $L(a) = L(c)$ and $L(b) = L(d)$ can cross.*

Proof. We prove by contradiction. Assume that edges (a, b) and (c, d) cross. Let the crossing point be o, as shown in Fig. 1. By the triangle inequality, $|ao| + |do| > |ad|$ and $|bo| + |co| > |bc|$. Combining the two inequalities, we get $|ab| + |cd| = |ao| + |ob| + |co| + |od| > |ad| + |bc|$, which implies $2 \max\{|ab|, |cd|\} \geq |ab| + |cd| > |ad| + |bc| \geq 2 \min\{|ad|, |bc|\}$. Without loss of generally, assume $|bc| \leq |ad|$. Then, $|bc| < \max\{|ab|, |cd|\}$. If $|bc| < |ab|$, then according to our layering tree construction rule, b would choose c rather than a as its father, a contradiction. Assume now that $|bc| \geq |ab|$. Then $\angle bac \geq \angle bca$, which implies $\angle oac > \angle oca$ and hence $|oc| > |oa|$. By the triangle inequality, $|ad| < |do| + |ao|$. Since $|oc| > |oa|$, we get $|ad| < |do| + |oc| = |dc|$. By the layering tree construction rule, d would choose a rather than c as its father, a contradiction. □

Lemma 3. *Let $(a, b), (c, d)$ be two edges in T_{orig} that intersect. If $L(a) = L(b) - 1$, $L(c) = L(d) - 1$ and $L(a) \leq L(c)$, then $L(a) = L(c) - 1$, $(a, d) \notin E$ and $(b, c) \in E$.*

Proof. By Lemma 2, $L(a) \neq L(c)$, i.e., $L(a) < L(c)$. By Lemma 1, $(a.d)$ or (c, b) is an edge of G. Hence, $L(a) \geq L(c) - 2$. Similarly, by Lemma 1, $L(a) \geq L(d) - 2$. Thus, $L(a) = L(c) - 1 = L(d) - 2$ must hold.

Now, because $L(a) = L(d) - 2$, (a, d) can not be an edge of G. Then, by Lemma 1, $(c, b) \in E$. Fig. 2 is an illustration. $\square$

For an UDG $G = (V, E)$, in what follows, by $G_p = (V_p, E_p)$ we denote the planar graph obtained from G by turning each edge intersection point in G into a vertex in G_p. The vertices of T_{orig} (i.e. vertices of G) will be called *real vertices*, to differentiate them from *imaginary* and *null* points that will be defined later. In the following, we will use the term "element" as a general name for real vertices, imaginary points and null points. For any graph $\mathcal{G}$, we will use $E(\mathcal{G})$ to denote the set of its edges and $V(\mathcal{G})$ to denote the set of its vertices (or elements, if $V(\mathcal{G})$ contains imaginary or null points). Below, we will create an imaginary point (details will be given later) at the point where two edges (a, b) and (c, d) from T_{orig} intersect. Recall that we agreed to assume that $L(a) = L(b) - 1$, $L(c) = L(d) - 1$ and $L(a) \leq L(c)$. By Lemma 3, we know that $L(a) = L(c) - 1$. Now, assuming that the imaginary point is m, we define $a(m) = a$, $b(m) = b$, $c(m) = c$ and $d(m) = d$.

4 Balanced Separator for Restricted UDGs

In this section, we consider a special unit disk graph, a simple-crossing UDG. On this simple case, we demonstrate our idea of construction of a balanced separator. It may help the reader to follow the much more complicated case, where we construct a balanced separator for an arbitrary UDG. We define a *simple-crossing UDG* to be an UDG $G = (V, E)$ with each edge crossing at most one other edge.

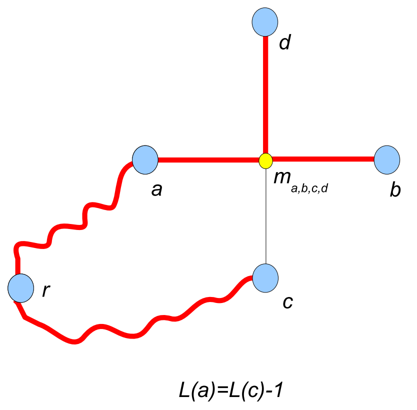


Fig. 3. Handling an intersection between two edges (a, b) and (c, d) of T_{orig} by creating an imaginary point $m_{a,b,c,d}$.

In what follows, we will transform tree T_{orig} into a special spanning tree T for the planar graph G_p. Let $T = T_{orig}$ initially. For each two intersecting edges (a, b) and (c, d) of T_{orig} (by Lemma 3, we know

$L(a) = L(c) - 1$), we do the following (see Fig. 3 for an illustration). Create a vertex $m_{a,b,c,d}$ at the point where (a, b) and (c, d) intersect. We call $m_{a,b,c,d}$ an *imaginary point*. Remove edges (a, b), (c, d) from T and add vertex $m_{a,b,c,d}$ and edges $(m_{a,b,c,d}, d)$, $(a, m_{a,b,c,d})$ and $(b, m_{a,b,c,d})$ into T. One can see that all the descendants of b and d in T find their way to the root via a.

There are two other kinds of edge intersections in G: the intersection between a tree-edge and a non-tree-edge and the intersection between two non-tree-edges. We handle them separately (see Fig. 4 for an illustration).

– Assume a tree-edge (u, w) intersects a non-tree-edge (s, t). We create a new vertex, called a *null point*, say o, at the point where (u, w) and (s, t) intersect. We remove edge (u, w) from T and add vertex o and edges (u, o), (o, w) into T.
– Assume two non-tree-edges (a, b) and (c, d) intersect. We create a new vertex, called a *null point*, say o, at the point where (a, b) and (c, d) intersect. We add vertex o (as a pendant vertex) and edge (a, o) into T.

It is easy to see that T is a spanning tree for the planar graph G_p. We will need the Lipton and Tarjan's planar separator theorem [22] in the following form.

Theorem 1 (Planar Separator Theorem). [22] *Let G be any planar graph with non-negative vertex weights and W be the total weight of G (which is the sum of the weights of its vertices). Let T be any spanning tree of G rooted at a vertex r. Then, there exist two vertices x and y in G such that if one removes from G the tree-paths connecting in T r with x and r with y, then each connected component of the resulting graph has total weight at most $2/3W$. Vertices x and y can be found in linear time.*

We can apply Theorem 1 to T and G_p by letting the weight of each real vertex be 1 and the weight of each imaginary or null point be 0 in G_p. Then, there must exist in T two paths $P_1 = P_T(r, x)$ and $P_2 = P_T(r, y)$ such that removal of them from G_p leaves no connected component with more than $2/3n$ real vertices.

Using paths $P_1 = (x_0 = r, x_1, \ldots, x_{k-1}, x_k = x)$ and $P_2 = (y_0 = r, y_1, \ldots, y_{l-1}, y_l = y)$ of G_p (of T), we can create a balanced separator for G as follows.

(1) Skip all the null points in P_1 and P_2.
(2) Skip every imaginary point in P_i which is collinear with its two neighbors in P_i $(i = 1, 2)$.
(3) For any imaginary point $m_{a,b,c,d}$ in P_i $(i = 1, 2)$ which is not collinear with its two neighbors in P_i (the only possible case is shown in Fig. 3, where $L(a) = L(c)-1$ and imaginary point $m_{a,b,c,d}$ connects a and d in P_i), replace the subpath $(a, m_{a,b,c,d}, d)$ by either (a, c, d) (if $(a, c) \in E$) or (a, b, d) (if $(b, d) \in E$). By Lemma 1, (a, c) or (b, d) is in E.

Let P'_i be the resulting path obtained from P_i $(i = 1, 2)$. It is easy to check that P'_1 and P'_2 are shortest paths in G. Here and in what follows, by a shortest path we mean a hop-shortest path. We can also show that the union of $N^1_G[P'_1]$ and $N^1_G[P'_2]$ is a balanced separator for G, i.e., removal of $N^1_G[P'_1] \cup N^1_G[P'_2]$ from G leaves no connected component with more that $2/3n$ vertices. Assume that removal of P_1 and P_2 from $G_p = (V_p, E_p)$ results in removing a set of edges E'_p from E_p, and removal of $N^1_G[P'_1]$ and $N^1_G[P'_2]$ from $G = (V, E)$ results in removing a set of edges E' from E. It is easy to check that, for any edge $e'_p \in E'_p$ there exists an edge $e' \in E'$ that covers e'_p. The latter implies that the union of $N^1_G[P'_1]$ and $N^1_G[P'_2]$ is a balanced separator for G. A formal proof of this will be presented in Section 5 (Theorem 3) where the general case, i.e., arbitrary UDGs, are discussed.

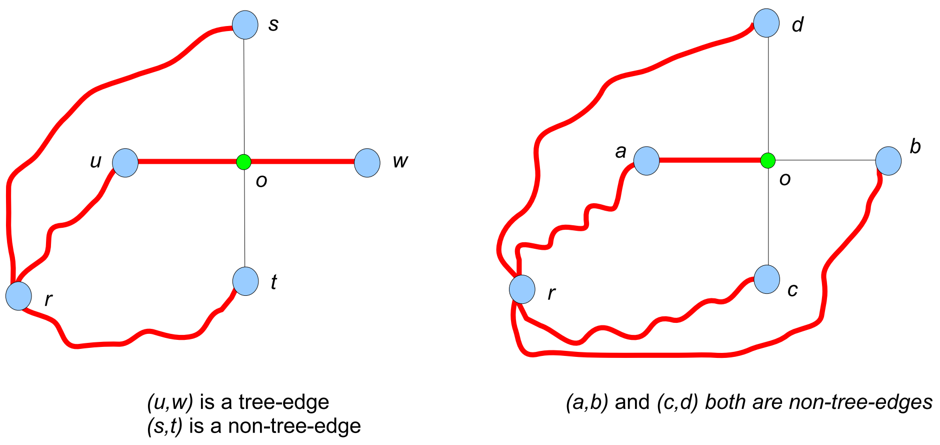

s
u
o
w
r
t
(u,w) is a tree-edge
(s,t) is a non-tree-edge
d
a
o
b
r
c
(a,b) and *(c,d) both are non-tree-edges*

Fig. 4. (left) Handling an intersection between a tree-edge and a non-tree-edge. (right) Handing an intersection between two non-tree-edges.

5 Balanced Separator for Arbitrary UDGs

In an arbitrary unit disk graph $G = (V, E)$, an edge may cross any number of other edges. Our basic strategy for building a balanced separator for G is similar to one we used in the case of a simple-crossing UDG, but details are more complicated. Let $T = T_{orig}$ initially. We will revise T to create a special spanning tree for the planar graph G_p obtained from G. Then, we will apply the Planar Separator Theorem from [22] (Theorem 1 above) to G_p and T to get a balanced separator S for G_p. Finally, we will recover from S the required separator for G.

5.1 Building a special spanning tree T of G_p

In what follows, the edges of the tree T_{orig} will be called *original tree-edges*. By Lemma 3, for any two intersecting original tree-edges (a, b) and (c, d) (for which we assumed that $L(a) = L(b) - 1$, $L(c) = L(d) - 1$

and $L(a) \leq L(c)$), we have $L(a) = L(c) - 1$, $(a, d) \notin E(G)$ and $(b, c) \in E(G)$. We handle this kind of intersections (between original tree-edges) using PROCEDURE 1. Fig. 5 gives a running example.

PROCEDURE 1. Handle original tree-edge intersections

Input: A layering tree T_{orig} rooted at r.

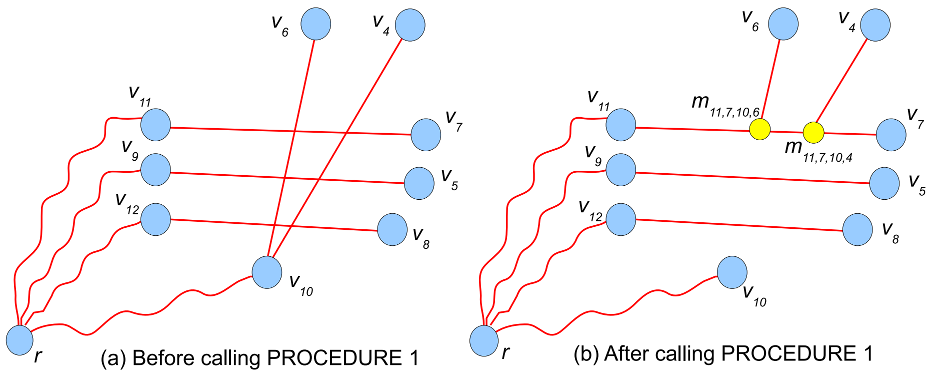
Output: A tree T where all original tree-edge intersections resolved.

Method: /* Break ties arbitrarily */

```
(1)  Let $L_i = \{v : L(v) = i\}$ and $T = T_{orig}$;
(2)  Let $q$ be the maximum layer number of $T$;
(3)  **FOR** $i = 1$ to $q$ **DO**
(4)    **FOR** each vertex $v_j \in L_i$ **DO**
(5)      **FOR** each vertex $v_k \in L_{i+1}$ adjacent to $v_j$ in $T$ **DO**
(6)        **IF** there is an original tree-edge intersection on $(v_j, v_k)$ such that $L(v_j)$ is the SECOND smallest
             layer index among the layer indices of all four end-vertices of the two edges giving the intersection
           **THEN DO**
(7)          Choose such an original tree-edge intersection closest to $v_k$ and assume it is the intersection
             between $(v_j, v_k)$ and $(x, y)$ in $T$ and between $(v_j, v_k)$ and $(v_p, v_h)$ in $T_{orig}$ (i.e., $(x, y) \subseteq (v_p, v_h)$);
(8)          Create an imaginary point $m_{j,k,p,h}$ at the point where $(v_j, v_k)$ and $(x, y)$ intersect;
(9)          Update $T$ by removing edges $(v_j, v_k)$ and $(x, y)$, and adding vertex $m_{j,k,p,h}$ and
             edges $(m_{j,k,p,h}, x)$, $(m_{j,k,p,h}, y)$, $(m_{j,k,p,h}, v_k)$;
(10)       **ENDIF**
(11)     **ENDFOR**
(12)   **ENDFOR**
(13) **ENDFOR**
(14) **RETURN** $T$
```

Lemma 4. *PROCEDURE 1 returns a tree T with all original tree-edge intersections resolved (i.e., edges of T do not cross each other).*

Proof. First, T contains no tree-edge intersections. This is because, in steps (3)-(13) of PROCEDURE 1, each tree-edge intersection, with v_j as the second smallest layer index among the layer indices of all four end-vertices of the two edges giving the intersection, has been eliminated or converted to an imaginary point. Second, one can easily check that each vertex in T has the father except the root. Therefore, T is still a tree. □



$d_G(v_{11},r)=d_G(v_9,r)=d_G(v_{12},r)=d_G(v_{10},r)-1$

Fig. 5. A running example for PROCEDURE 1.

In addition, there are two other kinds of intersections remaining: the intersection between an edge in $E(T)$ (T-edge) and an edge in $E(G) \setminus E(T)$ (non-T-edge), and intersection between two non-T-edges.

First we handle intersections between T-edges and non-T-edges. They are resolved the same way as in Section 4. Here, we rephrase the rule. Assume (u, w) is a T-edge, (s, t) is a non-T-edge. Add a null point, say o, at the point where (u, w) and (s, t) intersect. Remove edge (u, w) from T and add vertex o and edges (u, o), (o, w) into T. After resolving all intersections of this kind, T becomes a subgraph of G_p. Note that it is possible that T does not span yet all elements of $V(G_p)$. *Let name this T as T_{sub}.*

Now, we deal with intersections between two non-T_{sub}-edges. This is more complicated than it was in Section 4 for restricted UDGs. We will grow T_{sub} to a spanning tree T_{span} for G_p (extension T_{span} of T_{sub} will cover all elements of $V(G_p)$). We use a procedure similar to one of building a shortest path tree from a set of vertices. We assign to each vertex in T_{sub} a weight according to the following formula. In formula, if v is an imaginary point or a null point, we assume v is at the intersection between edges (a, b) and (c, d) of G.

$$weight(v) = \begin{cases} 0, & \text{if } v \text{ is a real vertex;} \\ min\{|av|, |bv|, |cv|, |dv|\}, & \text{if } v \text{ is an imaginary or a null point.} \end{cases}$$

To build our spanning tree for G_p, we use PROCEDURE 2. At the beginning, for any $v \in V(G_p)\setminus V(T_{sub})$, $distance[v] = \infty$ and father of v is undefined.

PROCEDURE 2. Build a spanning tree for G_p from T_{sub}

Input: A tree $T = T_{sub}$;

Output: A tree T_{span} as a spanning tree for G_p.

Method: /* Break ties arbitrarily */

```
(1)  FOR each $i$ in $V(T)$ DO
```

```
(2)     FOR each neighbor $j \in V(G_p)\backslash V(T)$ of $i$ DO
(3)       $tmp := weight[i] + |ij|$;
(4)       IF $tmp < distance[j]$ DO
(5)         $distance[j] := tmp$;
(6)         $father[j] := i$;
(7)       ENDIF
(8)     ENDFOR
(9)   ENDFOR
(10) $Q := V(G_p)\backslash V(T)$;
(11) WHILE $Q$ is not empty DO
(12)    $u$ :=node in $Q$ with smallest distance[$\cdot$];
(13)    remove $u$ from $Q$ and add $u$ into $T$;
(14)    FOR each neighbor $v \in Q$ of $u$ DO
(15)      $tmp := distance[u] + |uv|$;
(16)      IF $tmp < distance[v]$ DO
(17)        $distance[v] := tmp$;
(18)        $father[v] := u$;
(19)      ENDIF
(20)    ENDFOR
(21) ENDWHILE
(22) RETURN $T_{span} := T$.
```

It is easy to check that T_{span} is a spanning tree of the planar graph G_p.

5.2 Finding a balanced 2×shortest-paths—3-hop-neighborhood separator for G

Now we can apply Theorem 1 to G_p and T_{span} by letting the weight of each real vertex be 1 and the weight of each imaginary or null point be 0, and get a balanced separator S of G_p. Assume that S is the union of paths $P_1 = P_{T_{span}}(r, x)$ and $P_2 = P_{T_{span}}(r, y)$. There are three kinds of elements on P_1 and P_2: real vertices, imaginary points and null points. Generally, each imaginary point or null point is adjacent to at most four elements in G_p, and each element in P_1 or P_2 has the previous element and the next element, except for the root r (it has only the next element) and elements x and y (they have only the previous element). Let u be the last real or imaginary point in P_1 (or P_2). We name all null points after u in P_1 (or P_2) as the *tail null points*. For any element in P_1 or P_2, there are two possible relations between itself, its previous element and its next element:

– the element, its previous element and its next element are on the same line, which means its previous element and its next element are on the same edge of G (according to our general assumption that no two edges of G are on the same line);
– the element, its previous element and its next element are not on the same line, which means its previous element and itself are on one edge of G, and its next element and itself are on another edge of G.

Using paths $P_1 = (x_0 = r, x_1, \ldots, x_{k-1}, x_k = x)$ and $P_2 = (y_0 = r, y_1, \ldots, y_{l-1}, y_l = y)$ of G_p (of T_{span}), We will find the corresponding balanced separator for G using the following steps:

(1) We skip all null points in P_1 and P_2. Let the resulting paths be P_1' and P_2', respectively.
(2) We skip in P_1' and P_2' each imaginary point whose previous element and next element are on the same edge of T_{orig}. For example, let (x_f, x_i, x_j) be a fragment of path P_1' or P_2', where x_i is an imaginary point and $\{x_f, x_i, x_j\}$ are collinear, then (x_f, x_i, x_j) will be replaced with (x_f, x_j). Let the resulting paths be P_1'' and P_2'', respectively.
(3) Replace each remaining imaginary point m in P_1'' and P_2'' with two vertices: $b(m)$ followed by $c(m)$ (see end of Section 3 for these notations). For example, let (x_f, x_i, x_j) be a fragment of path P_1'' or P_2'', where x_i is an imaginary point and x_f is closest to the root r among $\{x_f, x_i, x_j\}$. Then, (x_f, x_i, x_j) will be replaced with $(x_f, b(x_i), c(x_i), x_j)$. Let the resulting paths be P_1''' and P_2''', respectively. By Lemma 3, the edge $(b(x_i), c(x_i))$ exists in G. It is easy to check that P_1''' and P_2''' are valid paths in G.

In what follows we will prove that P_1''' and P_2''' are 2×shortest paths of G. We define 2×shortest paths of G as follows.

Definition 1. *A path P of G is a 2×shortest path iff for any two vertices x,y in P, $d_P(x, y) \leq 2d_G(x, y)$.*

We will need the following lemma.

Lemma 5. *For any element $v \in P_1''$, $d_{P_1''}(v, r) = d_{T_{orig}}(v, r) = d_G(v, r)$ if v is a real vertex, and $d_{P_1''}(v, r) = d_{T_{orig}}(c(v), r) = d_{T_{orig}}(b(v), r) = d_G(c(v), r) = d_G(b(v), r)$ if v is an imaginary point.*

Proof. P_1'' contains real vertices and imaginary points but no null points. For two adjacent elements x and y in P_1'', where x is y's previous vertex, there are four possible cases.

– *x is a real vertex and y is a real vertex*: We immediately have $d_{T_{orig}}(x, r) = d_{T_{orig}}(y, r) - 1$ by the construction of T_{orig}.
– *x is an imaginary point and y is an imaginary point*: We have $d_{T_{orig}}(c(x), r) = d_{T_{orig}}(c(y), r) - 1$ by PROCEDURE 1. One can also refer to Figure 7 (as an example), where $x = v_i$ and $y = v_{i+1}$.

– *x is an imaginary point and y is a real vertex*: In this case, y can only be $d(x)$ but not $b(x)$. If y is $b(x)$, x should be removed in step (2) because it is collinear with its previous and next elements. Thus, we have $d_{T_{orig}}(c(x), r) = d_{T_{orig}}(y, r) - 1$.

– *x is a real vertex and y is an imaginary point*: In this case, x can only be $a(y)$ and we have $d_{T_{orig}}(x, r) = d_{T_{orig}}(c(y), r) - 1$ by Lemma 3.

Note also that, for any real vertex v, $d_{T_{orig}}(v, r) = d_G(v, r)$ because T_{orig} is a layering tree, and, for any imaginary point m, $d_{T_{orig}}(c(m), r) = d_{T_{orig}}(b(m), r) = d_{T_{orig}}(d(m), r) - 1$ according to Lemma 3.

Now, we can show that the lemma is correct by mathematical induction. For the root r and its next element v in P_1'', the above first or last case applies and the lemma is true. Suppose the lemma is true for the subpath of P_1'' from r to v_i. Then, it is easy to check that it is also true for the subpath of P_1'' from r to v_{i+1}, by applying the above four cases. □

Clearly, similar statement is true for path P_2''. Now we are ready to prove that P_1''' and P_2''' are $2\times$shortest paths of G.

Theorem 2. *P_1''' and P_2''' are $2\times$shortest paths in G.*

Proof. We will show the proof only for path P_1'''. Since each imaginary point in P_1'' is replaced by two vertices in step (3), for any two vertices u and w in P_1''', we have $d_{P_1'''}(u, w) \leq 2d_{P_1''}(f(u), f(w))$, where $f(\cdot)$ is defined as follows: for a real vertex v in P_1'', since it is still available in P_1''', $f(v) = v$; for an imaginary point m in P_1'', since it is replaced by real vertices $c(m)$ and $b(m)$ in P_1''', $f(c(m)) = m$ and $f(b(m)) = m$.

By Lemma 5, we have $d_{P_1''}(f(u), f(w)) = |d_{P_1''}(f(u), r) - d_{P_1''}(f(w), r)| = |d_G(u, r) - d_G(w, r)|$. By the triangle inequality, $d_G(u, r) - d_G(w, r) \leq d_G(u, w)$. Combining all this, we get $d_{P_1'''}(u, w) \leq 2d_G(u, w)$. □

Finally, we have the following separator theorem for an UDG G.

Theorem 3. *The union of $N_G^3[P_1''']$ and $N_G^3[P_2''']$, where P_1''' and P_2''' are $2\times$shortest paths of G described above, is a balanced separator for G with $2/3$-split, i.e., removal of $N_G^3[P_1'''] \cup N_G^3[P_2''']$ from G leaves no connected component with more than $2/3n$ vertices.*

Proof. We know that the union of P_1 and P_2 is a balanced separator (with 2/3-split) for $G_p = (V_p, E_p)$. Recall that G_p is the planar graph obtained from G by turning each edge intersection in $G = (V, E)$ into a graph vertex in G_p. Therefore, according to our general assumption, for any edge $e_p \in E_p$, there exists one and only one edge $e \in E$ such that e covers e_p. We say e covers e_p if $e_p \subseteq e$ as geometric segments. The removal of P_1 and P_2 from G_p will result in removing a set of elements and a set of edges (say E_p', $E_p' \subseteq E_p$) from G_p. Meanwhile, the removal of $N_G^3[P_1''']$ and $N_G^3[P_2''']$ from G will also result in removing a set of vertices and a set of edges (say E', $E' \subseteq E$) from G. We have the following claim.

Claim (1). If for any edge $e'_p \in E'_p$ there exists an edge $e' \in E'$ that covers e'_p, then the union of $N^3_G[P'''_1]$ and $N^3_G[P'''_2]$ is a balanced separator for G with 2/3-split.

Proof. Since erasing edges E'_p from G_p results in no connected component of G_p with more than $2/3n$ real vertices, and any edge in E'_p is covered by an edge in E', erasing edges E' from G will also result in no connected component of G with more than $2/3n$ vertices. □(Claim)

In what follows, we will prove that for any edge $e'_p \in E'_p$ there exists an edge $e' \in E'$ that covers e'_p.

We can classify edges in E'_p into four classes: class A is all edges for which at least one end is a real vertex from P'''_1 or P'''_2; class B is all edges in $E'_p \backslash A$ for which at least one end is an imaginary point from P''_1 or P''_2; class C is all edges in $E'_p \backslash (A \bigcup B)$ for which at least one end is an imaginary point from P'_1 or P'_2; class D is all edges in $E'_p \backslash (A \bigcup B \bigcup C)$ (all remaining edges). One can conclude that each edge in D has at least one end as a null point from P_1 or P_2.

It is easy to check with Lemma 1 that edges in A, B and C are covered by edges in E'. If edge $e \in D$ has an end as a null point on the edge between two real vertices in P'''_1 or P'''_2, then one can infer, by Lemma 1, that e must be covered by an edge in E', too. Any other edge $e \in D$ has an end which is a tail null point in P_1 or P_2 (see PROCEDURE 2).

To facilitate our discussion, for a tail null point o corresponding to an intersection between two non-T_{sub}-edges, assume the two edges are $(r_1(o), r_2(o))$ and $(r_3(o), r_4(o))$ from $E(G)$. We know that $\{r_1(o), r_2(o), r_3(o), r_4(o)\} \subseteq V(T_{sub})$.

Claim (2). If u is the last real or imaginary point in P_1 (or P_2), then for any tail null point o (at the intersection between edges $(r_1(o), r_2(o))$ and $(r_3(o), r_4(o))$) in P_1 (or P_2), we have $\{r_1(o), r_2(o), r_3(o), r_4(o)\} \subseteq N^3_G[u]$ if u is a real vertex, and $\{r_1(o), r_2(o), r_3(o), r_4(o)\} \subseteq N^3_G[c(u)]$ if u is an imaginary point.

Proof. Suppose w is the last T_{sub} element in P_1. w could be a real vertex, an imaginary point or a null point. There are four cases (see Figure 6).

(1): *$w = u$ and u is a real vertex.* Then we claim $|ur_1(o)| \leq 1$, $|ur_2(o)| \leq 1$, $|ur_3(o)| \leq 1$, $|ur_4(o)| \leq 1$. This claim can be proved by observing that the length $l_{T_{span}}(u, o)$ (and, hence, $|uo|$) is not larger than $|r_1(o)o|$, $|r_2(o)o|$, $|r_3(o)o|$ and $|r_4(o)o|$, according to PROCEDURE 2. Since $|r_1(o)r_2(o)| = |r_1(o)o| + |r_2(o)o| \leq 1$ and $|r_3(o)r_4(o)| = |r_3(o)o| + |r_4(o)o| \leq 1$, we have $|ur_1(o)| \leq 1$, $|ur_2(o)| \leq 1$, $|ur_3(o)| \leq 1$, $|ur_4(o)| \leq 1$. Therefore, $\{r_1(o), r_2(o), r_3(o), r_4(o)\} \subseteq N^1_G[u]$.

(2): *$w = u$ and u is an imaginary point (at the intersection of edges $(a(u), b(u))$ and $(c(u), d(u))$ in G).* Then, similarly as in case (1), we have that at least one of $a(u)$,$b(u)$,$c(u)$ and $d(u)$ is within unit distance

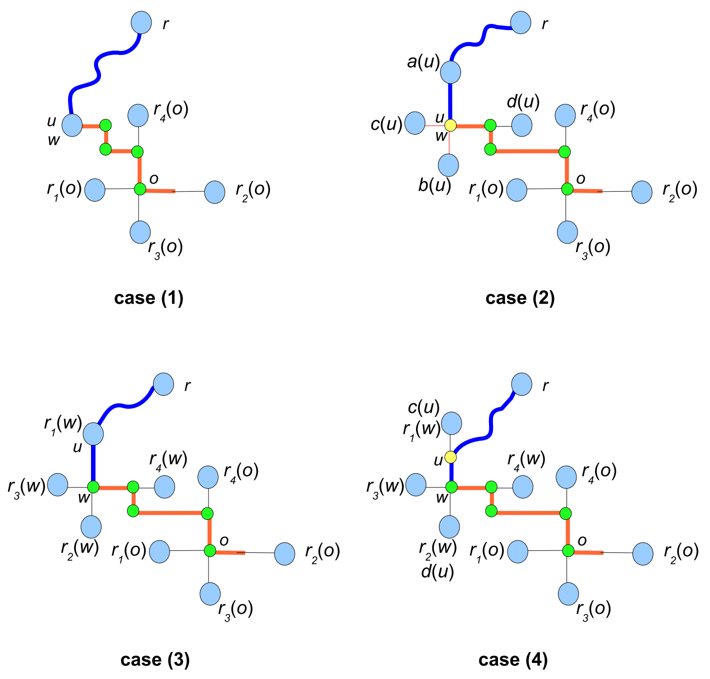


Fig. 6. Illustrations to the second claim in the proof of Theorem 3.

from $r_1(o), r_2(o), r_3(o), r_4(o)$. In addition, we know $\{a(u), b(u), c(u), d(u)\} \subseteq N^2_G[c(u)]$ by Lemma 1. Therefore, $\{r_1(o), r_2(o), r_3(o), r_4(o)\} \subseteq N^3_G[c(u)]$.

(3): *w is a null point (at the intersection between edges $(r_1(w), r_2(w))$ and $(r_3(w), r_4(w))$ in G) and u is a real vertex.* Since w is the last T_{sub} element in P_1 and u is the last real vertex or imaginary point in P_1, it is easy to see that u and w are on the same edge of G. Then, similarly as in case (2), we have that at least one of $r_1(w)$, $r_2(w)$, $r_3(w)$ and $r_4(w)$ is within unit distance from $r_1(o), r_2(o), r_3(o), r_4(o)$ and $\{r_1(w), r_2(w), r_3(w), r_4(w)\} \subseteq N^2_G[u]$. Therefore, $\{r_1(o), r_2(o), r_3(o), r_4(o)\} \subseteq N^3_G[u]$.

(4): *w is a null point (at the intersection between edges $(r_1(w), r_2(w))$ and $(r_3(w), r_4(w))$ in G) and u is an imaginary point.* Since w is the last T_{sub} element in P_1 and u is the last real or imaginary point in P_1, it is easy to see that u and w are on the same edge of G. Then, similarly as in case (2), we have that at least one of $r_1(w)$, $r_2(w)$, $r_3(w)$ and $r_4(w)$ is within unit distance from $r_1(o), r_2(o), r_3(o), r_4(o)$ and $\{r_1(w), r_2(w), r_3(w), r_4(w)\} \subseteq N^2_G[c(u)]$. Therefore, $\{r_1(o), r_2(o), r_3(o), r_4(o)\} \subseteq N^3_G[c(u)]$. □(Claim)

Thus, for any edge $e'_p \in E'_p$ there exists an edge $e' \in E'$ that covers e'_p. Hence, the theorem is proved. □

Theorem 2 and Theorem 3 tell us that there exist two paths P'''_1 and P'''_2 in G such that they are 2×shortest paths and the union of $N^3_G[P'''_1]$ and $N^3_G[P'''_2]$ is a balanced separator for G.

5.3 Finding a balanced shortest-paths—3-hop-neighborhood separator for G

In this section, we will improve the result of Section 5.2. We will show that any UDG G has two shortest paths P_1''' and P_2''' such that the union of $N_G^3[P_1''']$ and $N_G^3[P_2''']$ forms a balanced separator for G. Recall that, by a shortest path we mean a hop-shortest path.

Let P_1, P_2, P_1', P_2', P_1'' and P_2'' be the paths defined in Section 5.2. Analogs of paths P_1''' and P_2''' of Section 5.2 will be obtained from P_1'' and P_2'' in a more careful way (than in Section 5.2). We use PROCEDURE 3 for this.

PROCEDURE 3. Handle imaginary points

Input: Path $P \in \{P_1'', P_2''\}$ (containing still some imaginary points).

Output: Path P as a shortest path of G, with all imaginary points resolved.

Method: /* Break ties arbitrarily. */ /* The first vertex in P is the root r, a real vertex.*/

```
(1) Let $[v_1, \cdots, v_k]$ be the imaginary points in $P$ in the order from $r$;
(2) FOR $i = 1$ to $k$ DO
(3)   IF vertex $c(v_i)$ is adjacent to $prev_P(v_i)$ ($c(v_i)$ is always adjacent to $next_P(v_i)$, as it will be shown later.)
(4)     Replace $v_i$ with $c(v_i)$ in $P$;
(5)   ELSE (It implies that vertex $b(v_i)$ is adjacent to both $prev_P(v_i)$ and $next_P(v_i)$, as it will be shown later.)
(6)     Replace $v_i$ with $b(v_i)$ in $P$;
(7)   ENDIF
(8) ENDFOR
(9) RETURN $P$
```

We call PROCEDURE 3 for both P_1'' and P_2''. Let the resulting paths be P_1''' and P_2''', respectively. We have the following lemma.

Lemma 6. *In PROCEDURE 3, when an imaginary point v_i is replaced by v_i' (v_i' is either $c(v_i)$ or $b(v_i)$) in current P, we have $|prev_P(v_i)v_i'| \leq 1$, $|v_i'next_P(v_i)| \leq 1$ and $|v_i'd(v_i)| \leq 1$.*

Proof. We will show the proof only for path P_1''. According to the construction of path P_1'' from P_1' (see step (2) in Section 5.2), if v_i is an imaginary point in P_1'', then $prev_{P_1''}(v_i)$, v_i and $next_{P_1''}(v_i)$ cannot be collinear. For an imaginary point v_i in P_1'', its previous element in P_1'' is either a real vertex or an imaginary point. For the first imaginary point in P_1'', its previous element is a real vertex. We prove the lemma by mathematical induction.

Let v_1 be the first imaginary point in P_1''. Assume it is replaced by v_1' in P. We need to show that $|prev_P(v_1)v_1'| \leq 1$, $|v_1' next_P(v_1)| \leq 1$ and $|v_1' d(v_1)| \leq 1$.

According to our general assumption (no two intersecting edges are on the same line), we conclude $prev_P(v_1)$ is $a(v_1)$. In addition, $next_P(v_1)$ must lie on segment $(v_1, d(v_1))$. It cannot lie on segment $(v_1, b(v_1))$ since $prev_P(v_1)$, v_1 and $next_P(v_1)$ are not collinear. If $c(v_1)$ is chosen as v_1', we know $|prev_P(v_1)v_1'| \leq 1$ by PROCEDURE 3, and $|v_1' d(v_1)| \leq 1$ because $(c(v_1), d(v_1))$ is an edge in G. In addition, $|v_1' next_P(v_1)| \leq 1$ holds because $(v_1', d(v_1))$ is an edge in G and $next_P(v_1)$ lies on segment $(v_1, d(v_1))$. If $b(v_1)$ is chosen as v_1', $|prev_P(v_1)v_1'| \leq 1$ must hold because $(prev_P(v_1), b(v_1))$ is an edge in G. $|v_1' next_P(v_1)| \leq 1$ still holds because, by PROCEDURE 3, $|a(v_1)c(v_1)| > 1$ and, by Lemma 1, $|v_1' d(v_1)| \leq 1$, which implies $|b(v_1) next_P(v_1)| \leq 1$. The basis for induction is proved.

Now let assume that the lemma is true for $i < k$, i.e., when an imaginary point v_i is replaced with v_i' in P (v_i' is either $c(v_i)$ or $b(v_i)$), $|prev_P(v_i)v_i'| \leq 1$, $|v_i' next_P(v_i)| \leq 1$ and $|v_i' d(v_i)| \leq 1$ hold. We need to prove that the lemma is also true for $i + 1$.

Assume $c(v_{i+1})$ is chosen as v_{i+1}'. According to PROCEDURE 3, $|prev_P(v_{i+1})v_{i+1}'| \leq 1$. In addition, $next_P(v_{i+1})$ must lie on segment $(v_{i+1}, d(v_{i+1}))$. Since $(c(v_{i+1}), d(v_{i+1}))$ is an edge in G and $next_P(v_{i+1})$ lies on $(v_{i+1}, d(v_{i+1}))$, we have $|v_{i+1}' next_P(v_{i+1})| \leq 1$ and $|v_{i+1}' d(v_{i+1})| \leq 1$.

Assume now that $b(v_{i+1})$ is chosen as v_{i+1}'. There are two cases to consider.

(1): *$prev_{P_1''}(v_{i+1})$ is a real vertex.* According to our general assumption (no two intersecting edges are on the same line), we conclude $prev_P(v_{i+1})$ is $a(v_{i+1})$. Therefore, $|prev_P(v_{i+1})v_{i+1}'| \leq 1$ because $(a(v_{i+1}), b(v_{i+1}))$ is an edge in G. By PROCEDURE 3, $|prev_P(v_{i+1})c(v_{i+1})| > 1$. Then, by Lemma 1, we have $|v_{i+1}' d(v_{i+1})| \leq 1$, which implies $|v_{i+1}' next_P(v_{i+1})| \leq 1$ because $next_P(v_{i+1})$ lies on segment $(v_{i+1}, d(v_{i+1}))$.

(2): *$prev_{P_1''}(v_{i+1})$ is an imaginary point.* Let $prev_{P_1''}(v_{i+1})$ be v_i. The case is illustrated on Figure 7. As we discussed before, v_{i+1} is on the segment $(v_i, d(v_i))$. If $c(v_i) (\equiv a(v_{i+1}))$ is chosen as v_i', with similar arguments as in the case (1), we conclude that $|prev_P(v_{i+1})v_{i+1}'| \leq 1$, $|v_{i+1}' d(v_{i+1})| \leq 1$ and $|v_{i+1}' next_P(v_{i+1})| \leq 1$. If $b(v_i)$ is chosen as v_i', by mathematical induction, we know $|v_i' d(v_i)| \leq 1$, where $d(v_i) \equiv b(v_{i+1})$. By PROCEDURE 3, $|v_i' c(v_{i+1})| > 1$. If edges $(b(v_i), b(v_{i+1}))$ and $(c(v_{i+1}), d(v_{i+1}))$ of G intersect, then, by Lemma 1, we have $|v_{i+1}' d(v_{i+1})| \leq 1$, which also implies $|v_{i+1}' next_P(v_{i+1})| \leq 1$. So, it remains to prove that $(b(v_i), b(v_{i+1}))$ and $(c(v_{i+1}), d(v_{i+1}))$ intersect in G. Assume they do not intersect. Then, either $c(v_{i+1})$ or $d(v_{i+1})$ is inside $\triangle v_i b(v_i) b(v_{i+1})$, or edges $(b(v_i), a(v_i))$ and $(c(v_{i+1}), d(v_{i+1}))$ intersect. If $c(v_{i+1})$ is inside $\triangle v_i b(v_i) b(v_{i+1})$, then $|b(v_i)c(v_{i+1})| \leq 1$ must hold, contradicting $|v_i' c(v_{i+1})| > 1$. If $d(v_{i+1})$ is inside $\triangle v_i b(v_i) b(v_{i+1})$, then $(d(v_{i+1}), b(v_i))$ must be an edge in G, i.e., $d_G(r, d(v_{i+1})) \leq d_G(r, b(v_i)) + 1$, implying $L(d(v_{i+1})) \leq L(b(v_i)) + 1$. On the other hand, by Lemma 3, $L(d(v_{i+1})) = L(c(v_{i+1})) + 1 = L(b(v_{i+1})) + 1 = L(d(v_i)) + 1 = L(b(v_i)) + 2$, and a contradiction arises. Similarly, if edges $(b(v_i), a(v_i))$ and $(c(v_{i+1}), d(v_{i+1}))$ intersect, then, by Lemma 1, $a(v_i)$ must be adjacent with $d(v_{i+1})$ (since $c(v_{i+1})$ and $b(v_i)$ are not adjacent), contradicting with $L(d(v_{i+1})) = L(b(v_i)) + 2 = L(a(v_i)) + 3$.

Thus, the lemma is true for $i + 1$, too. This completes the entire proof. □

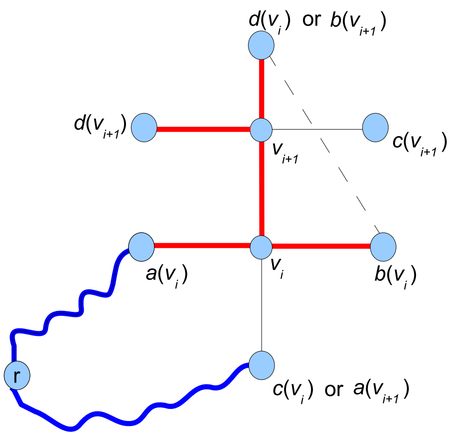


Fig. 7. An illustration to the proof of Lemma 6.

Combining Lemma 5 and Lemma 6, we obtain the following theorem.

Theorem 4. *P_1''' and P_2''' are shortest paths in G.*

Now, for the paths P_1''' and P_2''', a similar to Theorem 3 result holds.

Theorem 5. *The union of $N_G^3[P_1''']$ and $N_G^3[P_2''']$, where P_1''' and P_2''' are shortest paths of G described above, is a balanced separator for G with 2/3-split, i.e., removal of $N_G^3[P_1'''] \cup N_G^3[P_2''']$ from G leaves no connected component with more than $2/3n$ vertices.*

Proof. The proof is almost identical to the proof of Theorem 3. Only for cases (2) and (4) in the proof of *Claim (2)*, we need to make some additions to guarantee $\{r_1(o), r_2(o), r_3(o), r_4(o)\} \subseteq N_G^3[b(u)]$ if u is an imaginary point.

In case (2), we need to mention that, for an imaginary point u, both $\{a(u), b(u), c(u), d(u)\} \subseteq N_G^2[c(u)]$ and $\{a(u), b(u), c(u), d(u)\} \subseteq N_G^2[b(u)]$ hold. Therefore, $\{r_1(o), r_2(o), r_3(o), r_4(o)\} \subseteq N_G^3[c(u)]$ and $\{r_1(o), r_2(o), r_3(o), r_4(o)\} \subseteq N_G^3[b(u)]$.

In case (4), we need to add the following. If u is replaced with $c(u)(\equiv r_1(w))$ in PROCEDURE 3, *Claim (2)* still holds because $\{r_1(w), r_2(w), r_3(w), r_4(w)\} \subseteq N_G^2[r_1(w)]$. If u is replaced with $b(u)$ in PROCEDURE 3, by Lemma 6, $b(u)$ is adjacent to $d(u)(\equiv r_2(w))$. We also know that $b(u)$ is adjacent to $c(u)(\equiv r_1(w))$, by Lemma 3. If edge $(r_3(w), r_4(w))$ intersects $(b(u), a(u))$ or $(b(u), d(u))$, then, by Lemma 1, we also have $\{r_3(w), r_4(w)\} \subseteq N_G^2[b(u)]$. If $(r_3(w), r_4(w))$ intersects neither $(b(u), a(u))$ nor $(b(u), d(u))$, then $r_3(w)$ is inside $\triangle ub(u)d(u)$, implying that $|r_3(w)b(u)| \leq 1$, i.e., $r_3(w) \in N_G[b(u)]$ and $r_4(w) \in N_G^2[b(u)]$. Therefore, again $\{r_1(o), r_2(o), r_3(o), r_4(o)\} \subseteq N_G^3[b(u)]$. $\square$

6 Application of balanced separators for UDGs

In this section, we show how one can use the above balanced separator theorem for UDGs to develop for them a compact and low delay routing labeling scheme. For this, we combine strategies used in [5, 6, 13].

First, we prove the following important lemma. Let $G = (V, E)$ be an unit disk graph and $S = N_G^3[P1] \cup N_G^3[P2]$ be a balanced separator of G, where $P1$ and $P2$ are (hop-)shortest paths in G. Construct for G two *Breadth First Search trees* (*BFS-trees*) $T1$ and $T2$ as follows. $T1$ is a BFS-tree of G rooted (started) at path $P1$, i.e., $T1 := BFS-tree(G, P1)$. $T2$ is a BFS-tree of G rooted at path $P2$, i.e., $T2 := BFS-tree(G, P2)$. Both trees are (hop-)shortest path trees, rooted at $P1$ and $P2$, respectively.

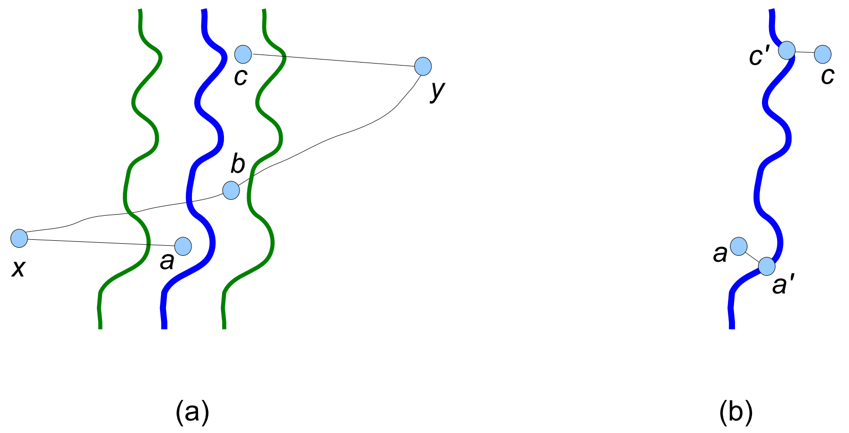


Fig. 8. An illustration to the proof of Lemma 7.

Lemma 7. *Let x, y be two arbitrary vertices of G and $P(x, y)$ be a (hop-)shortest path between x and y in G. If $P(x, y) \cap S \neq \emptyset$, then $d_{T1}(x, y) \leq 3d_G(x, y) + 12$ or $d_{T2}(x, y) \leq 3d_G(x, y) + 12$.*

Proof. Without loss of generality, assume $P(x, y)$ intersects $N_G^3[P1]$ in G. Let $b \in N_G^3[P1] \cap P(x, y)$. Let also $a \in N_G^3[P1]$ be a vertex such that $d_{T1}(x, a)$ is minimum and $c \in N_G^3[P1]$ be a vertex such that $d_{T1}(y, c)$

is minimum. Fig. 8(a) is an illustration, where the central darker curve is $P1$ and the area between two lighter curves represents $N^3_G[P1]$.

Since $T1$ is a (hop-)shortest path tree rooted at $P1$, we conclude $d_G(x,a) \le d_G(x,b)$ and $d_G(c,y) \le d_G(b,y)$. By the triangle inequality, we know that $d_G(a,c) \le d_G(a,x) + d_G(x,b) + d_G(b,y) + d_G(y,c) \le 2d_G(b,x) + 2d_G(y,b) = 2d_G(x,y)$.

Let c' be the vertex on $P1$ that is closest to c (equivalently, to y) in $T1$, let a' be the vertex on $P1$ that is closest to a (equivalently, to x) in $T1$ (see Fig. 8(b)). Since $P1$ is a (hop-)shortest path of G, we have $d_{T1}(a',c') = d_G(a',c') \le d_G(a,c) + d_G(a,a') + d_G(c,c') = d_G(a,c) + 6$.

Now, $d_{T1}(x,y) \le d_{T1}(x,a') + d_{T1}(a',c') + d_{T1}(c',y) = d_{T1}(x,a) + 3 + d_{T1}(a',c') + d_{T1}(c,y) + 3 = d_G(x,a) + d_G(c,y) + d_G(a,c) + 12 \le d_G(x,b) + d_G(b,y) + 2d_G(x,y) + 12 = 3d_G(x,y) + 12$. □

One can construct for an unit disk graph G a *(rooted) balanced decomposition tree* $\mathcal{BT}(G)$ as follows. Find a balanced separator $S = N^3_G[P1] \cup N^3_G[P2]$ for G, which exists according to Theorem 5. If S contains all vertices of G, then $\mathcal{BT}(G)$ is a one node tree. Otherwise, let $G_1, G_2, \dots, G_p$ be the connected components of the graph $G \setminus S$ obtained from G by removing vertices of S. For each graph G_i $(i = 1, \dots, p)$, which is also an UDG, construct a balanced decomposition tree $\mathcal{BT}(G_i)$ recursively, and build $\mathcal{BT}(G)$ by taking S to be the root and connecting the root of each tree $\mathcal{BT}(G_i)$ as a child of S. For a node X of $\mathcal{BT}(G)$, denote by $G(\downarrow X)$ the (connected) subgraph of G induced by vertices $\bigcup\{Y : Y$ is a descendent of X in $\mathcal{BT}(G)\}$ (here we assume that X is a descendent of itself). We know that X is a balanced separator of $G(\downarrow X)$.

It is easy to see that a balanced decomposition tree $\mathcal{BT}(G)$ of an n-vertex m-edge UDG G has depth at most $\log_{3/2} n$. Moreover, a balanced separator (mentioned above) can be found in $O(C + m)$ time, where C is the number of crossings in G, the tree $\mathcal{BT}(G)$ can be constructed in $O((C + m)\log n)$ total time.

Consider now two arbitrary vertices x and y of G and let $S(x)$ and $S(y)$ be the nodes of $\mathcal{BT}(G)$ containing x and y, respectively. Let also $NCA_{\mathcal{BT}(G)}(S(x), S(y))$ be the nearest common ancestor of nodes $S(x)$ and $S(y)$ in $\mathcal{BT}(G)$ and $(X_0, X_1, \dots, X_t)$ be the path of $\mathcal{BT}(G)$ connecting the root X_0 of $\mathcal{BT}(G)$ with $NCA_{\mathcal{BT}(G)}$ $(S(x), S(y)) = X_t$ (in other words, $X_0, X_1, \dots, X_t$ are the common ancestors of $S(x)$ and $S(y)$). Then, any path $P^G_{x,y}$, connecting vertices x and y in G, contains a vertex from $X_0 \cup X_1 \cup \dots \cup X_t$. Let $SP^G_{x,y}$ be a (hop-)shortest path of G connecting vertices x and y, and let X_i be the node of the path $(X_0, X_1, \dots, X_t)$ with the smallest index such that $SP^G_{x,y} \cap X_i \neq \emptyset$ in G. Then, the following lemma holds.

Lemma 8. **[6]** *We have* $d_G(x,y) = d_{G'}(x,y)$, *where* $G' := G(\downarrow X_i)$.

For unit disk graph $G' = G(\downarrow X_i)$ with balanced separator $X_i = N^3_{G'}[P1'] \cup N^3_{G'}[P2']$, consider BFS-trees $T1' := BFS - tree(G', P1')$ and $T2' := BFS - tree(G', P2')$. Since $SP^G_{x,y} \cap X_i \neq \emptyset$, by Lemma 7, there is a tree $T' \in \{T1', T2'\}$ which has the following distance property with respect to those vertices x and y.

Lemma 9. *For those vertices x, y of $G(\downarrow X_i)$, there exists a tree $T' \in \{T1', T2'\}$ such that $d_{T'}(x, y) \leq 3d_{G'}(x, y) + 12 = 3d_G(x, y) + 12$.*

Let now $B^i_1, \ldots, B^i_{p_i}$ be the nodes at depth i of the tree $\mathcal{BT}(G)$. Denote $G^i_j := G(\downarrow B^i_j)$, and let $B^i_j = N^3_{G^i_j}[P1^i_j] \cup N^3_{G^i_j}[P2^i_j]$ be the corresponding balanced separator of G^i_j ($i = 0, 1, ..., depth(\mathcal{BT}(G))$, $j = 1, 2, \ldots, p_i$). For each subgraph $G^i_j := G(\downarrow B^i_j)$ of G ($i = 0, 1, ..., depth(\mathcal{BT}(G))$, $j = 1, 2, \ldots, p_i$), denote by $\mathcal{T}^i_j := \{T1^i_j, T2^i_j\}$ two BFS-trees of graph G^i_j, rooted at paths $P1^i_j$ and $P2^i_j$. Thus, for each G^i_j, we construct two BFS-trees. We call them *local subtrees* of G. Lemma 9 implies

Lemma 10. *Let G be an unit disk graph, $\mathcal{BT}(G)$ be its balanced decomposition tree and $\mathcal{LT}(G) = \{T \in \mathcal{T}^i_j : i = 0, 1, \ldots, depth(\mathcal{BT}(G)), j = 1, 2, \ldots, p_i\}$ be its set of local subtrees. Then, for any two vertices x and y of G, there exists a local subtree $T' \in \mathcal{T}^{i'}_{j'} \subseteq \mathcal{LT}(G)$ such that $d_{T'}(x, y) \leq 3d_G(x, y) + 12$.*

Let $\mathcal{T}^i_j := \{T1^i_j, T2^i_j\}$ be two BFS-trees of graph G^i_j, rooted at paths $P1^i_j$ and $P2^i_j$, respectively. We arbitrarily extend forest $\{T1^i_1, T1^i_2, \ldots, T1^i_{p_i}\}$ ($\{T2^i_1, T2^i_2, \ldots, T2^i_{p_i}\}$) to a spanning tree $T1^i$ (respectively, $T2^i$) of the graph G. Thus, we obtain two spanning trees of G for each level i ($i = 0, 1, \ldots, depth(\mathcal{BT}(G))$) of the decomposition tree $\mathcal{BT}(G)$. Totally, this will result into at most $2 \times depth(\mathcal{BT}(G)) + 2$ spanning trees $\mathcal{T}(G) := \{T1^i, T2^i : i = 0, 1, \ldots, depth(\mathcal{BT}(G))\}$ of the original graph G. Thus, from Lemma 10, we have the following theorem.

Theorem 6. *Any unit disk graph G with n vertices and m edges admits a system $\mathcal{T}(G)$ of at most $2\log_{3/2} n + 2$ collective tree $(3, 12)$-spanners, i.e., for any two vertices x and y in G, there exists a spanning tree $T \in \mathcal{T}(G)$ with $d_T(x, y) \leq 3d_G(x, y) + 12$. Moreover, such a system $\mathcal{T}(G)$ can be constructed in $O((C + m)\log n)$ time, where C is the number of crossings in G.*

Let H be a spanning subgraph of G obtained by taking the union of all spanning trees from $\mathcal{T}(G)$. Clearly, H has at most $2(n-1)(\log_{3/2} n + 1)$ edges and, for any two vertices x and y of G, $d_H(x, y) \leq 3d_G(x, y) + 12$. Thus, we have the following corollary.

Corollary 1. *Any unit disk graph G with n vertices admits a hop $(3, 12)$-spanner with at most $2(n - 1)(\log_{3/2} n + 1)$ edges.*

6.1 Extracting an appropriate tree from $\mathcal{T}(G)$ and approximating distances

Now we will show that one can assign $O(\log^2 n)$ bit labels to vertices of G such that, for any pair of vertices x and y, a tree T in $\mathcal{T}(G)$ with $d_T(x,y) \le 3d_G(x,y)+12$ can be identified in only $O(\log n)$ time by merely inspecting the labels of x and y, without using any other information about the graph. Additionally, a value $\widehat{d}(x,y)$ with $d_G(x,y) \le \widehat{d}(x,y) \le 3d_G(x,y)+12$ can also be computed in $O(\log n)$ time from these labels of x and y.

Associate with each vertex x of G a $5 \times (depth(\mathcal{BT}(G))+1)$ array A_x such that, for each level i of $\mathcal{BT}(G)$, $A_x[1,i]=j$, $A_x[2,i]=d_{T1^i_j}(x,x'_1)$, $A_x[3,i]=d_{T1^i_j}(x'_1,r)$, $A_x[4,i]=d_{T2^i_j}(x,x'_2)$, $A_x[5,i]=d_{T2^i_j}(x'_2,r)$, if there exist local subtrees $T1^i_j$ and $T2^i_j$ in $\mathcal{LT}(G)$ containing vertex x, and $A_x[1,i]=nil$, $A_x[2,i]=A_x[3,i]=A_x[4,i]=A_x[5,i]=\infty$, otherwise (i.e., the depth in $\mathcal{BT}(G)$ of node $S(x)$ containing x is smaller than i). Here x'_1 is a vertex from $P1^i_j$ minimizing $d_{T1^i_j}(x,x'_1)$, x'_2 is a vertex from $P2^i_j$ minimizing $d_{T2^i_j}(x,x'_2)$, r is the root end (common end) of paths $P1^i_j$ and $P2^i_j$. Evidently, each label A_x ($x \in V$) can be encoded using $O(\log^2 n)$ bits and a computation of all labels A_x, $x \in V$, can be performed together with the construction of system $\mathcal{T}(G)$.

Given labels A_x, A_y of vertices x and y, the following procedure will return in $O(\log n)$ time an index $k \in \{0,1,\ldots,depth(\mathcal{BT}(G))\}$ and a number $q \in \{1,2\}$ such that $d_T(x,y) \le 3d_G(x,y)+12$ will hold for $T = T1^k$, if $q=1$, and for $T=T2^k$, if $q=2$.

```
set $k_1 := 0$, $k_2 := 0$;
set $minsum1 := A_x[2,0] + A_y[2,0] + |A_x[3,0] - A_y[3,0]|$;
set $minsum2 := A_x[4,0] + A_y[4,0] + |A_x[5,0] - A_y[5,0]|$;
set $i := 1$;
while $(A_x[1,i] = A_y[1,i] \neq nil)$ and $(i \le \log_{3/2} n)$ do
  if $A_x[2,i] + A_y[2,i] + |A_x[3,i] - A_y[3,i]| < minsum1$
    then set $k_1 := i$ and $minsum1 := A_x[2,i] + A_y[2,i] + |A_x[3,i] - A_y[3,i]|$;
  if $A_x[4,i] + A_y[4,i] + |A_x[5,i] - A_y[5,i]| < minsum2$
    then set $k_2 := i$ and $minsum2 := A_x[4,i] + A_y[4,i] + |A_x[5,i] - A_y[5,i]|$;
  $i := i + 1$;
enddo
if $minsum1 \le minsum2$ then set $k = k_1$ and $q = 1$;
else set $k = k_2$ and $q = 2$;
return $k$, $q$, $j := A_x[1,k]$ and $\widehat{d}(x,y) := \min\{minsum1, minsum2\}$.
```

This procedure simply finds, among all local subtrees containing both x and y, a subtree for which the sum $A_x[2,i]+A_y[2,i]+|A_x[3,i]-A_y[3,i]|$ (or $A_x[4,i]+A_y[4,i]+|A_x[5,i]-A_y[5,i]|$) is minimum.

Assume, without loss of generality, that the procedure above returned $q = 1$. Below we show that indeed $d_{T1^k}(x,y) \leq \widehat{d}(x,y) \leq 3d_G(x,y) + 12$. First note that $\widehat{d}(x,y) = d_{T1^k_j}(x,x'_1) + d_{T1^k_j}(y,y'_1) + |d_{T1^k_j}(x'_1,r) - d_{T1^k_j}(y'_1,r)| = d_{T1^k_j}(x,x'_1) + d_{T1^k_j}(y,y'_1) + d_{T1^k_j}(x'_1,y'_1)$ is an upper bound on $d_{T1^k_j}(x,y)$, by the triangle inequality (where $x'_1 \in P1^k_j$, $y'_1 \in P1^k_j$ with minimum $d_{T1^k_j}(x,x'_1)$, $d_{T1^k_j}(y,y'_1)$; r is the root end of $P1^k_j$).

Let $S(x)$ and $S(y)$ be the nodes of $\mathcal{BT}(G)$ containing vertices x and y, respectively, and let $(B^0, B^1_{j_1}, \ldots, B^t_{j_t})$ be the path of $\mathcal{BT}(G)$ connecting the root B^0 of $\mathcal{BT}(G)$ with $NCA_{\mathcal{BT}(G)}(S(x),S(y)) = B^t_{j_t}$. Since, among local subtrees $T1^0, T2^0, T1^1_{j_1}, T2^1_{j_1}, \ldots, T1^t_{j_t}, T2^t_{j_t}$, the subtree $T1^k_j$ has minimum sum $d_{T1^k_j}(x,x'_1) + d_{T1^k_j}(y,y'_1) + |d_{T1^k_j}(x'_1,r) - d_{T1^k_j}(y'_1,r)| = d_{T1^k_j}(x,x'_1) + d_{T1^k_j}(y,y'_1) + d_{T1^k_j}(x'_1,y'_1)$, by Lemma 9 and by the proof of Lemma 7 (see the last two lines), we conclude $d_{T1^k_j}(x,y) \leq \widehat{d}(x,y) \leq 3d_G(x,y) + 12$, i.e., $d_{T1^k}(x,y) \leq \widehat{d}(x,y) \leq 3d_G(x,y) + 12$ as $T1^k_j$ is a subtree of tree $T1^k$.

Thus, we have the following theorem.

Theorem 7. *The family of n-vertex unit disk graphs admits an $O(\log^2 n)$ bit $(3,12)$-approximate distance labeling scheme with $O(\log n)$ time distance decoder.*

6.2 Routing labeling scheme with bounded hop route-stretch

Existence of collective tree spanners established in Theorem 6 allows us to construct a compact and low delay routing labeling scheme for UDGs. We simply reduce the original problem of routing in UDGs to the problem of routing in trees.

We will need the following result from [9, 26].

Theorem 8. **[9, 26]** *There is a function labeling in $O(n)$ total time the vertices of an n-vertex tree T with labels of up to $O(\log n)$ bits such that given two labels $L(v), L(u)$ of two vertices v, u of T, it is possible to determine in constant time the port number, at v, of the first edge on the path in T from v to u, by merely inspecting the labels of v and u.*

Let now G be an UDG and let $\mathcal{T}(G) = \{T^1, T^2, \ldots, T^\mu\}$ ($\mu \leq O(\log n)$) be a system of μ collective tree $(3,12)$-spanners of G. We can preprocess each tree T^i using the $O(n)$ algorithm from [26] and assign to each vertex v of G a tree-label $L^i(v)$ of size $O(\log n)$ bits associated with the tree T^i. Then, we can form a label $L(v)$ of v of size $O(\log^2 n)$ bits by concatenating the μ tree-labels. We store in $L(v)$ also the string A_v of length $O(\log^2 n)$ bits described in Section 6.1. Thus,

$$L(v) := A_v \circ L^1(v) \circ \ldots \circ L^\mu(v).$$

Now assume that a vertex v wants to send a message to a vertex u. Given the labels $L(v)$ and $L(u)$, v first uses their substrings A_v and A_u to find in $O(\log n)$ time an index i such that for tree $T^i \in \mathcal{T}(G)$, $d_{T^i}(v,u) \leq 3d_G(v,u) + 12$ holds. Then, v extracts from $L(u)$ the substring $L^i(u)$ and forms a header of the message $H(u) := i \circ L^i(u)$. Now, the initiated message with the header $H(u) = i \circ L^i(u)$ is routed to the destination using the tree T^i: when the message arrives at an intermediate vertex x, vertex x using own substring $L^i(x)$ and the string $L^i(u)$ from the header makes a constant time routing decision.

Thus, the following result is true.

Theorem 9. *The family of n-vertex unit disk graphs admits an $O(\log^2 n)$ bit routing labeling scheme. The scheme has hop $(3, 12)$-route-stretch. Once computed by the sender in $O(\log n)$ time, headers never change, and the routing decision is made in constant time per vertex.*

6.3 Extension to routing labeling scheme with bounded length route-stretch

In this section, we show that our results on hop-distance and hop route-stretch can be extended to analogous results on length-distance and length route-stretch.

It is known (see [2, 17, 18, 20]) that, for UDGs, a constant hop route-stretch implies a constant length route-stretch and a constant power route-stretch. In particular, our routing labeling scheme with hop (3,12)-route stretch, according to [2] (see Proposition 1), will have length (6,15)-route stretch.

Proposition 1. **[2]** *Let G be an UDG and x, y be two arbitrary vertices of G. Then, $l_G(x,y) \leq d_G(x,y) \leq 2l_G(x,y) + 1$.*

Below, we show that using our approach a slightly better length route-stretch can be achieved.

First, we prove the following important lemma, which is similar to Lemma 7. Let $G = (V, E)$ be an unit disk graph and $S = N^3_G[P1] \cup N^3_G[P2]$ be a balanced separator of G, where $P1$ and $P2$ are (hop-)shortest paths in G. Denote by $G(S)$ a subgraph of G induced by vertices $S \subseteq V$. Construct for $G' = G(N^3_G[P1])$ and $G'' = G(N^3_G[P2])$ *Breadth First Search trees (BFS-trees)* T_{P1} and T_{P2} as follows. T_{P1} is a BFS-tree of G' rooted (started) at path $P1$, i.e., $T_{P1} := BFS - tree(G', P1)$. T_{P2} is a BFS-tree of G rooted at path $P2$, i.e., $T_{P2} := BFS - tree(G'', P2)$.

Construct also for G two *(length-)shortest path trees (LSP-trees)* $T1$ and $T2$ as follows. $T1$ is a LSP-tree of G rooted (started) at T_{P1}, i.e., $T1 := LSP - tree(G, T_{P1})$. $T2$ is a LSP-tree of G rooted at T_{P2}, i.e., $T2 := LSP - tree(G, T_{P2})$.

Lemma 11. *Let x, y be two arbitrary vertices of G and $P(x,y)$ be a (length-)shortest path between x and y in G. If $P(x,y) \cap S \neq \emptyset$, then $l_{T1}(x,y) \leq 5l_G(x,y) + 13$ or $l_{T2}(x,y) \leq 5l_G(x,y) + 13$.*

Proof. Without loss of generality, assume $P(x,y)$ intersects $N_G^3[P1]$ in G. Let $b \in N_G^3[P1] \cap P(x,y)$. Let also $a \in N_G^3[P1]$ be a vertex such that $l_{T1}(x,a)$ is minimum and $c \in N_G^3[P1]$ be a vertex such that $l_{T1}(y,c)$ is minimum. Fig. 8(a) can also serve as an illustration here.

Since $T1$ is a (length-)shortest path tree rooted at T_{P1}, we conclude $l_G(x,a) \leq l_G(x,b)$ and $l_G(c,y) \leq l_G(b,y)$. By the triangle inequality, we know that $l_G(a,c) \leq l_G(a,x) + l_G(x,b) + l_G(b,y) + l_G(y,c) \leq 2l_G(b,x) + 2l_G(y,b) = 2l_G(x,y)$.

Let c' be the vertex on $P1$ that is closest to c in T_{P1}, let a' be the vertex on $P1$ that is closest to a in T_{P1} (see Fig. 8(b)). Since $P1$ is a (hop-)shortest path of G, we have $d_{T_{P1}}(a',c') = d_G(a',c') \leq d_G(a,c) + d_G(a,a') + d_G(c,c') = d_G(a,c) + 6$.

Now, using the triangle inequality and Proposition 1, we get $l_{T1}(x,y) \leq l_{T1}(x,a') + l_{T1}(a',c') + l_{T1}(c',y) \leq l_{T1}(x,a)+3+l_{T1}(a',c')+l_{T1}(c,y)+3 \leq l_G(x,a)+l_G(c,y)+d_{T_{P1}}(a',c')+6 \leq l_G(x,a)+l_G(c,y)+d_G(a,c)+12 \leq l_G(x,a)+l_G(c,y)+2l_G(a,c)+13 \leq l_G(x,b)+l_G(b,y)+4l_G(x,y)+13 = 5l_G(x,y)+13$. □

Using Lemma 11 and similar arguments as before, we obtain the following results on length-distance and length route-stretch.

Theorem 10. *Any unit disk graph G with n vertices and m edges admits a system $\mathcal{T}(G)$ of at most $2\log_{3/2} n + 2$ collective tree length $(5,13)$-spanners, i.e., for any two vertices x and y in G, there exists a spanning tree $T \in \mathcal{T}(G)$ with $l_T(x,y) \leq 5l_G(x,y) + 13$. Moreover, such a system $\mathcal{T}(G)$ can be constructed in $O((C+m)\log n)$ time, where C is the number of crossings in G.*

Corollary 2. *Any unit disk graph G with n vertices admits a length $(5,13)$-spanner with at most $2(n-1)(\log_{3/2} n + 1)$ edges.*

Theorem 11. *The family of n-vertex unit disk graphs admits an $O(\log^2 n)$ bit $(5,13)$-approximate length-distance labeling scheme with $O(\log n)$ time distance decoder.*

Theorem 12. *The family of n-vertex unit disk graphs admits an $O(\log^2 n)$ bit routing labeling scheme. The scheme has length $(5,13)$-route-stretch. Once computed by the sender in $O(\log n)$ time, headers never change, and the routing decision is made in constant time per vertex.*

7 Conclusion

In this paper, we showed that every unit disk graph G has a balanced separator of form $N_G^3[P1] \cup N_G^3[P2]$, where $P1$ and $P2$ are hop-shortest paths of G. Using this separator theorem, we developed for unit disk graphs routing labeling schemes with $O(log^2n)$ bit labels and hop (3,12)-route-stretch and length (5,13)-route-stretch.

It is interesting to know if those stretch factors can be improved and if every unit disk graph G admits a balanced separator of form $N^1_G[P1] \cup N^1_G[P2]$, where $P1$ and $P2$ are (hop- or length-) shortest paths of G.

References

1. Alber, J., Fiala, J.: Geometric separation and exact solutions for the parameterized independent set problem on disk graphs. Journal of Algorithms 52, 134-151 (2004)
2. Alzoubi, K., Li, X.-Y., Wang, Y., Wan, P.-J., Frieder, O.: Geometric Spanners for Wireless Ad Hoc Networks. IEEE Transactions on Parallel and Distributed Systems 14, 408–421 (2003)
3. Clark, B.N., Colbourn, C.J.: Unit Disk Graphs. Discrete Mathematics 86, 165-177 (1990)
4. Bose, P., Morin, P., Stojmenovic, I., Urrutia, J.: Routing with guaranteed delivery in ad hoc wireless networks. Proceedings of the 3rd International Workshop on Discrete algorithms and Methods for Mobile Computing and Communications, ACM Press, pp.48–55 (1999)
5. Dragan, F.F., Yan, C., Corneil, D.G.: Collective Tree Spanners and Routing in AT-free Related Graphs. Journal of Graph Algorithms and Applications 10, no.2, 97-122 (2006)
6. Dragan, F.F., Yan, C., Lomonosov, I.: Collective tree spanners of graphs. SIAM J. Discrete Math 20, 241–260 (2006)
7. Fonseca, R., Ratnasamy, S., Zhao, J., Ee, C. T., Culler, D., Shenker, S., Stoica, I.: Beacon vector routing: Scalable point-to-point routing in wireless sensornets. Proceedings of the Second USENIX/ACM Syposium on Networked Systems Design and Implementation (NSDI 2005) (2005)
8. Fürer, M., Kasiviswanathan, S.P.: Spanners for Geometric Intersection Graphs. Workshop on Algorithms and Data Structures, pp. 312-324 (2007)
9. Fraigniaud, P., Gavoille, C.: Routing in Trees. ICALP 2001, 757–772 (2001)
10. Gao, J., Guibas, L.J., Hershberger, J., Zhang, L., Zhu, A.: Geometric spanner for routing in mobile networks. Proceedings of the 2nd ACM international symposium on mobile ad hoc networking & computing, October 04-05, 2001, Long Beach, CA, USA
11. Gao, J., Zhang, L.: Well-separated pair decomposition for the unit-disk graph metric and its applications. Proceedings of the thirty-fifth Annual ACM Symposium on Theory of Computing (STOC'03), pp. 483-492 (2003)
12. Giordano, S., Stojmenovic, I.: Position based routing algorithms for ad hoc networks: A taxonomy. In X. Cheng, X. Huang, and D. Du, editors, Ad Hoc Wireless Networking, pages 103–136. Kluwer (2004)
13. Gupta, A., Kumar, A., Rastogi, R.: Traveling with a Pez Dispenser (Or, Routing Issues in MPLS). 42nd IEEE Symposium on Foundations of Computer Science (FOCS'01), pp.148–157 (2001)

14. Karp, B., Kung, H.T.: GPSR: greedy perimeter stateless routing for wireless networks. Proceedings of the 6th ACM/IEEE MobiCom., ACM, pp. 243–254 (2000)
15. Kleinberg, R.: Geographic routing using hyperbolic space. In INFOCOM 2007, pp. 1902–1909 (2007)
16. Kuhn, F., Wattenhofer, R., Zhang, Y., Zollinger, A.: Geometric ad-hoc routing: of theory and practice. Proceedings of the 22nd Annual Symposium on Principles of Distributed Computing, ACM Press, pp. 63–72 (2003)
17. Li, X.-Y.: Topology Control in Wireless Ad Hoc Networks. Book Chapter of "Ad Hoc Networking", IEEE Press, edited by Stefano Basagni, Marco Conti, Silvia Giordano, and Ivan Stojmenovic (2003)
18. Li, X.-Y.: Applications of Computational Geomety in Wireless Ad Hoc Networks. Book Chapter of "Ad Hoc Wireless Networking", Kluwer, edited by XiuZhen Cheng, Xiao Huang, and Ding-Zhu Du (2003)
19. Li, X.-Y., Calinescu, G., Wan, P.-J.: Distributed Construction of a Planar Spanner and Routing for Ad Hoc Wireless Networks. INFOCOM (2002)
20. Li, X.-Y., Wan, P.-J., Wang, Y.: Power efficient and sparse spanner for wireless ad hoc networks. IEEE Int. Conf. on Computer Communications and Networks (ICCCN 2001), pp. 564-567 (2001)
21. Li, X.-Y., Wang, Y.: Geometrical Spanner for Wireless Ad Hoc Networks. Handbook of Approximation Algorithms and Metaheuristics (Editor: Teofilo F. Gonzalez), Chapman & Hall/Crc (2006)
22. Lipton, R.J., Tarjan, R.E.: A Separator Theorem for Planar Graphs. SIAM Journal on Applied Mathematics 36, 177-189, (1979).
23. Peleg, D.: Distributed Computing: A Locality-Sensitive Approach. SIAM Monographs on Discrete Math. Appl., SIAM, Philadelphia (2000).
24. Rao, A., Papadimitriou, C., Shenker, S., Stoica, I.: Geographical routing without location information. Proceedings of MobiCom 2003, pp. 96–108 (2003)
25. Thorup, M.: Compact Oracles for Reachability and Approximate Distances in Planar Digraphs. In 42nd Annual Symposium on Foundations of Computer Science (FOCS), pp. 242-251 (2001)
26. Thorup, M., Zwick, U.: Compact routing schemes. Proceedings of 13th Ann. ACM Symp. on Par. Alg. and Arch (SPAA 2001), pp. 1-10 (2001)
27. Yan, C.: Approximating Distances in Complicated Graphs by Distances in Simple Graphs With Applications. PhD Dissertation, Kent State University, 2007, http://www.ohiolink.edu/etd/send-pdf.cgi/Yan%20Chenyu.pdf?kent1184639623