Balanced Separator Theorem for Unit Disk Graphs with Applications

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

Abstract. In this paper, we establish a novel balanced separator theorem for Unit Disk Graphs (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., \( N^3_{3}[P(s,x) \cup P(s,y)] \)) 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 \( T(G) \) of at most \( 2 \log_2 n + 2 \) spanning trees of \( G \) such that, for any two vertices \( x \) and \( y \) of \( G \), there exists a tree \( T \) in \( T(G) \) 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 \( 2 \log_2 n + 2 \) of its spanning trees. Using these results, we propose a new compact and low delay routing labeling scheme for UDGs. They 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 (\( \equiv \) 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 guarantees delivery but has a low hop delay. It is easy to see that, for UDGs, a constant hop route-stretch implies a constant length route-stretch and a constant power route-stretch.

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 \( \beta \)-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 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 $\ell_1$ 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 guarantees delivery but has a low hop delay. It is easy to see that, for UDGs, a constant hop route-stretch implies 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., $N^3_3[P(s,x) \cup P(s,y)]$) 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 $\sqrt{n}$-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 $T(G)$ of at most $2 \log_2 n + 2$ spanning trees of $G$ such that, for any two vertices $x$ and $y$ of $G$, there exists a tree $T$ in $T(G)$ 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 $2 \log_2 n + 2$ of its spanning trees. 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 are 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 $\beta$-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 $\Gamma$ 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 $\Gamma$ 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 $\Gamma$, 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 than the two labels, regarding the graph from which the vertices were taken.
Similarly, a family \( \Gamma \) 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 \( \Gamma \) 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 \( R \) be a routing scheme and \( R(x, y) \) be a route (path) produced by \( R \) for a pair of vertices \( x \) and \( y \) in a graph \( G \). We say that \( R \) has

- hop \((\alpha, \beta)\)-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 \((\alpha, \beta)\)-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 \((\alpha, \beta)\)-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 \((\alpha, \beta)\)-spanner of \( G \) if \( d_H(x, y) \leq \alpha \cdot d_G(x, y) + \beta \), for any \( x, y \in V \),
- length \((\alpha, \beta)\)-spanner of \( G \) if \( l_H(x, y) \leq \alpha \cdot l_G(x, y) + \beta \), for any \( x, y \in V \),
- power \((\alpha, \beta)\)-spanner of \( G \) if \( p_H(x, y) \leq \alpha \cdot p_G(x, y) + \beta \), 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 \( \mu \) collective tree \((\alpha, \beta)\)-spanners if there is a system \( T(G) \) of at most \( \mu \) spanning trees of \( G \) such that for any two vertices \( x, y \) of \( G \) a spanning tree \( T \in T(G) \) exists such that \( d_T(x, y) \leq \alpha \cdot d_G(x, y) + \beta \).

For a vertex \( v \) of \( G \), the \( k \)th neighborhood of \( v \) in \( G \) is the set \( N^k_G[v] = \{ u \in V : d_G(v, u) \leq k \} \). For a vertex \( v \) of \( G \), the sets \( N_0[v] = N^0_G[v] \) and \( N_0(v) = N_{c}[v] \setminus \{ v \} \) are called the neighborhood and the open neighborhood of \( v \), respectively. For a set \( S \subseteq V \), by \( N^k_G[S] = \bigcup_{v \in S} N^k_G[v] \) we denote the \( k \)th 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 \(|ab| \leq 1\) and \(|cd| \leq 1\). According to the triangle inequality, \(|ao| + |co| > |ac|\) and \(|bo| + |do| > |bd|\). Combining these inequalities, we get \(2 > |ab| + |cd| = |ao| + |bo| + |co| + |do| > |ac| + |bd|\). The latter implies that \(|ac| \leq 1\) or \(|bd| \leq 1\), i.e., \((a, c)\) or \((b, d)\) must be in \( E \). Similarly, one can show that \((a, d)\) or \((c, b)\) must be in \( E \).

\(\square\)
Let \( r \) be an arbitrary but fixed vertex of an UDG \( G = (V, E) \), and \( L_0, L_1, \ldots, L_q \) 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 tree \( T_{\text{orig}} \) rooted at \( r \) as follows: each vertex \( v \in L_i \) (\( i \in \{1, \ldots, q\} \)) chooses a neighbor \( u \) in \( L_{i-1} \) such that \( |vu| \) is minimum (closest neighbor in \( L_{i-1} \)) to be its father in \( T_{\text{orig}} \) (breaking ties arbitrarily). Let \( E(T_{\text{orig}}) \) be the edge set of \( T_{\text{orig}} \). This tree \( T_{\text{orig}} \) 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 \( T_{\text{orig}} \), we assume \( L(a) = L(b) - 1 \). When we refer to any two intersecting edges \((a, b) \) and \((c, d) \) of \( T_{\text{orig}} \) (in that order), we assume that \( L(a) \leq L(c) \).

**Lemma 2.** In \( T_{\text{orig}} \), 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 \( |ab| + |cd| = |ao| + |ob| + |co| + |do| > |ad| + |bc| \), 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 \( \angle oac > \angle lca \) 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 \( T_{\text{orig}} \) 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.

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 \( T_{\text{orig}} \) (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 \( G \), we will use \( E(G) \) to denote the set of its edges and \( V(G) \) to denote the set of its vertices (or elements, if \( V(G) \) 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 \( T_{\text{orig}} \) 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.

In what follows, we will transform tree \( T_{\text{orig}} \) into a special spanning tree \( T \) for the planar graph \( G_p \). Let \( T = T_{\text{orig}} \) initially. For each two intersecting edges \((a, b)\) and \((c, d)\) of \( T_{\text{orig}} \) (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 \((m_{a,b,c,d}, a), (m_{a,b,c,d}, c), (m_{a,b,c,d}, d) \) 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 \(T\) with \(x\) and \(y\) with \(r\), 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 \(P_1 = (x_0 = r, x_1, \ldots, x_k = x)\) and \(P_2 = (y_0 = r, y_1, \ldots, y_l = y)\) 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 \(N_G^1[P_1']\) and \(N_G^1[P_2']\) is a balanced separator for \(G\), 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. 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 \(N_G^3[P_1']\) and \(N_G^3[P_2']\) from \(G = (V, E)\) results in removing a set of edges \(E'\) from \(E\). It is easy to check that, for any edge \(e' \in E'\) there exists an edge \(e' \in E'\) that covers \(e'\). The latter implies that the union of \(N_G^3[P_1']\) and \(N_G^3[P_2']\) 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.
Fig. 4. (left) Handling an intersection between a tree-edge and a non-tree-edge. (right) Handling 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 \( T = T_{\text{orig}} \) 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 \( T_{\text{orig}} \) 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 \( T_{\text{orig}} \) 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 \( T = T_{\text{orig}} \);
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 \( T_{\text{orig}} \) (i.e., \((x, y) \subseteq (v_p, v_h)\))
8. Create an imaginary point \( m_{j,p,h} \) 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 \( m_{j,p,h} \) and edges \((m_{j,p,h}, x), (m_{j,p,h}, y), (m_{j,p,h}, v_k)\);
10. ENDIF
11. ENDFOR
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. 

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 $G_p$.

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 */
We will find the corresponding balanced separator for most four elements in vertices, imaginary points and null points. Generally, each imaginary point or null point is adjacent to at least one real point other than the root element. Let \( \text{weight}(x) \) of paths \( P \) of each imaginary or null point be 0, and get a balanced separator \( S \).

Now we can apply Theorem 1 to \( G_p \) for the element, its previous element and its next element are on the same line, which means its previous element and its next element are not on the same line, which means its previous element and its next element are on another edge of \( G \).

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 least 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 \( P_1 = (x_0 = r, x_1, \ldots, x_k = x) \) and \( P_2 = (y_0 = r, y_1, \ldots, y_l = y) \) 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 \( T_{span} \). 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 \( \{x_f, x_i, y_j\} \) 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 \( 2 \times \) shortest paths of \( G \). We define \( 2 \times \) shortest paths of \( G \) as follows.

**Definition 1.** A path \( P \) of \( G \) is a \( 2 \times \) 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 \( v \in P''_1 \), \( d_{P''_1}(v, r) = d_{\text{T}_{\text{orig}}}(v, r) = d_G(v, r) \) if \( v \) is a real vertex, and \( d_{P''_1}(v, r) = d_{\text{T}_{\text{orig}}}(c(v), r) = d_{\text{T}_{\text{orig}}}(b(v), r) = d_G(c(v), r) = d_G(b(v), r) \) 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 \( d_{\text{T}_{\text{orig}}}(x, r) = d_{\text{T}_{\text{orig}}}(y, r) - 1 \) by the construction of \( T_{\text{orig}} \).
- \( x \) is an imaginary point and \( y \) is an imaginary point: We have \( d_{\text{T}_{\text{orig}}}(c(x), r) = d_{\text{T}_{\text{orig}}}(c(y), r) - 1 \) 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 \( d_{\text{T}_{\text{orig}}}(c(x), r) = d_{\text{T}_{\text{orig}}}(y, r) - 1 \).
- \( x \) is a real vertex and \( y \) is an imaginary point: In this case, \( x \) can only be \( a(y) \) and we have \( d_{\text{T}_{\text{orig}}}(x, r) = d_{\text{T}_{\text{orig}}}(c(y), r) - 1 \) by Lemma 3.

Note also that, for any real vertex \( v \), \( d_{\text{T}_{\text{orig}}}(v, r) = d_G(v, r) \) because \( T_{\text{orig}} \) is a layering tree, and, for any imaginary point \( m \), \( d_{\text{T}_{\text{orig}}}(c(m), r) = d_{\text{T}_{\text{orig}}}(b(m), r) = d_{\text{T}_{\text{orig}}}(d(m), r) - 1 \) 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 \( v_{i+1} \), 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. \( \square \)

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 \( d_{P''}(f(u), f(w)) = |d_{P''}(f(u), r) - d_{P''}(f(w), r)| = |d_G(u, r) - d_G(w, r)| \). By the triangle inequality, \( d_G(u, r) - d_G(w, r) \leq d_G(u, w) \). Combining all this, we get \( d_{P''}(u, w) \leq 2d_G(u, w) \). \( \square \)

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

**Theorem 3.** The union of \( N_G^2[P''_1] \) and \( N_G^3[P''_2] \) 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 \( G \). 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 \( N_G^3[P''_1] \) and \( N_G^3[P''_2] \) is a balanced separator for \( G \) with 2/3-split.
Proof. Since erasing edges $E'_1$ 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.

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 $E'_1$ 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 $E'_1 \setminus A$ for which at least one end is an imaginary point from $P'_1$ or $P'_2$; class $C$ is all edges in $E'_1 \setminus (A \cup B)$ for which at least one end is an imaginary point from $P'_1$ or $P'_2$; class $D$ is all edges in $E'_1 \setminus (A \cup B \cup C)$ (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 $\{r_1(o), r_2(o), r_3(o), r_4(o)\} \subseteq N_G^3[u]$ if $u$ is a real vertex, and $\{r_1(o), r_2(o), r_3(o), r_4(o)\} \subseteq N_G^3[\{u\}]$ 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).

![Fig. 6. Illustrations to the second claim in the proof of Theorem 3.](image)

(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 $|r_1(o)|$, $|r_2(o)|$, $|r_3(o)|$, $|r_4(o)|$, according to PROCEDURE 2. Since $|r_1(o)r_2(o)| = |r_1(o)| + |r_2(o)| \leq 1$ and $|r_3(o)r_4(o)| = |r_3(o)| + |r_4(o)| \leq 1$, we have $|ur_1(o)| \leq 1$, $|ur_2(o)| \leq 1$, $|ur_3(o)| \leq 1$, $|ur_4(o)| \leq 1$. Therefore, $\{r_1(o), r_2(o), r_3(o), r_4(o)\} \subseteq N_G^3[u]$. 

11
(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 from \( r_1(o), r_2(o), r_3(o), r_4(o) \). In addition, we have \( \{a(u), b(u), c(u), d(u)\} \subseteq N^2_G[c(u)] \) by Lemma 1. Therefore, \( \{r_1(o), r_2(o), r_3(o), r_4(o)\} \subseteq N^2_G[c(u)] \).

(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. For an imaginary point \( \{o^i\} \) in Section 5.2), if \( w \) have the following lemma.

**Lemma 6.**

**Proof.**

Therefore, \( \{r_1(o), r_2(o), r_3(o), r_4(o)\} \subseteq N^2_G[c(u)] \).\( \square \)(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. \( \square \)

Theorem 2 and Theorem 3 tell us that there exist two paths \( P_1'' \) and \( P_2'' \) in \( G \) such that they are 2-hop 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 \( N^3_G[P_1'' \) and \( N^3_G[P_2'' \) 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 \( [v_1, \ldots, v_k] \) 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 \( prevP(v_i) \) (\( c(v_i) \) is always adjacent to \( nextP(v_i) \), 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 \( prevP(v_i) \) and \( nextP(v_i) \), 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 \( |prevP(v_i)v'_i| \leq 1, |v'_inextP(v_i)| \leq 1 \) and \( |v'_id(v_i)| \leq 1 \).

**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 \( prevP_1''(v_i) \) and \( nextP_1''(v_i) \) 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 \( v_i \) 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 \( |\text{prev}_P(v_1)v_1'| \leq 1, |v_1'\text{next}_P(v_1)| \leq 1 \) and \( |v_1'd(v_1)| \leq 1 \).

According to our general assumption (no two intersecting edges are on the same line), we conclude \( \text{prev}_P(v_1) = a(v_1) \). In addition, \( \text{next}_P(v_1) \) must lie on segment \( (v_1,d(v_1)) \). It cannot lie on segment \( (v_1,b_1(v_1)) \) since \( \text{prev}_P(v_1), v_1 \) and \( \text{next}_P(v_1) \) are not collinear. If \( c(v_1) \) is chosen as \( v_1' \), we know \( |\text{prev}_P(v_1)v_1'| \leq 1 \) 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, \( |v_1'\text{next}_P(v_1)| \leq 1 \) holds because \( (v_1',d(v_1)) \) is an edge in \( G \) and \( \text{next}_P(v_1) \) lies on segment \( (v_1,d(v_1)) \). If \( b(v_1) \) is chosen as \( v_1' \), \( |\text{prev}_P(v_1)v_1'| \leq 1 \) must hold because \( (\text{prev}_P(v_1),b_1(v_1)) \) is an edge in \( G \). \( |v_1'd(v_1)| \leq 1 \) still holds because, by PROCEDURE 3, \(|a(v_1)c(v_1)| \geq 1 \) and, by Lemma 1, \( |v_1'd(v_1)| \leq 1 \), which implies \( |b(v_1)\text{next}_P(v_1)| \leq 1 \). 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 \( b(v_i) \), \( |\text{prev}_P(v_i)v_i'| \leq 1, |v_i'\text{next}_P(v_i)| \leq 1 \) 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, \( |\text{prev}_P(v_{i+1})v_{i+1}'| \leq 1 \). In addition, \( \text{next}_P(v_{i+1}) \) 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 \( \text{next}_P(v_{i+1}) \) lies on \( (v_{i+1},d(v_{i+1})) \), we have \( |v_{i+1}'\text{next}_P(v_{i+1})| \leq 1 \) 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. \( \text{prev}_P(v_{i+1}) \) is a real vertex. According to our general assumption (no two intersecting edges are on the same line), we conclude \( \text{prev}_P(v_{i+1}) = a(v_{i+1}) \). Therefore, \( |\text{prev}_P(v_{i+1})v_{i+1}'| \leq 1 \) because \( (a(v_{i+1}),b(v_{i+1})) \) is an edge in \( G \). By PROCEDURE 3, \( |\text{prev}_P(v_{i+1})c(v_{i+1})| > 1 \). Then, by Lemma 1, we have \( |v_{i+1}'d(v_{i+1})| \leq 1 \), which implies \( |v_{i+1}'\text{next}_P(v_{i+1})| \leq 1 \) because \( \text{next}_P(v_{i+1}) \) lies on segment \( (v_{i+1},d(v_{i+1})) \).

2. \( \text{prev}_P(v_{i+1}) \) is an imaginary point. Let \( \text{prev}_P(v_{i+1}) \) 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 \( c(v_i) = a(v_{i+1}) \) is chosen as \( v_{i+1}' \), with similar arguments as in the case (1), we conclude that \( |\text{prev}_P(v_{i+1})v_{i+1}'| \leq 1, |v_{i+1}'d(v_{i+1})| \leq 1 \) and \( |v_{i+1}'\text{next}_P(v_{i+1})| \leq 1 \). If \( b(v_i) \) is chosen as \( v_{i+1}' \), by mathematical induction, we know \( |v_{i+1}'d(v_{i+1})| \leq 1 \), where \( d(v_i) = b(v_i+1) \). By PROCEDURE 3, \( |v_{i+1}'c(v_{i+1})| > 1 \). 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 \( |v_{i+1}'\text{next}_P(v_{i+1})| \leq 1 \).

So, it remains to prove that \( (b(v_i),b_{i+1}) \) 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 \( \Delta v_i b(v_i)b(v_{i+1}) \), 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 \( \Delta v_i b(v_i)b(v_{i+1}) \), then \( |b(v_i)c(v_{i+1})| \leq 1 \) must hold, contradicting \( |v_{i+1}'c(v_{i+1})| > 1 \). If \( d(v_{i+1}) \) is inside \( \Delta v_i b(v_i)b(v_{i+1}) \), then \( |d(v_{i+1})| \) must be an edge in \( G \), i.e., \( d_2(c_1, d_1) \leq d_2(c_1, b_1) + 1 \), 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.

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 \( N^3_G[P_1''] \) and \( N^3_G[P_2''] \) is a balanced separator for \( G \) with 2/3-split, i.e., removal of \( N^3_G[P_1''] \cup N^3_G[P_2''] \) 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 \( \{r_1(o), r_2(o), r_3(o), r_4(o)\} \subseteq N^3_G(b(u)) \) if \( u \) is an imaginary point.

In case (2), we need to mention that, for an imaginary point \( u \), both \( \{a(u), b(u), c(u), d(u)\} \subseteq N^3_G(c(u)) \) and \( \{a(u), b(u), c(u), d(u)\} \subseteq N^3_G(b(u)) \) hold. Therefore, \( \{r_1(o), r_2(o), r_3(o), r_4(o)\} \subseteq N^3_G(c(u)) \) and \( \{r_1(o), r_2(o), r_3(o), r_4(o)\} \subseteq N^3_G(b(u)) \).
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 $S = N^3_G[P1] \cup N^3_G[P2]$ be a balanced separator of $G$, where $P1$ and $P2$ and (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.

**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 $N^3_G[P1]$ in $G$. Let $b \in N^3_G[P1] \cap P(x, y)$. Let also $a \in N^3_G[P1]$ be a vertex such that $d_{T1}(x, a)$ is minimum and $c \in N^3_G[P1]$ 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 $d_G(x, a) \leq d_G(x, b)$ and $d_G(c, y) \leq d_G(b, y)$. By the triangle inequality, we know that $d_G(a, c) \leq d_G(a, x) + d_G(x, b) + d_G(b, y) + d_G(y, c) \leq 2d_G(b, y) + 2d_G(y, b) = 2d_G(x, y)$. 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 $d_{T1}(a', c') = d_G(a', c') \leq d_G(a, c) + d_G(a, a') + d_G(c, c') = d_G(a, c) + 6$.

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

One can construct for an unit disk graph $G$ a (rooted) balanced decomposition tree $BT(G)$ 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 $BT(G)$ is a one node tree. Otherwise, let $G_1, G_2, \ldots, G_p$ be the connected components of the graph $G \setminus S$ obtained from $G$ by removing vertices of $S$. For each graph $G_i$ ($i = 1, \ldots, p$), which is also an UDG, construct a balanced decomposition tree $BT(G_i)$ recursively, and build $BT(G)$ by taking $S$ to be the root and connecting the root of each tree $BT(G_i)$ as a child of $S$. For a node $X$ of $BT(G)$, denote by $G(\{X\})$ the (connected) subgraph of $G$ induced by vertices $\{Y : Y$ is a descendent of $X$ in $BT(G)\}$ (here we assume that $X$ is a descendent of itself). We know that $X$ is a balanced separator of $G(\{X\})$.

It is easy to see that a balanced decomposition tree $BT(G)$ 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 $O((C + m) \log n)$ total time, where $C$ is the number of crossings in $G$, the tree $BT(G)$ 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 $BT(G)$ containing $x$ and $y$, respectively. Let also $NCA_{BT(G)}(S(x), S(y))$ be the nearest common ancestor of nodes $S(x)$ and $S(y)$ in $BT(G)$ and $(X_0, X_1, \ldots, X_i)$ be the path of $BT(G)$ connecting the root $X_0$ of $BT(G)$ with $NCA_{BT(G)}(S(x), S(y)) = X_i$ (in other words, $X_0, X_1, \ldots, X_i$ are the common ancestors of $S(x)$ and $S(y)$). Then, any path $P_{x,y}^{G}$ connecting vertices $x$ and $y$ in $G$, contains a vertex from $X_0 \cup X_1 \cup \cdots \cup X_i$. Let $SP_{x,y}^{G}$ be a (hop-)shortest path of $G$ connecting vertices $x$ and $y$, and let $X_i$ be the node of the path $(X_0, X_1, \ldots, X_i)$ with the smallest index such that $SP_{x,y}^{G} \cap X_i \neq \emptyset$ in $G$. Then, the following lemma holds.

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

For unit disk graph $G' = G(\{X_i\})$ with balanced separator $X_i = N_{G'_i}^{3}[P1_i'] \cup N_{G'_i}^{3}[P2_i']$, consider BFS-trees $T1' := BFS - tree(G', P1'_i)$ and $T2' := BFS - tree(G', P2'_i)$. Since $SP_{x,y}^{G} \cap X_i \neq \emptyset$, 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 $G(\{X_i\})$, 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 $B^{1}_{i,1}, \ldots, B^{1}_{i, p_i}$ be the nodes at depth $i$ of the tree $BT(G)$. Denote $G_j^i := G(\{B_j^i\})$, and let $B_j^i = N_{G_j^i}^{3}[P1_j^i] \cup N_{G_j^i}^{3}[P2_j^i]$ be the corresponding balanced separator of $G_j^i$ ($i = 0, 1, \ldots, \text{depth}(BT(G))$, $j = 1, 2, \ldots, p_i$). For each subgraph $G_j^i := G(\{B_j^i\})$ of $G$ ($i = 0, 1, \ldots, \text{depth}(BT(G))$, $j = 1, 2, \ldots, p_i$), denote by $T_j^i := \{T1_j^i, T2_j^i\}$ two BFS-trees of graph $G_j^i$, rooted at paths $P1_j^i$ and $P2_j^i$. Thus, for each $G_j^i$, we construct two BFS-trees. We call them local subtrees of $G$. Lemma 9 implies

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

![Image](image_url)
Let $T^i_j := \{T^i_1, T^i_2\}$ be two BFS-trees of graph $G^i_j$, rooted at paths $P^i_1$ and $P^i_2$, respectively. We arbitrarily extend forest $\{T^i_1, T^i_2, \ldots, T^i_n\}$ of spanning tree $T^i$ (respectively, $T^2$) of the graph $G$. Thus, we obtain two spanning trees of $G$ for each level $i (i = 0, 1, \ldots, \text{depth}(BT(G)))$ of the decomposition tree $BT(G)$. Totally, this will result into at most $2 \times \text{depth}(BT(G))$ + 2 spanning trees $T(G) := \{T^1_i, T^2_i : i = 0, 1, \ldots, \text{depth}(BT(G))\}$ 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 $T(G)$ 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 $T(G)$ with $d_T(x, y) \leq 3d_G(x, y) + 12$. Moreover, such a system $T(G)$ 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 $T(G)$. 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 $T(G)$ 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 $T(G)$ with $d_T(x, y) \leq 3d_G(x, y) + 12$ 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 $\hat{d}(x, y)$ with $d_G(x, y) \leq \hat{d}(x, y) \leq 3d_G(x, y) + 12$ can also be computed in $O(\log n)$ time from these labels of $x$ and $y$.

Associate with each vertex $x$ of $G$ a $5 \times (\text{depth}(BT(G)) + 1)$ array $A_x$ such that, for each level $i$ of $BT(G)$, $A_x[1,i] = j, A_x[2,i] = d_{T^i_j}(x, x'), A_x[3,i] = d_{T^i_1}(x, x'), A_x[4,i] = d_{T^i_2}(x, x'), A_x[5,i] = d_{T^i_2}(x', x'),$ if there exist local subtrees $T^i_1$ and $T^i_2$ in $\lambda T(G)$ containing vertex $x$, and $A_x[1,i] = \text{nil}, A_x[2,i] = A_x[3,i] = A_x[4,i] = A_x[5,i] = \infty$, otherwise (i.e., the depth in $BT(G)$ of node $S(x)$ containing $x$ is smaller than $i$). Here $x'$ is a vertex from $P^i_1$ minimizing $d_{T^i_1}(x, x')$, $x'_2$ is a vertex from $P^i_2$ minimizing $d_{T^i_2}(x, x'_2)$; $r$ is the root end (common end) of paths $P^i_1$ and $P^i_2$. 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 $T(G)$.

Given labels $A_x, A_y$ of vertices $x$ and $y$, the following procedure will return in $O(\log n)$ time an index $k \in \{0, 1, \ldots, \text{depth}(BT(G))\}$ and a number $q \in \{1, 2\}$ such that $d_T(x, y) \leq 3d_G(x, y) + 12$ will hold for $T = T^k_1$, if $q = 1$, and for $T = T^k_2$, if $q = 2$.

```
set k_1 := 0, k_2 := 0;
set \text{minsum}1 := A_x[2,0] + A_y[2,0] + |A_x[3,0] - A_y[3,0]|;
set \text{minsum}2 := 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 \text{nil}) and (i \leq \log_{3/2} n) do
  if A_x[4,i] + A_y[4,i] + |A_x[5,i] - A_y[5,i]| < \text{minsum2}
  i := i + 1;
endo
if \text{minsum}1 \leq \text{minsum}2 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 \hat{d}(x, y) := \min\{\text{minsum}1, \text{minsum}2\}.
```

Assume, without loss of generality, that the procedure above returned \( q = 1 \). Below we show that indeed \( d_{T^1}(x, y) \leq \hat{d}(x, y) \leq 3d_G(x, y) + 12 \). First note that \( \hat{d}(x, y) = d_{T^1}(x, x'_1) + d_{T^1}(y, y'_1) + |d_{T^1}(x'_1, r) - d_{T^1}(y'_1, r)| = d_{T^1}(x, x'_1) + d_{T^1}(y, y'_1) + (d_{T^1}(x'_1, y'_1)) \) is an upper bound on \( d_{T^1}(x, y) \), by the triangle inequality (where \( x'_1 \in P^1_k, y'_1 \in P^1_k \) with minimum \( d_{T^1}(x, x'_1), d_{T^1}(y, y'_1); r \) is the root end of \( P^1_k \)).

Let \( S(x) \) and \( S(y) \) be the nodes of \( \mathcal{B}(G) \) containing vertices \( x \) and \( y \), respectively, and let \( (B^0, B^1_j, \ldots, B^t_j) \) be the path of \( \mathcal{B}(G) \) connecting the root \( B^0 \) of \( \mathcal{B}(G) \) with \( NCA_{\mathcal{B}(G)}(S(x), S(y)) = B^t_j \). Since, among local subtrees \( T^1, T^2, T^1_j, \ldots, T^t_j, T^t_j \), the subtree \( T^t_j \) has minimum sum \( d_{T^t_j}(x, x'_1) + d_{T^t_j}(y, y'_1) + |d_{T^t_j}(x'_1, r) - d_{T^t_j}(y'_1, r)| = d_{T^t_j}(x, x'_1) + d_{T^t_j}(y, y'_1) + d_{T^t_j}(x'_1, y'_1) \), by Lemma 9 and by the proof of Lemma 7 (see the last two lines), we conclude \( d_{T^1_j}(x, y) \leq \hat{d}(x, y) \leq 3d_G(x, y) + 12 \), i.e., \( d_{T^1}(x, y) \leq \hat{d}(x, y) \leq 3d_G(x, y) + 12 \) as \( T^1_k \) is a subtree of \( T^1_k \).

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 \( T(G) = \{ T^1, T^2, \ldots, T^t \} \) \( (\mu \leq O(\log n)) \) be a system of \( \mu \) 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 \( \mu \) 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 \( T^i \in T(G) \), \( d_{T^i}(v, u) \leq 3d_G(v, u) + 12 \) 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 \( L^i(x) \) 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. \cite{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 $G'' = G[N^3_G[P2]]$ the First Search trees (BFS-trees) $T_{P1}$ and $T_{P2}$ as follows. $T_{P1}$ is a BFS-tree of $G'$ rooted (started) at path $P_1$, i.e., $T_{P1} := BFS - tree(G', P1)$. $T_{P2}$ is a BFS-tree of $G$ rooted at path $P_2$, 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 $N^3_G[P1]$ in $G$. Let $b \in N^3_G[P1] \cap P(x, y)$. Let also $a \in N^3_G[P1]$ be a vertex such that $l_{T1}(x, a)$ is minimum and $c \in N^3_G[P1]$ 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 $P1$, we conclude $l_{T1}(x, a) \leq l_{G}(x, b)$ and $l_{G}(c, y) \leq l_{G}(b, y)$. By the triangle inequality, we know that $l_{G}(a, c) \leq l_{G}(a, x) + l_{G}(x, b) + l_{G}(b, y) + l_{G}(c, y) \leq 2l_{G}(b, x) + 2l_{G}(b, y) = 2l_{G}(x, y)$.

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 $l_{T1}(x, y) \leq l_{T1}(x, a') + l_{T1}(a', c') + l_{T1}(c', y) \leq l_{T1}(x, a) + 3 + l_{T1}(a', c) + l_{T1}(c, y) + 3 \leq l_{G}(x, a) + l_{G}(c, y) + d_{T_{P1}}(a, c) + 6 \leq l_{G}(x, a) + l_{G}(c, y) + d_{G}(a, c) + 12 \leq l_{G}(x, a) + l_{G}(c, y) + 2l_{G}(a, c) + 13 \leq l_{G}(x, b) + l_{G}(b, y) + 4l_{G}(x, y) + 13 = 5l_{G}(x, y) + 13$. \hfill \Box

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 $T(G)$ of at most $2\log_{3/2} n + 2$ collective tree $(5, 13)$-spanners, i.e., for any two vertices $x$ and $y$ in $G$, there exists a spanning tree $T \in T(G)$ with $l_T(x, y) \leq 5l_G(x, y) + 13$. Moreover, such a system $T(G)$ can be constructed in $O((C + n) \log n)$ 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 $N^3_G[P1] \cup N^3_G[P2]$, where $P1$ and $P2$ are hop-shortest paths of $G$. Using this separator theorem, we developed for unit disk graphs routing labeling schemes with $O(\log^2 n)$ 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 $G$. 

18
References