Localized Spanners for Ad Hoc Wireless Networks

MIRELA DAMIAN1,* AND SRIRAM V. PEMMARAJU2

1Dept. Comput. Sci., Villanova University, Villanova, PA 19085, USA
E-mail: mirela.damian@villanova.edu
2Dept. Comput. Sci., University of Iowa, Iowa City, IA 52246, USA
E-mail: sriram@cs.uiowa.edu

Received: July 30, 2009. Accepted: Sept. 15, 2009.

We present a new efficient localized algorithm to construct, for any given quasi-unit disk graph \( G = (V, E) \) and any \( \varepsilon > 0 \), a \((1 + \varepsilon)\)-spanner for \( G \) of maximum degree \( O(1) \) and total weight \( O(\omega(MST)) \), where \( \omega(MST) \) denotes the weight of a minimum spanning tree for \( V \). We further show that similar localized techniques can be used to construct, for a given unit disk graph \( G = (V, E) \), a planar \( C_{\text{del}}(1+\varepsilon)(1+\frac{\pi}{2}) \)-spanner for \( G \) of maximum degree \( O(1) \) and total weight \( O(\omega(MST)) \). Here \( C_{\text{del}} \) denotes the stretch factor of the unit Delaunay triangulation for \( V \). Both constructions can be completed in \( O(1) \) rounds of communication, and require each node to know its own coordinates.

Keywords: Wireless ad-hoc networks, unit disk graphs, topology control, spanners, localized algorithms.

1 INTRODUCTION

An ad hoc network is a collection of autonomous devices that can communicate with each other without the aid of any fixed infrastructure. In this paper we assume that each wireless node is equipped with an omnidirectional antenna and is able to communicate with all nodes within its transmission range, using wireless broadcasts. Pairs of wireless nodes not within transmission range of each other communicate through multi-hop wireless links by using intermediate nodes to relay their message. Therefore, each node in an ad hoc network acts as both a host and a router. This extra routing activity consumes a significant amount of energy. Therefore, it is critical to design efficient routing

*Corresponding author. Supported by NSF grant CCF-0728909.
algorithms that minimize the overall energy used in routing. A first step towards this goal is designing efficient topologies on top of which routing is performed. This problem, called topology control [28], seeks to compute and maintain, at each node, only a subset of neighbors that the node communicates with, while maintaining some desirable properties of the communication graph (in addition to connectivity), to be discussed below. Then the energy spent in routing can be optimized by having nodes adjust their transmission power to efficiently select the next hop along the route on the defined topology.

Different topologies optimize different performance metrics. In this paper we focus on properties such as planarity, low degree, low weight, and the spanner property. Planarity is important to various memoryless routing algorithms [5, 18]; low degree at each node is important for balancing out the communication overhead among the wireless nodes; and low weight helps in reducing the total energy consumed by broadcasting sender nodes [33]. Another important property is low interference [6, 17, 32], which we do not address in this paper. We define these properties below, after formally introducing our network model.

Network Model. Let $V$ be a set of $n$ points in the plane representing wireless nodes. Let $G = (V, E)$ be a graph with vertex set $V$ and with edges in $E$ embedded as a straight-line segments in the plane (i.e, $G$ is an Euclidean graph). The graph $G$ is a $\alpha$-quasi unit disk graph ($\alpha$-QUDG) if, for every vertex pair $u, v \in V$, $uv \in E$ if $|uv| \leq \alpha$, and $uv \notin E$ if $|uv| > 1$; here $uv$ represents the edge with endpoints $u$ and $v$, and $|uv|$ is the Euclidean distance between $u$ and $v$. The existence of edges with length in the range $(\alpha, 1]$ is specified by an adversary, thus allowing us to model worst-case situations. If $\alpha = 1$, $G$ is called a unit disk graph (UDG). Quasi unit disk graphs have been proposed as models for ad-hoc wireless networks composed of homogeneous wireless nodes that communicate over a wireless medium without the aid of a fixed infrastructure [2]. Experimental studies show that the transmission range of a wireless node is not perfectly circular and exhibits a transitional region with highly unreliable links [36]. In addition, environmental conditions and physical obstructions adversely affect signal propagation and ultimately the transmission range of a wireless node. The parameter $\alpha$ in the $\alpha$-QUDG model attempts to take into account such imperfections. In this paper we model a wireless network as an UDG or a QUDG, depending on the problem at hand.

Topology Properties. Let $G = (V, E)$ be an Euclidean $\alpha$-QUDG, for fixed $0 < \alpha \leq 1$. A subgraph $H \subseteq G$ has low weight if the total weight $\omega(H)$, defined as the sum of the weights of all edges of $H$, is within a constant factor of the weight of a minimum spanning tree for $G$: $\omega(H) = O(\omega(MST(G)))$. The subgraph $H$ is planar if no two edges, drawn as straight lines, cross each other (i.e., they do not share a point other than an endpoint). For any two points $u, v \in V$, we denote their shortest path (i.e., a path of minimum weight) in $G$ by $s_{\mathcal{P}_G}(u, v)$, and the length of this path by $|s_{\mathcal{P}_G}(u, v)|$. For
fixed $t \geq 1$, $H$ is called a $t$-spanner for $G$ if, for all pairs of vertices $u, v \in V$, $|s_p H(u, v)| \leq t \cdot |s_p G(u, v)|$. The value $t$ is called the stretch factor of $H$. If $t$ is constant, then $H$ is called a length spanner, or simply a spanner. In this paper we seek to construct (planar) spanners of constant maximum degree and low weight, for an arbitrary distribution of ad hoc network nodes in the Euclidean plane.

Communication Model. Due to the limited resources and high mobility of the wireless nodes, it is important to efficiently construct and maintain a spanner in a localized manner. A localized algorithm is a distributed algorithm in which each node $u$ selects all its incident edges based on the information from nodes within a constant number of hops from $u$. Our communication model is the standard synchronous message passing model, which ignores channel access and collision issues. In this communication model, time is divided into rounds. In a round, a node is able to receive all messages sent in the previous round, execute local computations, and send messages to neighbors. Although this communication model is regarded as unrealistic, it is nevertheless interesting because it demonstrates the locality of computations. We measure the communication cost of our algorithms in terms of rounds of communication. Note that a localized algorithm can always be executed in $O(1)$ rounds of communication (by definition).

1.1 Our contribution

In this paper we present the first localized algorithm to construct, for any Euclidean $\alpha$-QUDG $G = (V, E)$ and any $\varepsilon > 0$, a $(1 + \varepsilon)$-spanner for $G$ of maximum degree $O(1)$ and total weight $O(\omega(MST))$, where $\omega(MST)$ denotes the weight of a minimum spanning tree for $V$; here the asymptotic notation hides dependencies on constants $\varepsilon$ and $\alpha$. We further extend our method to construct, for any Euclidean UDG $G = (V, E)$, a planar spanner for $G$ of maximum degree $O(1)$ and total weight $O(\omega(MST))$. The stretch factor of the spanner is bounded above by $C_{del}(1 + \varepsilon)(1 + \frac{\pi}{2})$, where $C_{del}$ is the stretch factor of the unit Delaunay triangulation for $V$ ($C_{del} \leq 2.42$ [22]). This second result resolves an open question posed by Li et al. in [24]. Both constructions can be completed in $O(1)$ communication rounds, and require each node to know its own coordinates.

1.2 Related work

Several excellent surveys on spanners exist [16, 27–29]. For QUDGs, there exist algorithms for constructing $(1 + \varepsilon)$-spanners of maximum degree $O(1)$ and total weight $O(\omega(MST))$, in $O(\log^* n)$ rounds of communication [7, 8]. These algorithms run with distance information only and make no use of coordinate information. In the following we restrict our attention to localized algorithms for constructing spanners in $O(1)$ rounds of communication. We proceed with a discussion on non-planar structures for UDGs first. Existing results are summarized in the first four rows of Table 1.
The Yao graph [35] with an integer parameter $k > 6$, denoted $YG_k$, is defined as follows. At each node $u \in V$, any $k$ equal-separated rays originated at $u$ define $k$ cones. In each cone, pick a shortest edge $uv$, if there is any, and add to $YG_k$ the directed edge $\overrightarrow{uv}$. Ties are broken arbitrarily or by smallest ID. The Yao graph is a spanner with stretch factor $1 - \frac{2 \sin \pi/k}{\pi}$, however its degree can be as high as $n - 1$. To overcome this shortcoming, Li et al. [20] proposed another structure called $YaoYGk$, which is constructed by applying a reverse Yao structure on $YG_k$: at each node $u$ in $YG_k$, discard all directed edges $\overrightarrow{vu}$ from each cone centered at $u$, except for a shortest one (again, ties can be broken arbitrarily or by smallest ID). $YaoYGk$ has maximum node degree $2k$, a constant. However, the tradeoff between degree and spanner property is unclear in the sense that the question of whether $YaoYGk$ is a spanner or not remains open. Both $YG_k$ and $YaoYGk$ have total weight $\Omega(n) \cdot \omega(MST)$. Li et al. [34] further proposed another sparse structure, called $YaoSinkYSk$, that satisfies both the spanner and the bounded degree properties. However, none of these structures has low weight.

We now turn to discussing planar structures for UDGs. The relative neighborhood graph (RNG) [31] and the Gabriel graph (GG) [14] can both be constructed locally, however neither is a spanner [3]. On the other hand, the Delaunay triangulation $Del(V)$\footnote{The circumcircle of each triangle in $Del(V)$ is empty of points in $V$.} is a planar $t$-spanner of the complete Euclidean graph $t$ with vertex set $V$. This result was first proved by Dobkin,\footnote{In a complete Euclidean graph, any two nodes are connected by an edge.}
Friedman and Supowit [12], for \( t = \frac{1+\sqrt{5}}{2} \pi \approx 5.08 \), and was further improved to \( t = \frac{4\sqrt{3}}{9} \pi \approx 2.42 \) by Keil and Gutwin [19]. Das and Joseph [9] generalize these results by identifying two properties of planar graphs, the good polygon and diamond properties, which imply that the stretch factor is bounded above by a constant. Details on these results can be found in Eppstein’s survey [13].

For a given point set \( V \) in the plane, the unit Delaunay triangulation of \( V \), denoted \( \text{UDel}(V) \), is the graph obtained by removing all Delaunay edges from \( \text{Del}(V) \) that are longer than one unit. The unit disk graph induced by a point set \( V \), denoted \( \text{UDG}(V) \), is the graph with vertex set \( V \) and edges connecting \( u, v \in V \) if and only if \( |uv| \leq 1 \). It was shown that \( \text{UDel}(V) \) is a t-spanner of \( \text{UDG}(V) \), with \( t = \frac{4\sqrt{3}}{9} \pi \approx 2.42 \) [22].

Gao et al. [15] present a localized algorithm to build a planar spanner called restricted Delaunay graph (RDG), which is a supergraph of \( \text{UDel}(V) \). The RDG graph, however, may have arbitrarily large degree and weight. Li et al. [22] introduce the notion of a k-localized Delaunay triangle: \( \triangle abc \) is called k-localized Delaunay if the interior of its circumcircle does not contain any node in \( V \) that is a k-neighbor of \( a, b \) or \( c \), and all edges of \( \triangle abc \) are no longer than one unit. The authors describe a localized method to construct, for fixed \( k \geq 1 \), the k-localized Delaunay graph \( \text{LDel}^k(V) \), which contains all Gabriel edges and edges of all k-localized Delaunay triangles. Their construction takes 4 rounds of communication. Araújo and Rodrigues [1] devise a method to compute a planar subgraph \( \text{PLDel}(V) \subseteq \text{LDel}^1(V) \) in one single communication step. Both \( \text{PLDel}(V) \) and \( \text{LDel}^k(V) \), for \( k \geq 1 \), may have arbitrarily large degree and weight.

To bound the degree, several methods apply the ordered Yao structure on top of an unbounded-degree planar structure. This idea was first introduced by Bose et al. in [4], and later refined by Li and Wang in [25, 34]. Since the ordered Yao structure is relevant to our work in this paper as well, we pause to discuss the method for constructing this structure (outlined in Table 2).

The main idea is to define an ordering \( \pi \) of the nodes such that each node \( u \) has a limited number of neighbors (at most 5) who are predecessors in \( \pi \); these predecessors are used to define a small number of open cones centered at \( u \), each of which will contain at most one neighbor of \( u \) in the final structure. To maintain the spanner property of the original graph, a short path connecting all neighbors of \( u \) in each cone is used to replace the edges incident to \( u \) that get discarded from the original graph. Theorem 1 summarizes the important properties of the resulting structure.

**Theorem 1.** If \( G \) is a planar graph with a given embedding in \( \mathbb{R}^2 \), then the output of \( \text{OrderedYao}(G) \) is a planar \((1 + \frac{\pi}{4})\)-spanner for \( G \) of maximum degree 25 [34].

Song et al. [30] apply the ordered Yao structure on top of the Gabriel graph \( \text{GG}(V) \) to produce a planar bounded-degree structure \( \text{OrdYaoGG} \). Their result
Algorithm ORDEREDYAO(\( G = (V, E) \)) [34]

{1. Find an order \( \pi \) for \( V \);}  
Initialize \( i = 1 \) and \( G_i = G \).
Repeat for \( i = 1, 2, \ldots, |V| \)  
   \( \)  
   Remove from \( G_i \) the node \( u \) of smallest degree  
   \( \)  
   (break ties by smallest ID.)  
   Call the remaining graph \( G_{i+1} \).  
   Set \( \pi_u = n - i + 1 \).

{2. Construct a bounded-degree structure for \( G \);}  
Mark all nodes in \( V \) unprocessed. Initialize \( E' \leftarrow \emptyset \) and \( G' = (V, E') \).
Repeat \(|V| \) times  
   \( \)  
   Let \( u \) be the unprocessed node with the smallest order \( \pi_u \).  
   Let \( v_1, v_2, \ldots, v_h \) be the processed neighbors of \( u \) in \( G \) (\( h \leq 5 \)).  
   Shoot rays from \( u \) through each \( v_i \), to define \( h \) sectors centered at \( u \).  
   Divide each sector into fewest open cones of degree at most \( \pi/3 \).  
   For each such open cone \( C_u \) (refer to Figure 1)  
      \( \)  
      Let \( s_1, s_2, \ldots, s_m \in C_u \) be the neighbors of \( u \) geometrically ordered.  
      Add to \( E' \) the shortest \( u s_j \) edge.  
      Add to \( E' \) all edges \( s_j s_{j+1} \), for \( j = 1, 2, \ldots, m - 1 \).  
   Mark node \( u \) processed.

Output \( G' = (V, E') \).

| TABLE 2 |
The ORDEREDYAO method

improves upon the earlier localized structure YaoGG [20], which may not have bounded degree. Both YaoGG and OrdYaoGG are power spanners\(^\dagger\), however neither is a length spanner. A more recent localized method for constructing an almost-planar \((3 + \varepsilon)\)-power spanner of bounded degree for QUDGs is presented in [37]. The output spanner is “almost planar” in the sense that each edge is crossed by \( O(\alpha^4) \) other edges. In contrast, we build an arbitrarily good planar spanner with length stretch factor \( 1 + \varepsilon \). Our focus in this paper is on length spanners (or spanners, for short).

The first efficient localized method to construct a bounded-degree planar spanner was proposed by Li and Wang in [25, 34]. Their method applies the ordered Yao structure on top of \( LDel(V) \) to bound the node degree. The resulting structure, called \( BPS(V) \) (Bounded-Degree Planar Spanner), has degree bounded above by \( 19 + \lceil \frac{2\alpha}{\varepsilon} \rceil \), where \( 0 < \alpha < \frac{\pi}{4} \) is an adjustable parameter. The total communication complexity for constructing \( BPS(V) \) is \( O(n) \) messages, however it may take \( O(n) \) rounds of communication for a node to find its rank in an ordering of \( V \) (a trivial example would be \( n \) nodes

\(^\dagger\)Each edge \( uv \) has weight \( |uv|^{\beta} \), with \( 2 \leq \beta \leq 5 \), as opposed to \( |uv| \).
lined up in increasing order by their ID). The BPS structure does not have low weight [21].

The first localized low-weight planar structure was proposed in [21]. This structure, called RNG', is based on a modified relative neighborhood graph, and satisfies the planarity, bounded-degree and bounded-weight properties. A similar result has been obtained by Li, Wang and Song [24], who propose a family of structures, called Localized Minimum Spanning Trees LMST_k, for k \geq 1. The authors show that LMST_k is planar, has maximum degree 6 and total weight within a constant factor of \omega(MST), for k \geq 2.

However, neither of these low-weight structures satisfies the spanner property. Song et al. [26] are the first to provide a distributed algorithm to construct a planar spanner of bounded-degree and low-weight. Their algorithm uses O(n) total messages, but it may take as many as O(n) rounds of communication, since it relies on some global ordering of O(n) edges.

Constructing low-weight, low-degree planar spanners in few rounds of communication is one of the open problems we resolve in this paper.

2 THE LOS ALGORITHM

We start with some notation and definitions to be used through the rest of the paper. For any weighted edge set E, let \omega(E) denote the total weight of E, defined as the sum of the weights of its constituent edges. For any nodes u and v, let \overrightarrow{uv} denote the edge directed from u to v. Let Cu denote an arbitrary cone with apex u, and let Cu(v) denote the (unique) cone with apex u containing v. For any edge set E and any cone Cu, let E \cap Cu denote the subset of edges in E that are incident to u and lie in Cu. For any graph G = (V, E) and any subset F \subseteq E, let G \setminus F denote the subgraph of G obtained by removing from G all edges that are in F, and G[F] denote the subgraph of G induced by the edge set F. For any node subset U \subseteq V, let G[U] denote the subgraph of G induced by U. If p(u, v) is a path from u to v in G, then |p(u, v)| is used to denote the length of p, defined as the sum of the lengths of all constituent
edges of \( p \). If \(|p(u, v)| \leq t \cdot |uv|\) for some constant \( t > 1 \), \( p(u, v) \) is called a \( t \)-spanner \( uv \)-path.

We assume that each node \( u \) has a unique identifier \( ID(u) \) and knows its coordinates \((x_u, y_u)\). Define the identifier \( ID(\overrightarrow{uv}) \) of a directed edge \( \overrightarrow{uv} \) to be the triplet \((|uv|, ID(u), ID(v))\). We say that \( ID(\overrightarrow{uv}) < ID(\overrightarrow{u'v'}) \) if \( ID(\overrightarrow{uv}) \) comes before \( ID(\overrightarrow{u'v'}) \) in lexicographic order. For an undirected edge \( uv \), define \( ID(uv) = \min\{ID(\overrightarrow{uv}), ID(\overrightarrow{vu})\} \). Note that according to this definition, each edge has a unique identifier.

Let \( H = (V, E_H) \) be an arbitrary subgraph of \( G = (V, E) \). A subset \( L_u \subseteq V \) is an \( r \)-cluster in \( H \) with center \( u \) if, for any \( v \in L_u \), \(|sp_H(u, v)| \leq r\). A set of disjoint \( r \)-clusters \( \{L_{u_1}, L_{u_2}, \ldots\} \) form an \( r \)-cluster cover for \( V \) in \( H \) if they satisfy two properties:

(i) for \( i \neq j \), \(|sp_H(u_i, u_j)| > r \) (the \( r \)-packing property)
(ii) the union \( \cup_i L_{u_i} = V \) (the \( r \)-covering property).

An \( r \)-cluster cover always exists for any \( r > 0 \) and can be easily computed using a greedy method.

A set of node subsets \( V_1, V_2, \ldots, V_h \subseteq V \) is a clique cover for \( V \) if \( G[V_\ell] \) is a clique for each \( \ell = 1, 2, \ldots, h \), and \( \cup_{\ell=1}^h V_\ell = V \).

2.1 The Algorithm

In the following we describe an algorithm called LOS (Localized Optimal Spanner) that takes as input an \( \alpha \)-QUDG \( G = (V, E) \), for some fixed \( 0 < \alpha \leq 1 \), and a value \( \varepsilon > 0 \), and computes a \((1 + \varepsilon)\)-spanner for \( G \) of maximum degree \( O(1) \) and total weight \( O(\omega(MST)) \). The main idea of our algorithm is to compute a particular clique cover \( V_1, V_2, \ldots, V, \) for \( V \), construct a \((1 + \varepsilon)\)-spanner for each \( G[V_\ell] \), then connect these smaller spanners into a \((1 + \varepsilon)\)-spanner for \( G \) using selected Yao edges. In the following we discuss the details of our algorithm.

Let \( 0 < \beta < \frac{\sqrt{2}}{\alpha} \) and \( 0 < \delta < \beta/4 \) be small constants to be fixed later. To compute a clique cover for \( V \), we start by covering the plane with a grid of overlapping square cells of size \( \beta \times \beta \), such that the distance between centers of adjacent cells is \( \beta - \delta \) (see Figure 2). Note that any two adjacent cells define a small band of width \( \delta \) where they overlap. The reason for enforcing this overlap is to ensure that edges not entirely contained within a single grid cell are longer than \( \delta \), i.e., they cannot be arbitrarily small. We identify each grid cell by the coordinates \((i, j)\) of its upper left corner. Any two vertices that lie within the same grid cell are no more than \( \alpha \) distance apart and therefore are connected by an edge in \( G \). This implies that the collection of vertices in each non-empty grid cell can be used to define a clique element of the clique cover. We call a clique cover computed in this manner a \((\beta, \delta)\)-clique cover.
Let $V_1, V_2, \ldots$ be the elements of a $(\beta, \delta)$-clique cover for $V$. Note that, since $\delta < \beta/4$, a node $u$ can belong to at most four subsets $V_\ell$.

Our LOS method consists of 4 main steps. First we construct, for each $G[V_\ell]$, a $(1 + \varepsilon)$-spanner $H_\ell$ of degree $O(1)$ and weight $O(\omega(MST(V_\ell)))$. Various methods for constructing $H_\ell$ exist – for instance, the well-known sequential greedy method produces a spanner with the desired properties [10]. Second, we use the Yao method to generate $(1 + \varepsilon)$-spanner paths between longer edges that span different grid cells. Third, we apply the reverse Yao step to reduce the number of Yao edges incident to each node. Finally, we apply a filtering method to eliminate all but a constant number of edges incident to a grid cell. This fourth step is necessary to ensure that the output spanner has bounded weight. These steps are described in detail in Table 3.

Note that the Yao and reverse Yao steps are restricted to edges in the set $E_0$ no shorter than $\delta$. The next three theorems prove the main properties of the LOS algorithm.

**Theorem 2.** The output $H$ generated by LOS($G, \varepsilon$) is a $(1 + \varepsilon)$-spanner for $G$.

**Proof.** Note that the connectivity property for $H$ is immediately implied by the spanner property, so it suffices to show that $H$ is a spanner. Let $uv \in E$ be arbitrary. If $uv \in G[V_\ell]$ for some $\ell$, then $H_\ell \subseteq H$ contains a $(1 + \varepsilon)$-spanner $uv$-path (since $H_\ell$ is a $(1 + \varepsilon)$-spanner for $G[V_\ell]$). Otherwise, $uv \in E_0$. The proof that $H$ contains a $(1 + \varepsilon)$-spanner $uv$-path is by induction on the ID of edges in $E_0$. Let $uv \in E_0$ be the edge with the smallest ID and assume without loss of generality that $ID(uv) = \overline{ID(uv)}$. Since $ID(uv)$ is smallest, $\overline{uv}$ gets added to $E_Y$ in step 2, and it stays in $E_Y$ in step 3. If $uv \in H$ at the end of step 4, then $sp_H(u, v) = uv$. Otherwise, let $L_x$ and $L_y$ be the r-clusters constructed in step 4, such that $u \in L_x$ and $v \in L_y$ (see Figure 3a). Also let $ab \in E_Y$ be the edge selected in step 4, such that $a \in L_x$ and $b \in L_y$. Note that, since $uv$ exists, $ab$ exists ($a$ and $u$ are not necessarily distinct, and similarly for $v$ and $b$). Since $L_x$ and $L_y$ are both r-clusters, we have that $|sp_H(u, x)| \leq r$, $|sp_H(a, x)| \leq r$, $|sp_H(v, y)| \leq r$ and $|sp_H(b, y)| \leq r$. It follows that $|ux| \leq r$, $|ax| \leq r$, $|vy| \leq r$ and $|by| \leq r$. By the triangle inequality, $|ab| < |uv| + 4r$ and
Algorithm LOS($G = (V, E), \varepsilon$)

[1. Compute a $(1 + \varepsilon)$-spanner cover:]
Fix $0 < \beta < \frac{a}{\sqrt{2}}$ and $0 < \delta < \beta/4$.
Compute a $(\beta, \delta)$-clique cover $V_1, V_2, \ldots$ for $V$.
For each $\ell$, compute a $(1 + \varepsilon)$-spanner $H_\ell$ for $G[V_\ell]$ as in [10].
Initialize $H = \cup \ell H_\ell$.

Let $E_0 = \{uv \in E \mid uv \notin G[V_\ell] \text{ for any } \ell\}$.

[2. Apply Yao on $E_0$:]
Let $k > 8$ be the smallest integer satisfying $\cos \frac{2\pi}{k} - \sin \frac{2\pi}{k} \geq \frac{\delta + 1 + \varepsilon}{(\delta + 1)(1 + \varepsilon)}$.
For each node $u$, divide the plane into $k$ incident equal-size cones.
Initialize $E_Y = \emptyset$.
For each cone $C_u$ such that $E_0 \cap C_u$ is non-empty
Pick the edge $uv \in E_0 \cap C_u$ of smallest ID and add $\overrightarrow{uv}$ to $E_Y$.

[3. Apply reverse Yao on $E_Y$:]
For each cone $C_u$ such that $E_Y \cap C_u$ is non-empty
Discard from $E_Y$ all edges $\overrightarrow{vu} \in E_Y \cap C_u$, but the one of smallest ID.

[4. Select connecting edges from $E_Y$:]
Pick $r$ such that $r \leq \frac{\delta + 1 + \varepsilon}{(\delta + 1)(1 + \varepsilon)} a$, where $\theta = 2\pi/k$.
Compute an $r$-cluster cover $L$ for $V$ in $H$.
For each pair of clusters $L_x, L_y \in L$
Select an arbitrary edge $ab \in E_Y$, with $a \in L_x$ and $b \in L_y$, if it exists.
Add $ab$ to $H$.

Output $H = (V, E_H)$.

‡ This value of $r$ results from our calculations in the proof of Theorem 2.

TABLE 3
The LOS algorithm

Therefore $sp_H(u, v) \leq |ab| + 4r < |uv| + 8r \leq |uv| + \delta \varepsilon$, for any $r \leq \delta \varepsilon/8$ (satisfied by the $r$ values restricted by the algorithm). Now using the fact that $uv \notin G[V_\ell]$ for any $\ell$ implies $|uv| > \delta$ and therefore $sp_H(u, v) < (1 + \varepsilon)|uv|$.

To prove the inductive step, let $uv \in E_0$ be arbitrary, and assume that $H$ contains $(1 + \varepsilon)$-spanner paths between the endpoints of any edge whose $ID$ is lower than $ID(uv)$.

Let $u_1 v_1 \in C_u(v)$ be the Yao edge selected in step 2 of the algorithm; let $u_1 v_1 \in C_{u_1}(u)$ be the Yao-Yao edge selected in step 3 of the algorithm; and let $ab \in E_Y$ be the edge added to $H$ in step 4 of the algorithm, such that $a$ and $u_1$ belong to a same cluster, and similarly for $b$ and $v_1$ (see Figure 3b). The following chains of inequalities hold: $ID(u_1 v_1) \leq ID(uv_1) \leq ID(uv)$ and $|u_1 v_1| \leq |uv_1| \leq |uv|$. Let $u_1'$ be the projection of $u_1$ on $uv_1$, and let $v_1'$ be the projection of $v_1$ on $uv$. Since $u_1 v_1 u_1' \leq \theta$, simple calculations
Theorem 2: (a) Base case. (b) sp

\[ \text{show that } |u'_1v_1| \leq |u_1v_1| \sin \theta \text{ and } |u'_1v_1| \geq |u_1v_1| \cos \theta. \] Similarly, \[ |v'_1v_1| \leq |u_1v_1| \sin \theta \text{ and } |v'_1v_1| \geq |u_1v_1| \cos \theta. \] These along with the triangle inequality property imply

\[
\begin{align*}
|uu_1| &\leq |uv_1| - |u'_1v_1| + |u'_1u_1| \\
&\leq |uv_1| - |u_1v_1| \cos \theta + |u_1v_1| \sin \theta. \\
v_1v_1 &\leq |uv_1| - |u'_1v_1| + |v'_1v_1| \\
&\leq |uv_1| - |u_1v_1| \cos \theta + |u_1v_1| \sin \theta.
\end{align*}
\]

(1)

Since \( \theta < \pi/3 \) and \( |u_1v_1| \leq |uv_1| \), we have that \( |uu_1| < |uv_1| \leq |uv| \). Similarly, \( |v_1v_1| < |uv| \). By the inductive hypothesis, \( H \) contains \((1 + \varepsilon)\)-spanner paths \( sp_H(u, u_1) \) and \( sp_H(v_1, v) \). Let \( P_1 = sp_H(u, u_1) \oplus sp_H(v_1, v) \), of length \( |P_1| \leq (1 + \varepsilon) \cdot (|uu_1| + |v_1v_1|) \); here we use the symbol \( \oplus \) to denote concatenation. Observe that \( P_1 \) is not a path (it is composed of two separate paths). Substituting (1) yields

\[
|P_1| \leq (1 + \varepsilon)|uv| + (1 + \varepsilon)|uv_1|(1 - \cos \theta + \sin \theta) \\
-(1 + \varepsilon)|u_1v_1|(\cos \theta - \sin \theta). \tag{2}
\]

Next we show that the path \( P = P_1 \oplus sp_H(u_1, a) \oplus ab \oplus sp_H(b, v_1) \) is a \((1 + \varepsilon)\)-spanner path from \( u \) to \( v \) in \( H \), thus proving the inductive step. Using the fact that \( |ab| < 4r + |u_1v_1|, |sp_H(u_1, a)| \leq 2r \) and \( |sp_H(b, v_1)| \leq 2r \), we get \( |P| \leq |P_1| + |u_1v_1| + 8r \). Recall that \( u_1v_1 \in E_Y \subseteq E_0 \) and thus \( |u_1v_1| \geq \delta \). Substituting this and \( |uv_1| \leq 1 \) in (2) yields

\[
|P| \leq (1 + \varepsilon)|uv| + \gamma, \text{ with} \\
\gamma = 8r + (1 + \varepsilon) - (1 + \varepsilon)(\cos \theta - \sin \theta)(1 + \delta) - \delta.
\]

Note that the term \( \gamma \) above is non-positive for any \( r \) and \( \theta \) satisfying the conditions of the algorithm. This shows that \( |P| \leq (1 + \varepsilon)|uv| \), thus completing the proof. \( \square \)
Before proving the other two properties of $H$ (bounded degree and bounded weight), we introduce an intermediate lemma. For fixed $c > 0$, call an edge set $F$ $c$-isolated if, for each node $u$ incident to an edge $e \in F$, the closed disk $\text{disk}(u, c)$ centered at $u$ of radius $c$ contains no other endpoints of edges in $F$. This definition is a variant of the isolation property introduced by Das et al. in [11]. The authors show that, if an edge set $F$ satisfies the isolation property, then $\omega(F)$ is within a constant factor of the minimum spanning tree connecting the endpoints of $F$. Here we prove a similar result.

**Lemma 3.** Let $F$ be an edge set that is $c$-isolated, for some constant $c > 0$, and contains only edges no longer than 1. Then $\omega(F) = O(1) \cdot \omega(T)$, where $T$ is the minimum spanning tree connecting the endpoints of edges in $F$.

**Proof.** Let $P$ be a Hamiltonian path obtained by a taking a preorder traversal of $T$. If each edge $uv \in P$ gets associated a weight value $\omega(uv) = |\text{sp}_T(u, v)|$, then it is well-known that $\omega(P) \leq 2\omega(T)$. So in order to prove that $\omega(F)$ is within a constant factor of $\omega(T)$, it suffices to show that $\omega(F) = O(\omega(P))$. Since $F$ is $c$-isolated, the distance between any two vertices in $T$ is greater than $c$ and therefore $\omega(P) \geq (n - 1)c$. On the other hand, no edge in $F$ is greater than 1 and therefore $\omega(F) \leq n$. It follows that $\omega(F) \leq 1 + w(P)/c$, and so $\omega(F) = O(1) \cdot \omega(P)$. \qed

**Theorem 4.** The output $H$ generated by running LOS($G, t$) has maximum degree $O(1)$ and total weight $O(1) \cdot \omega(\text{MST})$.

**Proof.** The fact that $H$ has maximum degree $O(1)$ follows immediately from three observations: (i) each spanner $H_\ell$ constructed in step 1 of the algorithm has degree $O(1)$ [10], (ii) a node $u$ belongs to at most four subgraphs $H_\ell$, and (iii) a node $u$ is incident to a constant number of YaoYao edges (at most $2k$) [20].

We now prove that the total weight for $H$ is within a constant factor of $\omega(\text{MST})$, which is optimal. The main idea is to partition the edge set $\mathcal{E}_H$ into a constant number of subsets, each of which has low weight.

Consider first the $(1 + \epsilon)$-spanners constructed in step 1 of the LOS algorithm. Each $(1 + \epsilon)$-spanner $H_\ell$ corresponds to a grid cell $(i, j)$. Let $F$ denote the set of edges in $\bigcup_\ell H_\ell$. Define the edge set $F_q \subseteq F$ to contain all spanner edges corresponding to those grid cells $(i, j)$ whose indices $i$ and $j$ satisfy the condition

$$(i \mod 3) \times 3 + j \mod 3 = q$$

By this definition, if two edges $e_1, e_2 \in F_q$ lie in two different grid cells, then those two grid cells are separated by at least two other grid cells (see Figure 4a). This further implies that the closest endpoints of $e_1$ and $e_2$ are at distance no smaller than $\alpha$. Also notice that it takes at most 9 subsets...
Next we show that \( \omega(F_q) = O(\omega(T_q)) \) for each \( q = 0, 1, \ldots, 8 \), where \( T_q \) is a minimum spanning tree connecting the endpoints of edges in \( F_q \). To see this, first note that \( F_q \) combines the edges of several low-weight \((1 + \varepsilon)\)-spanners \( H_{q1}, H_{q2}, \ldots \) (constructed in step 1 of LOS) with the property that \( \omega(H_{qi}) = O(\omega(T_{qi})) \), where \( T_{qi} \) is a minimum spanning tree connecting the nodes in \( H_{qi} \). Thus, in order to prove that \( \omega(F_q) = O(\omega(T_q)) \), it suffices to show that each subset contributes its own weight, specifically \( \sum_i \omega(T_{qi}) \leq \omega(T_q) \).

\[
\sum_i \omega(T_{qi}) \leq \omega(T_q)
\]

Our approach is to show that, if Kruskal’s algorithm is employed in constructing \( T_q \) and \( T_{qi} \), then \( T_{qi} \subseteq T_q \), for each \( i \). Since the trees \( T_{qi} \) are all disjoint (separated by at least 2 grid cells), the claim follows. Recall that Kruskal’s algorithm processes edges by increasing length and adds them to \( T_q \) as long as they do not close a cycle. This implies that all edges shorter than \( \alpha \) are processed before edges longer than \( \alpha \). Let \( e \in T_{qi} \) be arbitrary. Then \( |e| \leq \alpha \), since \( T_{qi} \) is restricted to one grid cell of diameter \( \alpha \). If \( e \not\in T_q \), then it must be that \( e \) closes a cycle \( C \) at the time it gets by Kruskal in the construction of \( T_q \). We now show that \( C \) must lie entirely in the grid cell containing \( T_{qi} \). Kruskal guarantees that, at the time \( e \) gets processed, \( C \) contains only edges processed prior to \( e \), and therefore no longer than \( \alpha \). This along with the fact that all edges of \( T_q \) not entirely contained within a cell are longer than \( \alpha \), implies that \( C \) cannot span multiple cells. It follows that \( C \) lies in one single grid cell (namely, the one containing \( e \)). Since \( e \) can be added to \( T_{qi} \),

<table>
<thead>
<tr>
<th>( q = 0 )</th>
<th>( q = 1 )</th>
<th>( q = 2 )</th>
<th>( q = 0 )</th>
<th>( q = 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q = 3 )</td>
<td>( q = 4 )</td>
<td>( q = 5 )</td>
<td>( q = 3 )</td>
<td>( q = 4 )</td>
</tr>
<tr>
<td>( q = 6 )</td>
<td>( q = 7 )</td>
<td>( q = 8 )</td>
<td>( q = 6 )</td>
<td>( q = 7 )</td>
</tr>
<tr>
<td>( q = 0 )</td>
<td>( q = 1 )</td>
<td>( q = 2 )</td>
<td>( q = 0 )</td>
<td>( q = 1 )</td>
</tr>
<tr>
<td>( q = 3 )</td>
<td>( q = 4 )</td>
<td>( q = 5 )</td>
<td>( q = 3 )</td>
<td>( q = 4 )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( s = 0 )</th>
<th>( s = 1 )</th>
<th>( s = 0 )</th>
<th>( s = 1 )</th>
<th>( s = 0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s = 2 )</td>
<td>( s = 3 )</td>
<td>( s = 2 )</td>
<td>( s = 3 )</td>
<td>( s = 2 )</td>
</tr>
<tr>
<td>( s = 0 )</td>
<td>( s = 1 )</td>
<td>( s = 0 )</td>
<td>( s = 1 )</td>
<td>( s = 0 )</td>
</tr>
<tr>
<td>( s = 0 )</td>
<td>( s = 1 )</td>
<td>( s = 0 )</td>
<td>( s = 1 )</td>
<td>( s = 0 )</td>
</tr>
</tbody>
</table>

FIGURE 4
Edge partition (a) 9 subsets (b) 4 subsets.
but not to $T_q$ (due to $C$), it must be that $C$ contains an edge $e' \not\in T_q$, such that $|e'| \leq |e|$. The case $|e'| = |e|$ cannot happen if Kruskal breaks ties in the same manner in both $T_q$ and $T_qi$, so it must be that $|e'| < |e|$. But then we could replace $e$ in $T_q$ by $e'$, resulting in a smaller spanning tree, a contradiction.

It follows that $e \in T_q$ and therefore $T_qi \subseteq T_q$, for each $i$. This shows that $\omega(F_q) = O(\omega(T_q))$, for each $q$. Since there are at most 9 such sets $F_q$ that cover $F$ and since $\omega(T_q) \leq \omega(MST)$, it follows that $\omega(F) = O(\omega(MST))$.

It remains to prove that $\omega(H \setminus F) = O(\omega(MST))$. Let $d \leq 2k$ be the maximum degree of $H \setminus F$. Partition the edges in $H \setminus F$ into no more than $2d \leq 4k$ subsets $E_1, E_2, \ldots$, such that no two edges in $E_i$ share a vertex, for each $i$. We now show that $\omega(E_i) = O(\omega(MST))$, for each $i$. The key observation here is that any two edges $uv, ab \in E_i$ have their closest endpoints – say, $u$ and $a$ – separated by a distance of at least $r/(1 + \epsilon)$. This is because $(1 + \epsilon)|ua| \geq |sp_H(u, a)| > r$; the first part of this inequality follows from the spanner property of $H$, and the second part follows from the fact that $u$ and $a$ are centers of different $r$-clusters (a property ensured by step 4 of the algorithm). This implies that $E_i$ is $r/(1 + \epsilon)$-isolated, and by Lemma 3 we have that $\omega(E_i) = O(\omega(MST))$. This along with the fact that there are no more than 4k sets $E_i$ shows that $\omega(H \setminus F) = O(\omega(MST))$.

Since $\omega(F) = O(\omega(MST))$ and $\omega(H \setminus F) = O(\omega(MST))$, we have that $\omega(H) = O(\omega(MST))$. \hfill $\Box$

2.2 Distributed LOS

In this section we show that the LOS algorithm has an efficient distributed implementation.

**Theorem 5.** The LOS algorithm can be implemented in $O(1)$ rounds of communication using messages that are $O(\log n)$ bits each.

**Proof.** Let $x_u$ and $y_u$ denote the coordinates of a node $u$. At the beginning of the algorithm, each node $u$ broadcasts the information $(\text{ID}(u), x_u, y_u)$ to its neighbors and collects similar information from its neighbors. Each node $u$ determines the grid cell(s) $(i, j)$ that $u$ belongs to from two conditions:

\[
\begin{align*}
  i\alpha/\sqrt{2} \leq x_u &< (i + 1)\alpha/\sqrt{2} \\
  j\alpha/\sqrt{2} \leq y_u &< (j + 1)\alpha/\sqrt{2}
\end{align*}
\]

Similarly, for each neighbor $v$ of $u$, each node $u$ determines the grid cell(s) that $v$ belongs to. Thus step 1 of the algorithm can be implemented in one round of communication: using the collected information, each node $u$ computes the cliques corresponding to those cells $(i, j)$ that $u$ belongs to (at most 4 cliques), then $u$ computes a $(1 + \epsilon)$-spanner for each clique by performing local computations. Note that knowledge of node coordinates is critical to implementing step 1 efficiently.
Step 2 (the Yao step) and step 3 (the reverse Yao step) are inherently local: each node $u$ computes its incident Yao and YaoYao edges based on the information gathered from its neighbors in step 1.

The arguments for step 4 are more subtle. We start by showing that 8 rounds of communication suffice to compute an $r$-cluster cover for $V$ in $H$. Define $U_s$ to be the set of vertices that lie in the grid cells $(i, j)$ such that

$$(i \mod 2) \times 2 + j \mod 2 = s$$

By this definition, two vertices in $U_s$ that lie in different cells are about one grid cell apart (see Figure 4b). Note that $V = \bigcup_{s=0}^{3} U_s$. To compute an $r$-cluster cover for $V$, each node $u$ executes the \textsc{ClusterCover($u, r$)} method presented in Table 4. For simplicity we assume that $r > \delta$, so that two cluster centers that lie in different grid cells are at least distance $r$ apart. However, the \textsc{ClusterCover} method can be easily extended to handle the situation $r \leq \delta$ as well.

No information on existing cluster centers is available in the first iteration of the \textsc{ClusterCover} method (i.e., for $s = 0$). Each node in $U_0$ skips directly to step (D), which implements the standard greedy method for computing an $r$-cluster cover for a given node set ($V_0$ in our case). In the second iteration (i.e., for $s = 1$), some of the clusters computed during the first iteration might be able to grow to incorporate new vertices from $U_1$. This is particularly true for cluster centers that lie in the overlap area of two neighboring cells. Information on such cluster centers is distributed to all relevant nodes in step (E) of the first iteration, then collected in step (A) and forwarded to all nodes in $V_1$ in step (B) of the second iteration. This guarantees that all nodes in $V_1$ have a

\begin{table}[h]
\begin{center}
\begin{tabular}{|l|}
\hline
\textbf{Algorithm} $\textsc{ClusterCover}(u, r)$ \\
\hline
\textbf{Repeat for} $s = 0, 1, 2, 3$ \\
\textbf{If} $u$ belongs to $U_s$ \\
\hspace{1cm} Let $V_s \subseteq U_s$ be the clique containing $u$ (computed in step 1 of LOS). \\
\hspace{1cm} (A) Collect information on cluster centers from neighbors (if any). \\
\hspace{1cm} (B) Broadcast information on existing cluster centers in $V_s$ to all nodes in $V_s$. \\
\hspace{1cm} For each cluster center $w \in V_s$ \\
\hspace{1.5cm} Add to $C_w$ all uncovered nodes $v \in V_s$ such that $d_H(w, v) \leq r$. \\
\hspace{1.5cm} Mark all nodes in $C_w$ covered. \\
\hspace{1cm} (C) For each cluster center $w \in V_s$ \\
\hspace{1.5cm} Pick the uncovered node $w \in V_s$ of highest ID as cluster center. \\
\hspace{1.5cm} Add to $C_w$ all uncovered nodes $v \in V_s$ such that $d_H(v, w) \leq r$. \\
\hspace{1.5cm} Mark all nodes in $C_w$ covered. \\
\hspace{1cm} (D) While $V_s$ contains uncovered nodes \\
\hspace{1.5cm} (E) Broadcast the cluster centers computed in step (D) to all neighbors. \\
\hline
\end{tabular}
\end{center}
\caption{Computing a $\textsc{ClusterCover}$}
\end{table}
consistent view of existing cluster centers in $V_\ell$ at the beginning of step (C). Existing clusters grow in step (C), if possible, and new clusters get created in step (D), if necessary. This procedure shows that it takes no more than 8 rounds of communication to implement step 4 of the LOS algorithm. One final note is that information on a constant number of cluster centers is communicated among neighbors in steps (A), (B) and (D) of the ClusterCover method. This is because only a constant number of $r$-clusters can be packed into a grid cell. So each message is $O(\log n)$ bits long, necessarily so to include a constant number of node identifiers, each of which takes $O(\log n)$ bits.

3 THE PLOS ALGORITHM

In this section we impose our spanner to be planar, at the expense of a bigger stretch factor. This tradeoff is unavoidable, since there are UDGs that contain no $(1 + \varepsilon)$-spanner planar subgraphs, for arbitrarily small $\varepsilon$ (a simple example would be a square of unit diameter).

Our PLOS algorithm consists of 4 major steps. In a first step we construct the unit Delaunay triangulation $U\text{Del}(V)$ using the method described in [23]. Remaining steps use the grid-based idea from Sec. 2.1 to refine the Delaunay structure. Let $V_1, V_2, \ldots$ be a $(\beta, \delta)$-clique cover for $V$, as defined in Sec. 2.1. In step 2 of the PLOS algorithm we apply the OrderedYao method on edge subsets of $U\text{Del}$ incident to each clique $V_i$. The reason for restricting this method to each clique, as opposed to the entire spanner $U\text{Del}(V)$ as in [34], is to reduce the total of $O(n)$ rounds of communication to $O(1)$. The individual degree of each node increases as a result of this alteration, however it remains bounded above by a constant. Intuitively, this is because the unit disk centered at a node $u$ intersects $O(\frac{1}{\beta^2})$ grid cells, meaning that $u$ belongs to a constant number of graphs $Y\text{Del}_i$. Steps 3 and 4 aim to reduce the total weight of the spanner. Step 3 uses a Greedy method to filter out edges with both endpoints in one same clique $V_i$. Step 4 uses clustering to filter out edges spanning multiple cliques. These steps are described in detail in Table 5.

The reason for breaking up step 3 of the algorithm into 8 different rounds is to guarantee that edges used to construct spanner paths in one round are processed by Greedy in a different round, a guarantee that is provided by the following lemma.

\textbf{Lemma 6.} For any $\varepsilon < 2$, the shortest path query $|sp_Q(u, v)| \leq (1 + \varepsilon)|uv|$ in step 3 of the PLOS algorithm involves only those grid cells incident to the cell $A$ containing $uv$.

\textbf{Proof.} For a fixed edge $uv$, the locus of all points $z$ with the property that $|uz| + |zv| \leq (1 + \varepsilon)|uv|$ is a closed ellipse $O$ with focal points $u$ and $v$. Clearly, a point exterior to $O$ cannot belong to a $(1 + \varepsilon)$-spanner path $p(u, v)$ from $u$ to $v$, so it suffices to limit the search for $p(u, v)$ to the interior of $O$. 

Algorithm PLOS($G = (V, E), \varepsilon$)

1. Start with the localized Delaunay structure for $G$:
   Compute $LDel = (V, E_{LDel})$ for $G$ as in [23].
   Fix $0 < \beta \leq \frac{1}{\sqrt{2}}$ and $0 < \delta < \frac{\beta}{4}$.
   Compute a $(\beta, \delta)$-clique cover $V_1, V_2, \ldots$ for $V$.

2. Bound the degree:
   For each clique $V_i$ do the following:
   2.1 Let $E_i \subseteq E_{LDel}$ contain all unit Delaunay edges incident to $V_i$.
   2.2 Execute $YDel_i \leftarrow ORDEREDYAO(G_i = (V, E_i))$ (see Table 2).
   Set $YDel = (V, E_{YDel}) = \bigcup_i YDel_i$.

3. Bound the weight of edges confined to single grid cells:
   Initialize $E_H = \emptyset$ and $H = (V, E_H)$.
   Repeat for $q = 0, 1, \ldots, 8$
   [Use Greedy on non-adjacent grid cells:]
   For each grid cell $A$ of coordinates $(i, j)$ such that
   $$(i \mod 3) \times 3 + j \mod 3 = q$$
   3.1 Let $E_A = E_{YDel} \cap A$ contain all edges in $YDel$ that lie in $A$.
   Let $E_Q = E_{YDel} \setminus E_A$ and $Q = (V, E_Q)$ be the query graph for $E_A$.
   3.2 Sort $E_A$ in increasing order by edge ID.
   For each edge $e = uv \in E_A$, resolve a shortest path query:
   If $SP_Q(u, v) \leq (1 + \varepsilon)|uv|$, then eliminate $uv$ from $YDel$.
   Otherwise, add $uv$ to $H$ and $Q$.

4. Bound the weight of edges spanning multiple grid cells:
   Pick $r$ such that $r \leq \frac{\beta \varepsilon}{4}$ and compute an $r$-cluster cover $L$ for $YDel$.
   For each pair of $r$-clusters $L_x, L_y \in L$.
   Add to $H$ a unique edge $ab \in YDel$, with $a \in L_x$ and $b \in L_y$, if any.

Output $H = (V, E_H)$.

TABLE 5
The PLOS algorithm

Figure 5 (left and middle) shows the search domains for edges corresponding to one diagonal $(uv)$ and one side $(ab)$ of a grid cell. For any grid cell $A$, the union of $A$ and the search ranges for the two diagonals and four sides of $A$ covers the search domain for any edge that lies entirely in $A$ (see Figure 5 right). It can be easily verified that, for $\varepsilon < 2$, the search domain for $A$ fits in the union of $A$ and its eight surrounding grid cells.

We now turn to proving some important properties of the output spanner $H$. We start with two preliminary lemmas (Lemmas 7 and 8).
FIGURE 5
Valid ranges for $|s_{PH}(u,v)| \leq (1+\varepsilon)|uv|$ queries (step 3 of the PLOS algorithm), illustrated for $\varepsilon = 1/2$: query range for edge $uv$ (left), for edge $ab$ (middle), and for the entire grid cell $A$ (right).

Lemma 7. The graph $Y_{Del}$ constructed in step 2 of the PLOS algorithm is a planar $t_1$-spanner for $G$, for any $t_1 \geq C_{del}(\frac{\pi}{2} + 1)$. Moreover, for each edge $ab \in G$, $Y_{Del}$ contains a $t_1$-spanner $ab$-path with edges shorter than $ab$.

Proof. $L_{Del}$ is a planar $C_{del}$-spanner for $G$ [23]. By Theorem 1, $Y_{Del},_i$ is a planar $(\frac{\pi}{2} + 1)$-spanner for $G_i$, for each $i$. These together with the fact that $L_{Del} = \bigcup_i G_i$ show that $Y_{Del}$ is a $t_1$-spanner for $G$.

The fact that $Y_{Del}$ is planar follows an observation in [34] stating that, if a non-Delaunay edge $e \in Y_{Del}$ crosses a Delaunay edge $e'$, then $e'$ must be longer than one unit and does not belong to $Y_{Del}$. More precisely, the following properties hold:

(a) A non-Delaunay edge $ab \in Y_{Del}$ cannot cross a Delaunay edge $uv \in Y_{Del}$. Recall that each non-Delaunay edge $ab \in Y_{Del}$ closes an empty triangle $\triangle abc$ whose other two edges $ac$ and $bc$ are Delaunay edges. Thus, if $ab$ crosses $uv$, then at least one of $ac$ and $bc$ must cross $uv$, contradicting the planarity of $L_{Del}$ (see Fig 6a).

(b) No two non-Delaunay edges $ab, uv \in Y_{Del}$ cross each other. The arguments here are similar to the ones above: if $ab$ and $uv$ intersect,

FIGURE 6
$Y_{Del}$ is planar: edges $ab$ and $uv$ cannot cross.
then at least two of the incident Delaunay edges intersect, contradicting the planarity of $\text{LDel}$ (see Figure 6b).

The second part of the lemma follows from [34].

**Lemma 8.** At the end of each iteration $q$ in step 3 of the PLOS algorithm, $Q$ contains $(1 + \varepsilon)^{q+1}$-spanner paths between the endpoints of any $\text{YDel}$ edge processed in iterations 1 through $q$.

**Proof.** The proof is by induction on $q$. The base case corresponds to $q = 0$. In this case, Greedy ensures that $Q$ contains a $(1 + \varepsilon)$-spanner $uv$-path for each edge $uv$ processed in this iteration. This is because $uv \in \text{YDel}$ either gets added to $Q$ in step 3.1 (and never removed thereafter), or gets queried in step 3.2. To prove the inductive step, consider a particular iteration $q > 0$, and assume that the lemma holds for iterations $\ell = 1 \cdots q - 1$. Again Greedy ensures that $Q$ contains a $(1 + \varepsilon)$-spanner $uv$-path for each edge $uv$ processed in iteration $q$. Consider now an arbitrary edge $uv$ processed in iteration $q$. By the inductive hypothesis, at the end of round $q - 1$, $Q$ contains a path $p(u, v)$ no longer than $(1 + \varepsilon)^q$. However, it is possible that $p(u, v)$ contains edges processed in round $q$ (since Greedy does not restrict $p(u, v)$ to lie entirely in the cell containing $uv$). For each such edge, Greedy ensures the existence of a $(1 + \varepsilon)$-spanner path in $Q$. It follows that, at the end of iteration $q$, $Q$ contains a $(1 + \varepsilon)^{q+1}$-spanner $uv$-path.

**Theorem 9.** The output $H$ generated by $\text{PLOS}(G, \varepsilon)$ is a planar $t$-spanner for $G$, for $t = C_{\text{del}}(1 + \varepsilon)^3(1 + \frac{\varepsilon}{2})$.

**Proof.** Since $H \subseteq \text{YDel}$, by Lemma 7 we have that $H$ is planar. We now show that $H$ is a $t$-spanner for $G$ (which immediately implies that $H$ is connected). The proof is by induction on the length of edges in $H$. The base case corresponds to the edge $uv \in G$ of smallest ID. Clearly $uv \in \text{LDel}$, since $uv$ is a Gabriel edge. Also $uv \in \text{YDel}$, since it has the smallest ID among all edges and therefore it belongs to the Yao structure for $\text{LDel}$. We now distinguish two cases:

(a) There is a grid cell containing both $u$ and $v$. In this case $uv \in H$, since $uv$ is the first edge queried by Greedy in step 3 and therefore it gets added to $H$.

(b) There is no grid cell containing both $u$ and $v$. Let $ab$ be the edge selected in step 4 of the algorithm, such that $u \in L_a$ and $v \in L_b$ (see Figure 3a). Then arguments similar to the ones used for the base case of Theorem 2 show that $sp_H(u, a) \oplus ab \oplus sp_H(b, v)$ is a $(1 + \varepsilon)$-spanner $uv$-path, for any $r \leq \varepsilon\delta/4$. 

"aswin147" — 2009/12/19 — 14:15 — page 323 — #19
This concludes the base case. To prove the inductive step, pick an arbitrary edge $uv \in G$, and assume that $H$ contains $t$-spanner paths between the endpoints of each edge in $G$ of smaller ID. By Lemma 7, $\text{YDel}_1$ contains a path $u = u_0, u_1, \ldots, u_m = v$ such that:

$$\sum_{i=0}^{m} |u_iu_{i+1}| \leq \frac{t}{(1+\varepsilon)^8} |uv| \tag{3}$$

For each edge $u_iu_{i+1} \in \text{YDel}_1$, one of the following cases applies:

(a) There is a grid cell containing both $u_i$ and $u_{i+1}$. In this case, step 3 of the algorithm guarantees that $|\text{sp}_H(u_i, u_{i+1})| \leq (1+\varepsilon)^8 |u_iu_{i+1}|$ (by Lemma 8).

(b) There is no grid cell containing both $u_i$ and $u_{i+1}$. Arguments similar to the ones for the base case show that $|\text{sp}_H(u_i, u_{i+1})| \leq (1+\varepsilon) |u_iu_{i+1}|$.

In either case, $H$ contains a $(1+\varepsilon)^8$-spanner $u_iu_{i+1}$-path. This together with (3) shows that

$$|\text{sp}_H(u, v)| = \sum_{i=0}^{m} |\text{sp}_H(u_i, u_{i+1})|$$

$$\leq (1+\varepsilon)^8 \sum_{i=0}^{m} |u_iu_{i+1}| \leq t|uv|.$$

This completes the proof. \qed

**Theorem 10.** The output $H$ generated by PLOS has maximum degree $O(1)$.

**Proof.** Since $H \subseteq \text{YDel}_1$, it suffices to show that the graph $\text{YDel}_1$ constructed in step 2 of the PLOS algorithm has degree bounded above by a constant. By Theorem 1, the maximum degree of $\text{YDel}_1$ is 25, for each $i$. Also note that unit disk centered at a node $u$ intersects $O\left(\frac{1}{\beta^2}\right)$ grid cells, meaning that $u$ is a neighbor of nodes in $O\left(\frac{1}{\beta^2}\right)$ grid cells and therefore $u$ belongs to a constant number of graphs $\text{YDel}_1$. This implies that the maximum degree of $u$ is $25 \cdot O\left(\frac{1}{\beta^2}\right)$, which is a constant. \qed

**Definition 11.** For any $t \geq t' > 1$, a set $F$ of edges has the $(t', t)$-leapfrog property if, for every subset $S = \{u_1v_1, u_2v_2, \ldots, u_mv_m\}$ of $F$,

$$t' \cdot |u_1v_1| < \sum_{i=2}^{m} |u_iv_i| + t \cdot \left(\sum_{i=1}^{m-1} |v_iu_{i+1}| + |v_mu_m|\right). \tag{4}$$

Das and Narasimhan [10] show the following connection between the leapfrog property and the weight of the spanner.
Lemma 12. Let \( t \geq t' > 1 \). If the line segments \( F \) in \( d \)-dimensional space satisfy the \((t', t)\)-leapfrog property, then \( \omega(F) = O(\omega(MST)) \), where MST is a minimum spanning tree connecting the endpoints of line segments in \( F \).

Theorem 13 (Leapfrog Property). Let \( A \) be an arbitrary grid cell and let \( F \subseteq E_A \) be the set of edges with both endpoints in \( A \) that get added to \( H \) in step 3 of the algorithm. Then \( F \) satisfies the \((1 + \epsilon, t)\)-leapfrog property, for \( t = \left(1 + \epsilon\right)^8(\frac{T}{2} + 1)C_{del} \).

Proof. Consider an arbitrary subset

\[ S = \{u_1v_1, u_2v_2, \ldots, u_mv_m\} \subseteq F \]

To prove inequality (4) for \( S \), it suffices to consider the case when \( u_1v_1 \) is a longest edge in \( S \). Define

\[ S' = \{v_mu_1\} \cup \{v_ju_{j+1} \mid 1 \leq j < m\} \]

Since \( u_j \) and \( v_j \) lie in \( A \) for each \( j \), all edges from \( S' \) lie entirely in \( A \). Let \( ab \in S' \) be arbitrary. If \( |ab| \geq |u_1v_1| \), then inequality (4) trivially holds, so assume that \( |ab| < |u_1v_1| \).

Next we show that \( Q \) contains an \( ab \)-path of length no greater than \( t|ab| \) at the time \( u_1v_1 \) gets queried. If \( ab \in Y_{Del} \), then \( ab \) gets queried in step 3 of PLOS prior to \( u_1v_1 \). This implies that, at the time \( u_1v_1 \) gets queried, \( Q \) contains a path \( P_Q(a, b) \) of length \( |P_Q(a, b)| \leq (1 + \epsilon)^8|ab| \) (by Lemma 8). Consider now the case \( ab \notin Y_{Del} \). By Lemma 7, \( Y_{Del} \) contains a path \( P_{Y_{Del}}(a, b) \) of length

\[ |P_{Y_{Del}}(a, b)| \leq \frac{t}{(1 + \epsilon)^8}|ab| \tag{5} \]

that includes only edges shorter than \( ab \). For each edge \( pq \in P_{Y_{Del}}(a, b) \), \( Q \) contains a path \( P_Q(p, q) \) of length \( |P_Q(p, q)| \leq (1 + \epsilon)^8|pq| \), at the time \( u_1v_1 \) gets queried (by Lemma 8). Thus we have that

\[ |P_Q(a, b)| = \sum_{pq \in P_{Y_{Del}}(a, b)} |P_Q(p, q)| \]

\[ \leq (1 + \epsilon)^8 \sum_{pq \in P_{Y_{Del}}(a, b)} |pq| \]

\[ \leq t|ab| \]

This latter inequality follows from (5).

For \( 1 \leq j < m \), let \( P_j \) be a shortest \( v_ju_{j+1} \)-path in \( Q \), and let \( P_m \) be a shortest \( v_mu_1 \)-path in \( Q \). By the arguments above, such paths exists in \( Q \) at the time \( u_1v_1 \) gets queried, and their stretch factor does not exceed \( t \). Then \( P = P_1 \oplus u_2v_2 \oplus P_2 \oplus u_3v_3 \oplus \ldots \oplus P_m \) is a path from \( u_1 \) to \( v_1 \) in \( Q \), and
ω(P) is no greater than the right hand side of the leapfrog inequality (4). Furthermore, ω(P) > (1 + ε)|u1v1|, otherwise the edge u1v1 would not have been added to H (and Q) in step 3 of the algorithm. These together show that S satisfies the (1 + ε, t)-leapfrog property.

**Theorem 14.** The output H generated by PLOS has total weight O(ω(MST)).

**Proof.** The proof is similar to the proof of Theorem 4 and uses the results of Lemma 12 and Theorem 13.

**Theorem 15.** The PLOS algorithm can be implemented in O(1) rounds of communication.

**Proof.** For simplicity, we restrict our arguments here to ε ≤ 2, however they can be generalized to arbitrary ε values. Computing LDel in step 1 of the algorithm takes at most 4 communication rounds [23]. As shown in the proof of Theorem 4, computing the clique cover in step 1 takes at most 8 rounds of communication. Step 2 of the algorithm is restricted to cliques. A node u belongs to at most 4 cliques. For each such clique, u executes step 2 locally, on the neighborhood collected in step 1.

It takes two more rounds of communication for each node u to learn the topology induced by YDel on its 2-neighborhood. Now note that, for each node u, a farthest node v from u that lies in one of the grid cells incident to the one containing u, lies in the 2-neighborhood of u (see Fig. 5). By Lemma 6, u is able to execute step 3 of the algorithm locally, by checking the existence of short enough paths in its collected 2-neighborhood.

Finally, arguments similar to the ones used in the proof of Theorem 5 show that 8 rounds of communication suffice to execute step 4 of the PLOS algorithm.

4 CONCLUSIONS

We present the first localized algorithm that produces, for any given QUDG G and any ε > 0, a (1 + ε)-spanner for G of maximum degree O(1) and total weight O(ω(MST)), in O(1) rounds of communication. We also present the first localized algorithm that produces, for any given UDG G, a planar O(1)-spanner for G of maximum degree O(1) and total weight O(ω(MST)), in O(1) rounds of communication. Our work leaves open the question of eliminating knowledge of node coordinates without compromising the quality of the spanners.

REFERENCES

Localized Spanners


