Computing Lightweight Spanners Locally

Iyad A. Kanj[1,⋆], Ljubomir Perković[1], and Ge Xia[2]

[1] School of Computing, DePaul University, 243 S. Wabash Ave., Chicago, IL 60604
[2] Department of Computer Science, Lafayette College, Easton, PA 18042

Abstract. We consider the problem of computing bounded-degree lightweight plane spanners of unit disk graphs in the local distributed model of computation. We are motivated by the hypothesis that such subgraphs can provide the underlying network topology for efficient unicasting and multicasting in wireless distributed systems. We present the *first* local distributed algorithm that computes a bounded-degree plane lightweight spanner of a given unit disk graph. The upper bounds on the degree, the stretch factor, and the weight of the spanner, are very small. For example, our results imply a local distributed algorithm that computes a plane spanner of a given unit disk graph U, whose degree is at most 14, stretch factor at most 8.81, and weight at most 8.81 times the weight of a Euclidean Minimum Spanning Tree of $V(U)$.

We show a wider application of our techniques by giving an $O(n \log n)$ time centralized algorithm that constructs bounded-degree plane lightweight spanners of unit disk graphs (which include Euclidean graphs), with the *best* upper bounds on the spanner degree, stretch factor, and weight.

1 Introduction

Efficiency, fault tolerance, scalability, and robustness are central goals in distributed computing. This is especially true for emerging wireless distributed systems such as ad-hoc, mesh, ubiquitous, and sensor networks. Efficiency is critical because wireless devices have typically very limited power. Fault tolerance is required because wireless communication is prone to many errors. Scalability is important because, in practice, wireless systems are often very large. Robustness is necessary to deal with the devices' mobility and the dynamic nature of wireless networks.

Most of the above goals can be achieved, to some extent, with algorithms developed under the *local* distributed computational model, as defined by Linial [15] and Peleg [16]. Assuming that the distributed system is modeled as a graph, a distributed algorithm is said to be *k-local* if, "intuitively", the computation at each point of the graph depends solely on the information about the points at

⋆ The corresponding author. Email: ikanj@cs.depaul.edu. Supported in part by a DePaul University Competitive Research Grant.

G. Taubenfeld (Ed.): DISC 2008, LNCS 5218, pp. 365–378, 2008.

distance (number of edges) at most k from the point (i.e., within k hops from the point). This notion can be formalized as follows [15,16,18]: a distributed algorithm is *k-local* if it runs in at most k synchronous communication rounds for some integer parameter $k > 0$. An algorithm is called *local* if it is k-local for some integer constant k. Efficient local distributed algorithms are naturally fault-tolerant and robust because faults and changes can be handled locally by such algorithms. These algorithms are also scalable because the computation performed by a device is not affected by the total size of the network. Therefore, it is natural to study what problems can or cannot be solved under this model, as did Kuhn, Moscibroda, and Wattenhofer in [12].

We focus our attention in this paper on developing efficient local algorithms for fundamental problems in emerging distributed systems technologies, such as wireless ad-hoc and sensor networks. For these applications, the network is often modeled as a *unit disk graph* (UDG) in the Euclidean plane: the points of the UDG correspond to the mobile wireless devices, and its edges connect pairs of points whose corresponding devices are in each other's transmission range equal to one unit.

The fundamental problem under consideration in this paper is the construction of *lightweight spanners* of a UDG U. The weight of each edge in U is defined to be its Euclidean distance, and the weight of a subgraph of U is the sum of the weights of its edges. It is well-known that a connected UDG contains a Euclidean Minimum Spanning Tree (EMST) of its point-set. A spanning subgraph of U is said to have *low* weight, or to be *lightweight*, if its weight is at most $c \cdot wt(\text{EMST})$ for some constant c. A subgraph H of U is said to be a *spanner* of U if there exists a constant ρ such that: for every two points $A, B \in U$, the weight of a shortest path between A and B in H is at most ρ times the weight of a shortest path between A and B in U. The constant ρ is called the *stretch factor* of H (with respect to U). Lightweight spanners of UDGs are fundamental to wireless distributed systems because they represent topologies that can be used for efficient unicasting *and* broadcasting. Lightweight spanners are also important in computational geometry, and much of the early work on lightweight spanners was done from that perspective under the centralized model of computation [1,2,5,6,7,8,13]. Additional requirements on spanners that have been considered are planarity and bounded degree [2,8,9,14,17]. These requirements are usually motivated by applications in wireless and sensor networks, whose devices have limited resources. For example, the planarity of the topology is often a requirement for efficient routing (see [3,9,11,14,17]).

The specific problem we are thus considering is the design of algorithms (in particular, local distributed algorithms) that construct bounded-degree plane lightweight spanners of unit disk graphs. Levcopoulos and Lingas [13] developed the first centralized algorithm for this problem on Euclidean graphs (i.e., the complete graph on n points in the plane), which are a special case of UDGs. Their $O(n \log n)$ time algorithm, given a rational $\lambda > 2$, produces a plane spanner with stretch factor $(\lambda-1)\cdot C_{del}$ and total weight $(1+\frac{2}{\lambda-2})\cdot wt(\text{EMST})$, where

the constant $C_{del} \approx 2.42$ is the stretch factor of the Delaunay subgraph of the Euclidean graph. Althöfer et al. [1] gave a polynomial time greedy algorithm that constructs a lightweight plane spanner of a Euclidean graph having the same upper bound on the stretch factor and weight as the algorithm by Levcopoulos and Lingas [13]. The degree of the lightweight spanner in both [13] and [1], however, may be unbounded: it is not possible to bound the degree without worsening the stretch factor or the weight. A more recent $O(n \log n)$ time algorithm by Bose, Gudmundsson, and Smid [2] for Euclidean graphs, succeeded in bounding the degree of the plane spanner by 27 but at a large cost: the stretch factor of the obtained plane spanner is approximately 10.02, and its weight is $O(wt(\text{EMST}))$, where the hidden constant in the asymptotic notation is undetermined.

Our contribution with regard to this problem is a centralized algorithm for unit disk graphs, which include Euclidean graphs, that improves the above algorithms. We design a centralized algorithm that, for any integer constant $\Delta \geq 14$ and constant $\lambda > 2$, constructs a plane spanner of a unit disk graph (or a Euclidean graph) having degree at most Δ, stretch factor $(\lambda - 1) \cdot (1 + 2\pi(\Delta \cos \frac{\pi}{\Delta})^{-1}) \cdot C_{del}$, and weight at most $(1 + \frac{2}{\lambda - 2}) \cdot wt(\text{EMST})$ (Theorem 3.1). We can compare our algorithm with the algorithm by Bose, Gudmundsson, and Smid [2] if we let $\Delta = 14$ and $\lambda \approx 2.475$ in Theorem 3.1: we obtain an $O(n \log n)$ time algorithm that, given a unit disk graph (or a Euclidean graph) on n points, computes a plane spanner of the given graph having degree at most 14, stretch factor at most 5.22, and weight at most $5.22 \cdot wt(\text{EMST})$.

We consider next the problem of computing bounded-degree plane lightweight spanners of unit disk graphs using a local distributed algorithm. To the best of our knowledge, the only distributed algorithm for this problem is the algorithm in [4]. While the distributed algorithm in [4] solves the problem for a generalization of unit disk graphs, called quasi-unit ball graphs, in higher dimensional Euclidean spaces, the algorithm is not local (it runs in a poly-logarithmic number of rounds), and the weight and the degree of the spanner are only bounded asymptotically. We note that distributed algorithms for computing lightweight spanners of general graphs have been extensively considered in the literature; see for example [16] for a survey on some of these results. In this paper we show that (Theorem 4.2): for any integer constant $\Delta \geq 14$ and constant $\lambda > 2$, there exists a k-local distributed algorithm, where $k = \lfloor (8/\pi) \cdot (\lambda + 1)^2 \rfloor$, that computes a plane spanner of a given unit disk graph containing a EMST on its point-set, of degree at most Δ, weight at most $(1 + \frac{2}{\lambda - 2}) \cdot wt(\text{EMST})$, and stretch factor $(\lambda - 1)^4 \cdot (1 + 2\pi(\Delta \cos \frac{\pi}{\Delta})^{-1}) \cdot C_{del}$. This is the first local algorithm for this problem. If we set $\Delta = 14$ and $\lambda \approx 2.256$, we obtain a k-local algorithm with k at most 26, that computes a plane spanner of degree at most 14, stretch factor at most 8.81, and weight at most $8.81 \cdot wt(\text{EMST})$, of the given unit disk graph.

The remainder of this paper is organized as follows. We cover the preliminaries in Section 2. In Section 3 we present the centralized algorithm, and in Section 4 we present the local distributed algorithm. In Section 5 we give some further comparisons between our algorithm and the previous ones.

2 Preliminaries

Given a set of points S in the plane, the *Euclidean graph* E on S is defined to be the complete graph whose point-set is S. The *unit disk graph* U on S is the subgraph of E with the same point-set as E, and such that AB is an edge of U if and only if $|AB| \leq 1$, where $|AB|$ is the Euclidean length of edge AB. We assume in this paper that the unit disk graph U is connected. We define the *weight* of an edge AB to be the Euclidean distance between points A and B, that is $wt(AB) = |AB|$. For a subgraph $H \subseteq E$, we denote by $V(H)$ and $E(H)$ the set of vertices and the set of edges of H, respectively, and by $wt(H)$ the sum of the weights of all the edges in H, that is, $wt(H) = \sum_{XY \in E(H)} wt(XY)$. The *length* of a path P (resp. cycle C) in a subgraph $H \subseteq E$, denoted $|P|$ (resp. $|C|$), is the number of edges in P (resp. C). A point B is said to be a *k-neighbor* of A in a subgraph $H \subseteq E$, if there exists a path P from A to B in H satisfying $|P| \leq k$.

Each of the synchronous communication rounds in a local distributed algorithm consists of two phases: phase 1, in which every point receives messages sent to it in the preceding phase, and phase 2, in which every point sends messages to its neighbors. The local computation in a round occurs between the two phases. Since our focus is on wireless systems, we will assume that a message broadcast by a point in U will be received by all its neighbors.

The local distributed algorithm we develop in this paper constructs a subgraph of U and takes two steps. In the first step, all points learn about their k-hop neighbors using a local distributed algorithm. In the second step, each point runs a local computation to make a decision on what incident edges to select in the final spanner (no messages are exchanged in this step). A *k-local k-neighborhood algorithm* is a k-local algorithm in which each point learns about the coordinates of its k-hop neighbors. A basic k-local k-neighborhood algorithm runs as follows. In the first round, every point broadcasts its ID and coordinates to its neighbors in U. In the remaining $k-1$ rounds, every point broadcasts the ID and coordinates of every point it learned about in the previous round.

Let G be a plane graph and let T be a spanning tree of G. Call an edge $e \in E(T)$ a *tree edge* and an edge $e \in E(G) - T$ a *non-tree edge*. Every non-tree edge induces a unique cycle in the graph $T + e$ called the *fundamental cycle* of e. Since T is embedded in the plane, we can talk about the *fundamental region* of e, which is the closed region in the plane enclosed by the fundamental cycle of e (other than the outer face of $T + e$).

Definition 2.1. Define a relationship $\preceq$ on the set $E(G)$ as follows. For every edge e, $e \preceq e$. For two edges e and e' in $E(G)$, $e \preceq e'$ if and only if e is contained in the fundamental region of e'.

It is not difficult to verify that $\preceq$ is a partial order relation on $E(G)$, and hence $(E(G), \preceq)$ is a partially ordered set (POSET). Note that any two distinct tree edges are not comparable by $\preceq$, and that every tree edge is a minimal element in $(E(G), \preceq)$. Therefore, we can topologically sort the edges in $E(G)$ to form a

list $\mathcal{L} = \langle e_1, \ldots, e_r \rangle$, in which no non-tree edge appears before a tree edge, and such that if $e_i \preceq e_j$ then e_i does not appear after e_j in $\mathcal{L}$.

Lemma 2.1. *Let e_i be a non-tree edge. Then there exists a unique face F_i in G such that every edge e_j of F_i satisfies $e_j \preceq e_i$.*

Proof. Let F_i be the face of G containing e_i and residing in the fundamental region of e_i, and let e_j be an edge on F_i. Since e_j is on F_i, e_j is contained in the fundamental region of e_i. By the definition of $\preceq$, we have $e_j \preceq e_i$. This shows the existence of such a face F_i.

To prove the uniqueness of F_i, suppose that there is another distinct face F_i' with the above properties. Since every edge e_j on F_i' satisfies $e_j \preceq e_i$, every edge on F_i' is contained in the fundamental region of e_i, and hence the whole face F_i' is contained in the fundamental region of e_i. This means that there are two distinct faces containing e_i that are enclosed within the fundamental cycle of e_i. This contradicts the planarity of G.

We will call the unique face associated with a non-tree edge e_i, described in Lemma 2.1, the *fundamental face* of e_i.

The following result is a consequence of the proof of Theorem 2 in [1]. A similar, but less general result, was also proved earlier by Levcopoulos and Lingas [13]. A different proof can also be found in [10].

Theorem 2.1. ([1])

(i) Let G be a connected weighted planar graph with nonnegative weights satisfying the following property: for every cycle C in G and every edge $e \in C$, $wt(C) \geq \lambda \cdot wt(e)$ for some constant $\lambda > 2$. Then $wt(G) \leq (1 + \frac{2}{\lambda - 2}) \cdot wt(T)$, where T is a MST of G.

(ii) Let G be a connected weighted plane graph with nonnegative weights, and let T be a spanning tree in G. Let $\lambda > 2$ be a constant. Suppose that for every edge $e \in E(G) - T$ we have $wt(F_e) \geq \lambda \cdot wt(e)$, where F_e is the boundary cycle of the fundamental face of e in G. Then $wt(G) \leq (1 + \frac{2}{\lambda - 2}) \cdot wt(T)$.

3 The Centralized Algorithm

In this section we present a centralized algorithm that constructs a bounded-degree plane lightweight spanner of U.

Kanj and Perlović [9] gave an $O(n \log n)$ time centralized algorithm that, given a Euclidean graph E on a set of n points in the plane, and an integer parameter $\Delta \geq 14$, constructs a plane spanner G' of E containing a EMST of $V(E)$, of degree at most Δ, and of stretch factor $\rho = (1 + 2\pi(\Delta \cos \frac{\pi}{\Delta})^{-1}) \cdot C_{del}$, where $C_{del} \approx 2.42$ is the stretch factor of the Delaunay subgraph of E. This result can be extended to unit disk graphs:

Lemma 3.1. *For any $\Delta \geq 14$, the subgraph G'_U of the spanner G' described in [9], consisting of those edges in G' of weight at most 1, is a plane spanner*

of the unit disk graph U on $V(E)$ of degree at most Δ, and of stretch factor $\rho = (1 + 2\pi(\Delta \cos \frac{\pi}{\Delta})^{-1}) \cdot C_{del}$ (with respect to U). Moreover, G'_U contains a EMST of $V(U)$.

Proof. We need to verify that the subgraph G'_U of G' obtained by removing every edge of weight greater than 1 from G', is also a spanner of the unit disk graph U satisfying the same properties that G' satisfies with respect to Euclidean graph E.

It was shown in [9] that the spanner G' satisfies the property that, for every edge $AB \in E$ that is not in G', there exists a path P_{AB} from A to B in G' of weight at most $\rho \cdot wt(AB)$, and such that AB has maximum weight among all edges on P_{AB} (see Theorem 2.10 in [9]). Since the unit disk graph U is the subgraph of E consisting precisely of those edges in E of weight at most 1, by discarding from G' every edge of weight greater than 1, we obtain a subgraph G'_U of U that is plane and of degree at most Δ. Since U is connected, U contains a EMST of $V(U)$, and hence every edge in the EMST has weight at most 1. Since G' contains a EMST of $V(U)$, it follows from the preceding statement, and from the definition of G'_U, that G'_U contains a EMST of $V(U)$ as well. If an edge $AB \in U$ is not in G', from the properties of the spanner G' described above, there exists a path P_{AB} in G' whose weight is at most $\rho \cdot wt(AB)$, and on which AB is the edge of maximum weight. From the definition of G'_U, the path P_{AB} is also in G'_U. It follows that the same algorithm described in [9] computes a plane spanner G'_U of the unit disk graph U, of degree at most Δ, and of stretch factor $(1 + 2\pi(\Delta \cos \frac{\pi}{\Delta})^{-1}) \cdot C_{del}$, for any integer parameter $\Delta \geq 14$. □

The spanner G'_U, however, may not be of light weight. Therefore, we need to discard edges from G'_U so that the resulting subgraph is of light weight, while at the same time not affecting the stretch factor of G'_U by much. To do so, since G'_U is a plane graph containing a EMST of $V(U)$, we would like to employ part (ii) of Theorem 2.1. However, there is one technical problem: the fundamental faces of G'_U may not satisfy the condition in part (ii) of Theorem 2.1, namely that the weight of every fundamental face F_e of a non-EMST edge e in G'_U satisfies $wt(F_e) \geq \lambda \cdot wt(e)$ ($\lambda > 2$ is a constant). We will show next how to prune the set of edges in G'_U so that this condition is satisfied.

Let T be a EMST of $V(U)$ contained in G'_U. As described in Section 2, we can order the non-tree edges in G'_U with respect to the partial order $\preceq$ described in Definition 2.1. Let $\mathcal{L}' = \langle e_1, e_2, \ldots, e_s \rangle$ be the sequence of non-tree edges in G'_U sorted in a non-decreasing order with respect to the partial order $\preceq$. Note that, by the definition of the partial order $\preceq$, if we add the edges in $\mathcal{L}'$ to T in the respective order they appear in $\mathcal{L}'$, once an edge e_i is added to form a fundamental face in the partially-grown graph, this fundamental face will remain a face in the resulting graph after all the edges in $\mathcal{L}'$ have been added to T. That is, the face will not be affected (i.e., changed/split) by the addition of any later edge in this sequence.

Given a constant $\lambda > 2$, to construct the desired lightweight spanner G, we first initialize G to the EMST T. We consider the non-tree edges of G'_U in the order that they appear in $\mathcal{L}'$. Inductively, suppose that we have processed the

edges $e_1, \ldots, e_{i-1}$ in $\mathcal{L}'$. To process edge e_i, let F_i be the fundamental face of e_i in $G + e_i$. If $wt(F_i) > \lambda \cdot wt(e_i)$, we add e_i to G; otherwise, e_i is not added to G. This completes the description of the construction process. Let G be the resulting graph at the end of the construction process.

Lemma 3.2. *Given the set of n points $V(E)$ in the plane, the graph G can be constructed in $O(n \log n)$ time.*

Proof. We first describe how to compute the sequence $\mathcal{L}'$.

The bounded-degree plane spanner G' of E can be constructed in $O(n \log n)$ time [9], and obviously so can G'_U. Since every point in G'_U has bounded degree, and since G'_U is a geometric plane graph, in $O(1)$ time we can compute a rotation system for the points in G'_U (for example, for every point in G'_U, we can list its incident edges in clockwise order). Moreover, since G'_U has $O(n)$ edges, the EMST T contained in G'_U can be computed in $O(n \log n)$ time by a standard MST algorithm. Now using the rotation system of G'_U, we can traverse the edges on the boundary face of G'_U. As we traverse these edges, we remove them from the graph and push the non-tree (with respect to T) edges into a stack; we also remove any isolated points resulting from this process. Note that the non-tree edges on the outer face of G'_U are the maximal edges with respect to the ordering $\preceq$. We repeat this process until G'_U is empty, and at that point, the stack contains the sequence of non-tree edges, sorted according to the partial order $\preceq$; this stack constitutes the list $\mathcal{L}'$. Clearly, this process can be carried out in $O(n)$ time.

After computing $\mathcal{L}'$, we initialize G to the EMST T. As we consider the edges in $\mathcal{L}'$, when we add an edge e in $\mathcal{L}'$ to form a fundamental face F_e in $G + e$, we need to check whether the fundamental face F_e satisfies the condition $wt(F_e) > \lambda \cdot wt(e)$. To do so, we need to traverse the edges on F_e. If e is not subsequently added to G, we might need to traverse some edges on F_e multiple times when we later consider edges that are larger than e in the ordering $\preceq$. To avoid this problem, we can do the following. If we decide to add an edge to G, we add this edge and mark it as a "real" edge of G. On the other hand, if e is not to be added to G, we still add e to G but we mark it as a "virtual" edge of G, and assign it a weight equal to the weight of its fundamental face. The graph G will consist of the tree T plus the set of edges that were marked as real edges. This way each edge in G is traversed at most twice (as every edge appears in at most two faces), and the running time is kept $O(n)$.

It follows that G can be constructed in $O(n \log n)$ time, and the proof is complete. □

Theorem 3.1. *For any integer parameter $\Delta \geq 14$ and any constant $\lambda > 2$, the subgraph G of the unit disk graph U constructed above is a plane spanner of U containing a EMST of $V(U)$, whose degree is at most Δ, whose stretch factor is $(\lambda - 1) \cdot \rho$, where $\rho = (1 + 2\pi(\Delta \cos \frac{\pi}{\Delta})^{-1}) \cdot C_{del}$, and whose weight is at most $(1 + \frac{2}{\lambda - 2}) \cdot wt(EMST)$. Moreover, G can be constructed in $O(n \log n)$ time.*

Proof. The planarity and degree bound of G follow from the fact that G is a subgraph of G'_U. By construction, G contains a EMST of $V(U)$, and every fundamental face F_e of a non-tree edge e in G satisfies $wt(F_e) \geq \lambda \cdot wt(e)$. Therefore, by part (ii) of Theorem 2.1, we have $wt(G) \leq (1 + \frac{2}{\lambda - 2}) \cdot wt(\text{EMST})$. Since by Lemma 3.2 G can be constructed in $O(n \log n)$ time, it suffices to show that the stretch factor of G with respect to U is $(\lambda - 1) \cdot \rho$.

Note that G'_U has stretch factor ρ with respect to U. If an edge e_i is in G'_U but not in G, then by the construction of G, when the edge e_i is considered, the fundamental face F_i of e_i in $G+e_i$ satisfies $wt(F_i) \leq \lambda \cdot wt(e_i)$ (otherwise, the edge e_i would have been added). Therefore, when edge e_i was considered, G contained a path between the endpoints of e_i whose weight is at most $(\lambda - 1) \cdot wt(e_i)$. This path will remain in G after all edges in $\mathcal{L}'$ have been considered. Therefore, every edge in $E(G'_U) - E(G)$ is stretched by a factor at most $\lambda - 1$. Since G'_U has stretch factor ρ with respect to U, it follows that the stretch factor of G with respect to U is $(\lambda - 1) \cdot \rho$. This completes the proof. □

Note that since a Euclidean graph is a unit disk graph with radius equal to ∞, the above theorem holds for Euclidean graphs as well.

4 The Local Distributed Algorithm

In this section we present a local distributed algorithm that constructs a bounded-degree plane lightweight spanner of U.

The same paper by Kanj and Perlović [9], described above, presents a 3-local distributed algorithm that, given a unit disk graph U and an integer parameter $\Delta \geq 14$, constructs a plane spanner G' of U containing a EMST of $V(U)$, of degree at most Δ and stretch factor $\rho = (1 + 2\pi(\Delta \cos \frac{\pi}{\Delta})^{-1}) \cdot C_{del}$. Again, G' might not be of light weight, and we need to discard edges from G' so that the obtained subgraph is of light weight. Ultimately, we would like to be able to apply Theorem 2.1. However, a serious problem, which was not present previously in the centralized model, poses itself here in the local model: the removal of the edges from the spanner by different points in the graph needs to be coordinated. This problem was overcome in the centralized model by using a global ordering among the edges of the spanner. Clearly, no local distributed algorithm is capable of computing the global partial order described in Definition 2.1. To coordinate the removal of edges, we use an idea that at its core sits a clustering technique.

Fix an infinite rectilinear tiling $\mathcal{T}$ of the plane whose tiles are $\ell \times \ell$ squares, for some positive constant ℓ to be determined later. Assume, without loss of generality, that one of the tiles in $\mathcal{T}$ has its bottom-left corner coinciding with the origin $(0, 0)$, and that this fact is known to the points in U. Note that this assumption is justifiable in practice because an absolute reference system usually exists (a coordinates system, for example). Therefore, any point in U can determine (using simple arithmetic operations) which tile of $\mathcal{T}$ it resides in. We start with the following simple fact whose proof is easy to verify.

Fact 4.1. *Let C be a cycle of weight at most ℓ. The orthogonal projection* [1] *of C on any straight line has weight at most $\ell/2$.*

Let T_I be the translation with vector $(0, 0)$ (the identity translation), T_H the translation of vector $(\ell/2, 0)$ (horizontal translation), T_V the translation of vector $(0, \ell/2)$ (vertical translation), and T_D the translation of vector $(\ell/2, \ell/2)$ (diagonal translation). We have the following simple lemma.

Lemma 4.1. *Let C be any cycle of weight at most ℓ. There exists a translation T in $\{T_I, T_H, T_V, T_D\}$ such that the translate of C, $T(C)$, resides in a single tile of $\mathcal{T}$.*

Proof. (Sketch) If C resides within a single tile of $\mathcal{T}$ then clearly translation T_I serves the purpose. If C resides within exactly two horizontal (resp. vertical) tiles of $\mathcal{T}$, then these two tiles must be adjacent, and it is easy to verify using Fact 4.1 that translation T_H (resp. T_V) serves the purpose. Finally, if C resides within more than two tiles of $\mathcal{T}$, then again, using Fact 4.1, it can be easily verified that translation T_D serves the purpose. □

Even though a cycle of weight ℓ may not reside within a single tile of $\mathcal{T}$, Lemma 4.1 shows that by affecting some translation T in $\{T_I, T_H, T_V, T_D\}$, the translate of C under T will reside in a single tile. For each translation T in $\{T_I, T_H, T_V, T_D\}$, the points in G whose translates under T reside in a single tile will form a separate cluster. Then, these points will coordinate the detection and removal of the low-weight cycles residing in the cluster by applying a centralized algorithm to the cluster. Since the clusters do not overlap, and since each cluster works as a centralized unit, this maintains the stretch factor under control, while ensuring the removal of every low weight cycle. The centralized algorithm that we apply to each cluster is the standard greedy algorithm that has been extensively used (see for example [1]) to compute lightweight spanners. Given a graph H and a parameter $\alpha > 1$, this greedy algorithm sorts the edges in H in a non-decreasing order of their weight, and starts adding these edges to an empty graph in the sorted order. The algorithm adds an edge AB to the growing graph if and only if no path between A and B whose weight is at most $\alpha \cdot wt(AB)$ exists in the growing graph. We will call this algorithm **Centralized Greedy**. The following properties about this greedy algorithm are known:

Fact 4.2. *Let H be a subgraph of the Euclidean graph E, and let $\alpha > 1$ be a constant. Let H' be the subgraph of H constructed by the algorithm* **Centralized Greedy** *when applied to H with parameter α. Then:*

(i) H' is a spanner of H with stretch factor α.
(ii) H' contains a MST of H.
(iii) For any cycle C in H' and any edge e on C, $wt(C) > (1 + \alpha) \cdot wt(e)$.

[1] By the orthogonal projection of C on a given line we mean the set of points that are the orthogonal projections of the points in C on the given line. Note that, by the continuity of the curve C, this set of points is a straight line segment.

Lemma 4.2. *Let t_0 be a tile in $\mathcal{T}$, and let U_{t_0} be the subgraph of U induced by all the points of U residing in tile t_0. If A and B are two points in the same connected component of U_{t_0}, then A and B are $(\lfloor(8/\pi)\cdot(\ell+1)^2\rfloor)$-hop neighbors in U (i.e., A and B are at most $\lfloor(8/\pi)\cdot(\ell+1)^2\rfloor$ hops away from one another in U).*

Proof. Let $P_{min} = (A = p_0, p_1, \ldots, p_x = B)$ be a path between A and B in t_0 of minimum length. Let D_i, for $i = 0, \ldots, x$, be the disk centered at p_i and of radius $1/2$, and observe that all the disks D_i are contained within a bounding square-box B of dimensions $(\ell+1)\times(\ell+1)$, whose center coincides with the center of t_0. Observe also that the disks D_i, for even i, are mutually disjoint; that is, the points p_i, for even i, form an independent set in U (otherwise, P_{min} would not be a minimal-length path between A and B). Therefore, the area of the region R, denoted a, determined by the union of the disks D_i, for even i, is the sum of the areas determined by these individual disks. The value of a is precisely $(\pi/4)\cdot\lceil x/2\rceil$. Since the region R is contained in the bounding box B of area $(\ell+1)\times(\ell+1)$, we have $a \leq (\ell+1)^2$. Consequently, $(\pi/4)\cdot\lceil x/2\rceil \leq (\ell+1)^2$. Solving for the integer x in the previous equation we obtain $x \leq \lfloor(8/\pi)\cdot(\ell+1)^2\rfloor$. This shows that the length of the path P_{min}, which is x, is bounded by $\lfloor(8/\pi)\cdot(\ell+1)^2\rfloor$, and the proof is complete. □

We now present the local distributed algorithm formally and prove that it constructs the desired lightweight spanner. The input to the algorithm is the spanner G' of U constructed in [9], and a constant $\lambda > 2$. We set $\ell = \lambda$ in the above tiling $\mathcal{T}$. We assume that each point in U has computed its $(\lfloor(8/\pi)\cdot(\lambda+1)^2\rfloor)$-hop neighbors in U by applying the k-local k-neighborhood algorithm described in Section 2, where $k = \lfloor(8/\pi)\cdot(\lambda+1)^2\rfloor$. By Lemma 4.2, this ensures that every point knows all the points in its connected component residing with it in the same tile under any translation.[2] After that, for every round $j \in \{I, H, V, D\}$, each point $p \in U$ executes the following algorithm **Local-LightSpanner**:

(i) p applies translation T_j to compute its virtual coordinates under T_j; Suppose that the translate of p under T_j, $T_j(p)$, resides in tile $t_0 \in \mathcal{T}$;
(ii) p determines the set $S_j(p)$ of all the points in the resulting subgraph of G' (prior to round j) whose translates under T_j reside in the same connected component as $T_j(p)$ in tile t_0;
(iii) p applies the algorithm **Centralized Greedy** to the subgraph $H_j(p)$ of the resulting graph of G' induced by $S_j(p)$ with parameter $\alpha = \lambda - 1$; **if** p decides to remove an edge (p, q) from $H_j(p)$ **then** p removes (p, q) from its adjacency list in G';

Note that since all the points whose translate reside in a single tile apply the same algorithm to the same subgraph during any round j, if a point p decides to remove an edge (p, q), then point q must reach the same decision of removing edge (p, q).

[2] Note that the subgraph of G' induced by the set of points in a single tile may not be connected.

Let G be the subgraph of G' consisting of the set of remaining edges in G' after each point $p \in G'$ applies the algorithm **Local-LightSpanner**.

Theorem 4.1. *The subgraph G of G' is a spanner of U containing a EMST of $V(U)$, with stretch factor $\rho \cdot (\lambda - 1)^4$, and satisfying $wt(G') \leq (1 + \frac{2}{\lambda - 2}) \cdot wt(EMST)$, where ρ is the stretch factor of G'.*

Proof. We first show that G is of light weight. To do so, we need to show that G satisfies the conditions of part (i) in Theorem 2.1. We show first that G contains a EMST of $V(U)$.

Since G' contains a EMST of $V(U)$, it suffices to show that after each round of the algorithm **Local-LightSpanner**, the resulting graph still contains a EMST of $V(U)$. Fix a round $j \in \{I, H, V, D\}$, and let G'^{+} be the graph resulting from G' just before the execution of round j, and G'^{-} that resulting from G' after the execution of round j. Assume inductively that G'^{+} contains a EMST of $V(U)$. Note that any edge removed from G'^{+} in round j must have its translate contained within a single tile in $\mathcal{T}$. Let t_0 be a tile in $\mathcal{T}$. In round j, each point p whose translate $T_j(p)$ is in t_0, applies the algorithm **Centralized Greedy** to the subgraph of G'^{+}, $H_j(p)$, induced by the set of vertices $S_j(p)$ defined in the algorithm **Local-LightSpanner**. By part (ii) of Fact 4.2, this algorithm computes a spanner for $H_j(p)$ containing a "local" EMST τ_0 of $H_j(p)$. It is easy to see that an edge e in a EMST of G'^{+} whose translate $T_j(e)$ is in $H_j(p)$, its translate $T_j(e)$ is either an edge of τ_0, or is contained in a cycle whose edges other than e have the same weight as e and are in τ_0. Otherwise, by adding $T_j(e)$ to τ_0, we create a cycle on which $T_j(e)$ is the edge of maximum weight (if not, $T_j(e)$ could replace an edge of τ_0 of larger weight than e, contradicting the minimality of τ_0), and this means that $T_j(e)$ would be the edge of maximum weight on some cycle of G'; since a translation is an isometric transformation—and hence preserves length, this contradicts the fact that e is an edge in a EMST of G'^{+}. Therefore, if an edge in a EMST of G'^{+} is removed during round j, then G'^{-} will still contain a path between the endpoints of e all of whose edges have the same weight as e. Consequently, G'^{-} will still contain a EMST of $V(U)$. It follows that G contains a EMST of $V(U)$.

Now we show that for every cycle C in G, and for every edge e on C, we have $wt(C) \geq \lambda \cdot wt(C)$. Suppose not, and let cycle C and edge $e \in C$ be a counter example. Since every edge in U has weight at most 1, and $wt(C) < \lambda \cdot wt(e)$, it follows that $wt(C) < \lambda$, and by Lemma 4.1, there exists a round j in which the translate of C resides in a single tile t_0 of $\mathcal{T}$. By part (iii) of Fact 4.2, after the application of the algorithm **Centralized Greedy** to the connected component κ containing the translate of C in tile t_0 in round j, no cycle of weight smaller or equals to $(1+\alpha) \cdot wt(e) = (1+\lambda-1) \cdot wt(e) = \lambda \cdot wt(e)$ in the inverse translation of κ remains; in particular, the cycle C will no longer be present in the resulting graph. This is a contradiction. It follows that G satisfies the conditions of part (i) in Theorem 2.1, and $wt(G) \leq (1 + \frac{2}{\lambda - 2}) \cdot wt(\text{EMST})$.

Finally, it remains to show that the stretch factor of G, with respect to U, is at most $\rho \cdot (\lambda - 1)^4$. Since G' has stretch factor ρ, it suffices to show that after each round of the algorithm **Local-LightSpanner**, the stretch factor of

the resulting graph increases from the previous round by a multiplicative factor of at most $(\lambda - 1)$. Fix a round $j \in \{I, H, V, D\}$, and let G'^{+} and G'^{-} be as above. Suppose that an edge e is removed by the algorithm in round j. Then the translate of e in round j must reside in a single tile t_0 of $\mathcal{T}$. Since by part (i) of Fact 4.2 the algorithm **Centralized Greedy** has stretch factor $\alpha = \lambda - 1$, and since a translation is an isometric transformation, a path of weight at most $(\lambda-1) \cdot wt(e)$ remains between the endpoints of e in G'^{-}. Therefore, the stretch factor of G'^{-} with respect to G'^{+} increases by a multiplicative factor of at most $(\lambda - 1)$ during round j. This completes the proof. □

We conclude with the following theorem:

Theorem 4.2. *Let U be a connected unit disk graph, $\Delta \geq 14$ be an integer constant, and $\lambda > 2$ be a constant. Then there exists a k-local distributed algorithm with $k = \lfloor (8/\pi) \cdot (\lambda + 1)^2 \rfloor$, that computes a plane spanner of U containing a EMST of $V(U)$, of degree at most Δ, weight at most $(1 + \frac{2}{\lambda - 2}) \cdot wt(EMST)$, and stretch factor $(\lambda - 1)^4 \cdot (1 + 2\pi(\Delta \cos \frac{\pi}{\Delta})^{-1}) \cdot C_{del}$, where $C_{del} \approx 2.42$.*

5 Conclusion

We have developed in this paper a robust, scalable, and efficient algorithm for a fundamental communication problem—constructing efficient topologies for broadcasting *and* unicasting—in systems modeled as unit disk graphs. The bounds on the parameters of the algorithm and the constructed topology are small, and suggest that the algorithm and the topology are practical, as the following discussion shows.

In table 1, we compare the centralized Euclidean graph lightweight spanner algorithms LL92 by Levcopoulos and Lingas [13], ADDJS93 by Althöfer et al. [1], and BGS05 by Bose, Gudmundsson, and Smid [2] with our centralized algorithm KPX08 and our local distributed algorithm KPXLoc08, both developed to compute lightweight spanners of the more general unit disk graphs. The table gives the bounds on the stretch factor, the weight factor (the constant c^* such that the weight of the spanner is at most $c^* \cdot wt(\text{EMST})$), the maximum degree and the running time. Note that the first two algorithms (LL92 and ADDJS93) do not guarantee an upper bound on the degree of the spanner. Our algorithms

Table 1. A comparison of lightweight spanner algorithms given the constant $\lambda > 2$ and the maximum degree bound Δ; the following notations are used: $\rho^* = (\lambda - 1) \cdot C_{del}$, $c^* = (1 + \frac{2}{\lambda - 2})$, and $a^* = 1 + 2\pi(\Delta \cos \frac{\pi}{\Delta})^{-1}$

Algorithm	LL92 [13]	ADDJS93 [1]	BGS05 [2]	KPX08	KPXLoc08
Stretch factor	ρ^*	ρ^*	10.02	$a^* \cdot \rho^*$	$a^* \cdot (\lambda - 1)^3 \cdot \rho^*$
Weight factor	c^*	c^*	$O(1)$	c^*	c^*
Max. degree	∞	∞	27	Δ	Δ
Running time	$O(n \log n)$	$O(n^2 \log n)$	$O(n \log n)$	$O(n \log n)$	N/A

Table 2. Comparison between algorithm BGS05 [2] and our algorithms KPX08 and KPXLoc08 for different values of Δ

$\Delta =$	14	27
BGS05	N/A	$\rho^* = 10.02, c^* = O(1)$
KPX08	$\rho^*, c^* = 5.22$	$\rho^*, c^* = 4.63$
KPXLoc08	$\rho^*, c^* = 8.81$	$\rho^*, c^* = 8.08$

match their bounds on the weight factor to provide a maximum degree bound at a small multiplicative cost in the stretch factor (a^* for our centralized algorithm and $(\lambda - 1)^3 \cdot a^*$ for our local distributed algorithm). For example, for a degree bound of 14, our upper bound on the stretch factor increases (with respect to [13] and [1]) by a multiplicative constant of 1.47 for the centralized algorithm, and of 2.92 (corresponding to $\lambda = 2.256$) for the local distributed algorithm. For larger values of Δ, the multiplicative factors are even smaller.

In table 2 we use some concrete values for Δ and λ in order to compare our algorithms with the algorithm BGS05 by Bose, Gudmundsson, and Smid [2]. Their algorithm only guarantees a maximum degree bound of 27. The listed bounds for stretch factor ρ^* and weight factor c^* for $\Delta = 27$ are obtained by setting $\lambda = 2.551$ in KPX08 and $\lambda = 2.282$ in KPXLoc08. The bounds for stretch factor ρ^* and weight factor c^* when $\Delta = 14$ are obtained by setting $\lambda = 2.475$ in KPX08 and $\lambda = 2.256$ in KPXLoc08.

References

1. Althöfer, I., Das, G., Dobkin, D., Joseph, D., Soares, J.: On sparse spanners of weighted graphs. Discrete & Computational Geometry 9, 81–100 (1993)
2. Bose, P., Gudmundsson, J., Smid, M.: Constructing plane spanners of bounded degree and low weight. Algorithmica 42(3-4), 249–264 (2005)
3. Bose, P., Morin, P., Stojmenovic, I., Urrutia, J.: Routing with guaranteed delivery in ad hoc wireless networks. wireless networks 7(6), 609–616 (2001)
4. Damian, M., Pandit, S., Pemmaraju, S.: Local approximation schemes for topology control. In: Proceedings of PODC, pp. 208–217 (2006)
5. Das, G., Heffernan, P., Narasimhan, G.: Optimally sparse spanners in 3-dimensional euclidean space. In: Proceedings of SoCG, pp. 53–62 (1993)
6. Das, G., Narasimhan, G.: A fast algorithm for constructing sparse euclidean spanners. In: Proceedings of SoCG, pp. 132–139 (1994)
7. Das, G., Narasimhan, G., Salowe, J.: A new way to weigh malnourished euclidean graphs. In: Proceedings of SODA, pp. 215–222 (1995)
8. Gudmundsson, J., Levcopoulos, C., Narasimhan, G.: Fast greedy algorithms for constructing sparse geometric spanners. SIAM J. Comput. 31(5), 1479–1500 (2002)
9. Kanj, I., Perković, L.: On geometric spanners of euclidean and unit disk graphs. In: Proceedings of STACS (2008)
10. Kanj, I., Perkovic, L., Xia, G.: Computing lightweight spanning subgraphs locally. Technical report # 08-002, `http://www.cdm.depaul.edu/research/Pages/TechnicalReports.aspx`

11. Kranakis, E., Singh, H., Urrutia, J.: Compass routing on geometric networks. In: Proceeding of CCCG, vol. 11, pp. 51–54 (2005)
12. Kuhn, F., Moscibroda, T., Wattenhofer, R.: What cannot be computed locally! In: Proceedings of PODC, pp. 300–309 (2004)
13. Levcopoulos, C., Lingas, A.: There are planar graphs almost as good as the complete graphs and almost as cheap as minimum spanning trees. Algorithmica 8(3), 251–256 (1992)
14. Li, X.-Y., Calinescu, G., Wan, P.-J., Wang, Y.: Localized delaunay triangulation with application in ad hoc wireless networks. IEEE Trans. on Parallel and Distributed Systems. 14(10), 1035–1047 (2003)
15. Linial, N.: Locality in distributed graph algorithms. SIAM J. Comput. 21(1), 193–201 (1992)
16. Peleg, D.: Distributed computing: A Locality-Sensitive Approach. SIAM Monographs on Discrete Mathematis and Applications (2000)
17. Wang, Y., Li, X.-Y.: Localized construction of bounded degree and planar spanner for wireless ad hoc networks. MONET 11(2), 161–175 (2006)
18. Wattenhofer, R.: Sensor networks: distributed algorithms reloaded - or revolutions? In: Proceedings of SIROCCO, pp. 24–28 (2006)