Separability and Topology Control of Quasi Unit Disk Graphs

Jianer Chen*, Anxiao(Andrew) Jiang*, Iyad A. Kanj†, Ge Xia‡, and Fenghui Zhang*
* Dept. of Computer Science, Texas A&M Univ. College Station, TX 77843. {chen, ajiang, fhzhang}@cs.tamu.edu.
† School of CTI, DePaul University, 243 S. Wabash Avenue, Chicago, IL 60604. ikanj@cs.depaul.edu.
‡Department of Computer Science, Lafayette College, Easton, PA 18042. gexia@cs.lafayette.edu.

***Abstract*— A deep understanding of the structural properties of wireless networks is critical for evaluating the performance of network protocols and improving their designs. Many protocols for wireless networks — routing, topology control, information storage/retrieval and numerous other applications — have been based on the idealized unit-disk graph (UDG) network model. The significant deviation of the UDG model from many real wireless networks is substantially limiting the applicability of such protocols. A more general network model, the quasi unit-disk graph (quasi-UDG) model, captures much better the characteristics of wireless networks. However, the understanding on the properties of general quasi-UDGs has been very limited, which is impeding the designs of key network protocols and algorithms.**

In this paper, we present results on two important properties of quasi-UDGs: separability and the existence of power efficient spanners. Network separability is a fundamental property leading to efficient network algorithms and fast parallel computation. We prove that every quasi-UDG has a corresponding grid graph with small balanced separators that captures its connectivity properties. We also study the construction of wireless backbones through topology control for efficient communication and present a distributed localized algorithm that builds a *nearly planar* backbone in any quasi-UDG with low constant stretch factor and degree. We demonstrate the excellent performance of these properties through simulations and show, among many applications, their application in efficient routing.

I. Introduction

The connectivity structures of wireless networks exhibit strong correlations with the physical environment due to the signal transmission model of wireless nodes. A deep understanding of the structural properties of wireless networks is critical for evaluating the performance of network protocols and improving their designs. So far, many protocols have been based on the idealized unit-disk graph (UDG) network model, where two wireless nodes can directly communicate if and only if their physical distance is within a fixed parameter R. Examples of these protocols include routing [3], [9], topology control [1], distributed information storage/retrieval [4] and a great variety of other applications. In practice, however, the UDG model significantly deviates from many real wireless networks, due to reasons including multi-path fading [6], [13], antenna design issues, inaccurate node position estimation, etc. It is not uncommon to observe stable links that are five times longer than unstable short links [13]. The significant deviation of the UDG model from practice is substantially limiting the applicability of protocols based on UDGs. To combat the problem, a much more general network model, the quasi unit-disk graph (quasi-UDG) model, has been proposed in recent years to capture the non-uniformity characteristic of most wireless networks. Formally, it is defined as follows.

Definition 1: A *quasi-UDG* model is characterized by two positive parameters R and r ($R \geq r$). For any two nodes u, v in a quasi-UDG network deployed in a plane, let $d(u, v)$ denote their Euclidean distance. Then, if $d(u, v) \leq r$, an edge (link) exists between u and v; if $d(u, v) > R$, the edge does not exist; if $r < d(u, v) \leq R$, the edge may or may not exist.

The understanding on the properties of general quasi-UDGs, however, has been very limited. That is in sharp contrast to UDG, whose properties have been well understood [1], [9]. Among the limited knowledge about quasi-UDG, a notable result is the "link-crossing" property discovered for quasi-UDGs where $R \leq \sqrt{2} \cdot r$ [2]. The serious lack of understanding on the properties of general quasi-UDGs is impeding the designs of key network protocols and algorithms.

In this paper, we present results on two important properties of quasi-UDGs: separability and the existence of power efficient spanners. Network separability is a fundamental property leading to efficient network algorithms and fast parallel computation [11]. A (vertex) *separator* of a graph G is a set of vertices whose removal splits the graph into two non-adjacent parts of similar sizes. We call a graph G *well separable* if any subgraph of G has relatively small separators. A well separable graph has strong locality properties. As a result, the performance of protocols for routing, information retrieval, network monitoring, etc., can be significantly improved for such graphs. We first construct a grid graph that is an abstraction of the given quasi-UDG G and show that the grid graph is well separable. The separator we obtain is of size $O(\sqrt{N})$ and can split the graph into two parts of size roughly $\frac{N}{2}$, where N is the number of nodes of the grid graph. In addition, both the degrees of the grid nodes and the number of edges crossing any edge are upper bounded by constants. Among many applications of the separators, we present, as an example, a compact routing protocol based on the grid graph construction and distance labelling. We prove that the routing table size of each node in our protocol is bounded by $O(\sqrt{N} \log N)$, which is much better than the tight bound proved for general graphs and close to the lower bound of $\Omega(\sqrt{N})$ for degree bounded graphs in [7]. The ratio of the routing path length to the shortest path length is upper bounded by $2 + \epsilon$ where ϵ is a small constant. More extensions of the

results are also included.

In the second part of the paper we study the existence and the construction of energy efficient backbones for quasi-UDGs. A backbone is a spanning subgraph of the wireless network for efficient communication, obtained through pruning a set of edges. By using only those edges in the backbone for communication, signal interference, routing table size and power usage can be substantially reduced. A major requirement for backbone construction is to preserve the shortest path distances between vertices as much as possible. For a backbone B of a graph $G = (V, E)$, the *stretch factor* is defined as $s(B) = \max\{\frac{f_B(u,v)}{f_G(u,v)} | u, v \in V\}$, where $f_B(u, v)$ and $f_G(u, v)$ are the distances between vertices u, v in B and G, respectively. The stretch factor reflects the quality of the backbone. There have been results showing that for UDGs, bounded degree and planar spanners can be constructed when the distance function $f(u, v)$ is defined as the minimum power needed to send a message from u to v [8][14]. In this paper, we present a distributed algorithm that constructs a backbone B for any quasi-UDG G with a constant power stretch factor. The node degrees of the backbone B are upper bounded by a constant. In addition, although it is in general impossible to construct planar backbones with constant stretch factors for quasi-UDG, we show that B is *nearly planar*, specifically, B has a constant upper bound on the average number of edges crossing an edge. The latter property is useful for geographic routing algorithms based on cross link detection [10].

We evaluate the performance of the separators, the routing protocol and the backbone construction through extensive simulations. Their performance is much better compared to the theoretical analysis of the worst cases. This shows that although the quasi-UDG model is quite different from the UDG model, efficient algorithms can still be developed by exploiting the locality in the model.

The rest of the paper is organized as follows. In section II, we present the grid graph construction and prove its separability result. In section III, we present the backbone construction through topology control. In Section IV, we present the compact routing protocol based on the grid graph and distance labelling, as well as the simulation results. We conclude the paper in section V.

II. Grid graph of quasi-UDGs

In this section, we present a distributed algorithm for constructing a grid graph for any quasi-UDG, and prove that the grid graph is well separable. The grid graph, whose node density and edge density are both upper bounded by constants, is an abstraction of the quasi-UDG. A quasi-UDG may have highly variable node and edge densities, which prevent it from having small separators. The grid graph is a "sparsified" version of the quasi-UDG, which retains the distance information for vertices and well represents the deployment region of the quasi-UDG. As a result, the connectivity-related results for the grid graph can be easily mapped to results for the quasi-UDG. An example of a quasi-UDG and its corresponding grid graph is shown in Fig. 1(a), (b). In the following, we present details on the grid graph.

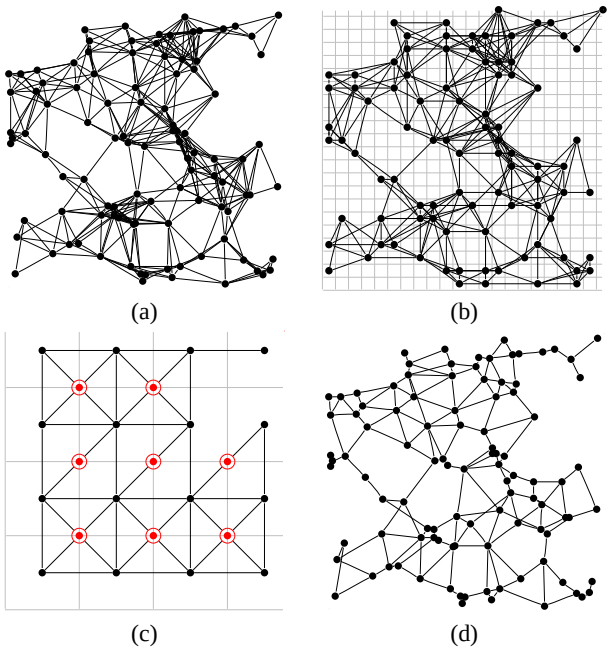


Fig. 1. Grid Graph Example. (a) A quasi-UDG G with 100 vertices and $R/r = 0.5$; (b) The grid graph corresponding to G; (c) The auxiliary graph used to find the top level separator of G; (d) The backbone of G.

A. Construction of the grid graph H

To obtain a grid graph H for a quasi-UDG G, we impose a grid on the plane and view each non-empty cell as a vertex. The construction is shown in Fig. 2.

Algorithm **GridGraph**

INPUT: $G = (V_G, E_G)$: *a quasi-UDG with parameters* R *and* r
OUTPUT: $H = (V_H, E_H)$: *the grid graph for* G

1. Impose a grid of cell size $\frac{r}{\sqrt{2}} \times \frac{r}{\sqrt{2}}$ on the plane;
2. For each cell that has at least one vertex of G, H has a corresponding vertex, whose position is set at the center of the cell;
3. There is an edge between two vertices of H if and only if there is at least one edge connecting two vertices of G that are, respectively, in the two corresponding cells.

Fig. 2. Constructing grid graph for quasi-UDG

All the vertices of G in the same grid cell are adjacent. The algorithm **GridGraph** can be easily implemented in a distributed manner. The following theorem proves the constant upper bounds for the node density, edge density and the number of edges crossing any edge in the grid graph H.

Theorem 1: The algorithm **GridGraph** constructs a grid graph H for given quasi-UDG G such that: (1) inside any disk of radius y, there are at most $O(\frac{y^2}{r^2})$ vertices; (2) the degree of each vertex is upper bounded by $O(\frac{R^2}{r^2})$;(3) the number of edges crossing any edge is upper bounded by $O(\frac{R^4}{r^4})$.

Proof: By the algorithm, the Euclidean distance between any two vertices of H is at least $\frac{r}{\sqrt{2}}$. Hence if we place an

open disk of radius $\frac{r}{2\sqrt{2}}$ centered at every vertex, no two disks will intersect. Therefore given any disk of radius y, the number of such open disks intersecting it is upper bounded by $O(\frac{y^2}{r^2})$. So is the number of vertices of H inside the disk.

Consider a vertex U of H, denote by $v(U)$ the set of nodes of G inside the cell represented by U. The number of vertices of H within distance $R+r$ to U is bounded by $O(\frac{R^2}{r^2})$. No node of G in $v(U)$ can be adjacent to $w \in v(V)$ where V is more than distance $R+r$ from U. Hence the degree of U is upper bounded by $O(\frac{R^2}{r^2})$.

Similarly, for an edge $\{U,V\}$ of H, the number of grid vertices within distance $R+r$ to any point in the line segment connecting U and V is also upper bounded by $O(\frac{R^2}{r^2})$. Therefore, the total number of edges crossing $\{U,V\}$ is upper bounded by $O(\frac{R^4}{r^4})$. ■

If two vertices of H are h hops away from each other, then two vertices of G in the two corresponding cells are at most $2h+1$ hops away from each other. Note that the above method for constructing grid graphs, and the above results, can be easily extended to three and higher dimensional spaces.

B. Separability of the grid graph H

Network separability is a fundamental property that leads to efficient network algorithms (in particular, those algorithms based on the divide and conquer paradigms),— fast parallel computation, and improvements in the study of computational complexity [11]. Many applications in wireless ad hoc networks (routing, information retrieval, etc.), as well as quite a number of hard theoretical problems, have more efficient solutions if the underlying graph is well separable. For example, shortest path routing can be realized with small routing tables when the graph has small separators, as in the case of planar graphs or graphs with bounded tree width [7]. Also, NP hard problems such as vertex cover and independent set are solvable in polynomial time if the input graph and all its subgraphs have bounded separators.

In this subsection, we study the separability of the grid graph obtained above. We begin with a formal definition of the separability of graphs.

Definition 2: Given a graph G of n vertices, a b-**separator** of G is a set of vertices whose removal splits G into two non-adjacent subgraphs, each of which contains at most bn vertices. We call a graph G $(f(n'),b)$-**separable** if every subgraph of G has a b-separator of at most $O(f(n'))$ vertices, where $f(n')$ is a function of the number of vertices n' in that subgraph.

In order to compute a small separator for the grid graph H, we use the help of a planar auxiliary graph T. First, we impose a larger grid on the plane and map the grid graph H to an auxiliary graph that is nearly planar. Then, we planarize it by adding a virtual vertex at the middle of each diagonal edge, eliminating all edge crossings. (Note that we see all the edges as being straight.) The detailed construction of the auxiliary graph T is presented in Fig. 3. All the virtual vertices in T are denoted by *red vertices* and the others — which represent cells — are denoted by *black vertices*. Each red vertex has weight zero, while each black vertex has a weight that equals the number of vertices of H in the corresponding cell.

Algorithm **AuxiliaryGraph**

INPUT: $H=(V_H,E_H)$*: a grid graph with parameters* R *and* r
OUTPUT: $T=(V_T,E_T)$*: the auxiliary graph for* H

1. Impose a grid of cell-size $(R+\frac{r}{\sqrt{2}})\times(R+\frac{r}{\sqrt{2}})$ on the plane;
2. For each cell that has at least one vertex of H, T has a corresponding *black* vertex v, whose position is set at the center of the cell; we assign to v a weight that equals the number of vertices of H in that cell;
3. Add an edge between two black vertices u,v of T if and only if there is at least one edge connecting two vertices of H that are, respectively, in the two corresponding cells;
4. For each pair of crossing edges $\{u,v\}$, $\{w,x\}$, add a *red* vertex at the intersection of the two edges and replace those two original edges with four new edges that connect the red vertex, respectively, to the four black vertices u,v,w and x; let the weight of the red vertex to be 0;
5. For each diagonal edge between two black vertices, we add a *red* vertex of weight 0 at the middle of the edge and replace that original diagonal edge with two new edges that connect the red vertex, respectively, to those two black vertices.

Fig. 3. AuxiliaryGraph(H)

Fig. 1(c) shows an example of the auxiliary graph. The longest edge in the auxiliary graph has length $R+\frac{\sqrt{2}r}{2}$, and red vertices are either of degree 2 or 4. Since the cell we apply in this algorithm is large enough (of side length $R+\frac{r}{\sqrt{2}}$) and all black vertices are placed at the centers of their corresponding cells, any black vertex may only connect to the eight black vertices around it before the red vertices were added. Therefore, around each black vertex, there can be at most four red vertices; and no two red vertices are adjacent to each other. Formally, we have the following lemma.

Lemma 1: Let $N_{T,b}$ be the number of black vertices in the auxiliary graph T. Then T is a planar graph of at most $2N_{T,b}$ vertices, and no two red vertices are adjacent.

Lipton and Tarjan proved in their celebrated Separation Theorem [11] that for any vertex-weighted planar graph of n vertices, there exists a set of $O(\sqrt{n})$ vertices that separates the graph into two non-adjacent subgraphs, each of which weighs at most $\frac{2}{3}$ of the total weight of the graph. The separator algorithm presented in [11], however, is relatively complex. For the planar auxiliary graph T, which has a constrained structure, we present a simpler and practically more efficient algorithm for finding such a small separator. Based on that, the algorithm also finds a small separator for the grid graph H. The details of the algorithm are presented in Fig. 4.

We now prove that the algorithm **Separator** constructs small balanced separators for H and T. We start with a lemma.

Lemma 2: Let $\hat{T}$ be any subgraph of the auxiliary graph T. If its outer face has k vertices, then the number of inner vertices (the vertices not on the outer face) is at most $\lfloor\frac{k^2}{2\pi}\rfloor$.

Proof: The outer face of the planar graph T' is a closed curve (or closed curves, if $\hat{T}$ is disconnected) on the plane. Let $x=R+r/\sqrt{2}$ be the side length of the cells in the construction of the auxiliary graph T. For each inner vertex of $\hat{T}$, we place a $\frac{\sqrt{2}x}{2}\times\frac{\sqrt{2}x}{2}$ square centered at it, then rotate

Algorithm **Separator**

INPUT: *H: a grid graph with parameters R and r*
OUTPUT: *S_H: a separator for H.*
S_T: a separator for T. (T is the auxiliary graph of H.)

1. Let T be the auxiliary graph of H. Let T' be a copy of T.
2. Build a breadth-first search (BFS) tree for a dynamically changing graph T' (T' changes because new edges are added to it during the BFS procedure) in the following way: (1) pick a vertex v on the outer face of T' to be the root and start the BFS; (2) during the BFS process, when a vertex u is dicovered (put into the BFS tree), for every face containing u, add edges from u to as many other vertices in the face as possible so long as T' remains a simple planar graph; if after adding those edges, there are still faces containing u that are not triangulated, add edges to triangulate them arbitrarily. During the BFS, a vertex's undiscovered neighbors are visited in the clockwise order (starting with the vertex's parent in the BFS tree as the reference point);
3. Check every *fundamental cycle* (a cycle formed by a non-tree edge and some tree edges) in the BFS tree. Let S_T be a fundamental cycle that separates T' (therefore also T) in the most balanced way, i.e. the difference between the summation of the weights of vertices in the two separated subgraphs A_1, B_1 is minimized.
4. Consider the graph T. Let S'_T be a copy of S_T. For each red vertex u in S'_T with the set of neighboring vertices $N(u)$, we distinguish two cases: **Case (1)** All vertices in $N(u)$ belong to A_1(respectively, B_1) except those in S'_T. Then, we move u from S'_T to A_1(respectively, B_1); **Case (2)** Both A_1 and B_1 contain vertices of $N(u)$. Then, we put all vertices in $N(u)$ into S'_T and move u from S'_T to A_1.
5. Let S_H be the set of vertices of H in those cells corresponding to the black vertices of T in S'_T. Let A_2, B_2 be the two sets of vertices of H in those cells corresponding to the black vertices of T in A_1 and B_1. Clearly, S_H separates H into A_2 and B_2.

Fig. 4. Separator

the square by 45 degrees. It is simple to see that now these (diamond shaped) squares centered at the inner vertices do not overlap each other. The area of each square is $\frac{x^2}{2}$.

First consider the case when the outer face is connected, i.e. $\hat{T}$ is connected. The outer face of $\hat{T}$ consists of several (at least one) simple cycles. Suppose there are i such simple cycles of size $k_1, k_2, \ldots, k_i$ in the outer face. $\sum_{j=1}^{i} k_j$ can be greater than k, the number of vertices in the outer face, because in the summation a vertex can be counted more than once. The simple cycles form the outer face of a planar graph, so the number of times vertices are over-counted is exactly $i-1$. Thus $\sum_{j=1}^{i} k_j = k + i - 1$.

First we have $k^2 = \left[\left(\sum_{j=1}^{i} k_j\right) - i + 1\right]^2 = \sum_{j=1}^{i} k_j^2 + \sum_{j=1}^{i}\left(k_j \sum_{l\neq j}^{i} k_l\right) - \sum_{j=1}^{i} 2(i-1)k_j + (i-1)^2 \geq \sum_{j=1}^{i} k_j^2 + \sum_{j=1}^{i}\left[k_j\left(\sum_{l\neq j}^{i} k_l - 2(i-1)\right)\right] \geq \sum_{j=1}^{i} k_j^2$. The last inequality holds because $k_j \geq 2$ and $\sum_{l\neq j} k_l$ contains exactly $i-1$ terms. The equality holds when $i = 1$.

Each simple cycle of k_j vertices has k_j edges, thus the perimeter of the cycle is at most $k_j x$. Therefore the area of the region inside the cycle k_j is at most $\lfloor \frac{k_j^2 x^2}{4\pi} \rfloor$ and the total area of the regions inside the outer face is bounded by $\sum_{j=1}^{i} \lfloor \frac{k_j^2 x^2}{4\pi} \rfloor \leq \lfloor \frac{k^2 x^2}{4\pi} \rfloor$.

Now if there are several disconnected cycles in the outer face, each connected part — say, of k' vertices — surrounds a region of area no more than $\lfloor \frac{k'^2 x^2}{4\pi} \rfloor$, since $\sum k'^2 \leq (\sum k')^2 = k^2$, the total area of the regions surrounded by the outer face is also bounded by $\lfloor \frac{k^2 x^2}{4\pi} \rfloor$ Thus, in all cases, the total number of inner vertices is bounded by $\lfloor \frac{k^2 x^2}{4\pi} \rfloor / \frac{x^2}{2} = \lfloor \frac{k^2}{2\pi} \rfloor$. ■

Define the *depth* of a tree to be the maximum number of edges in a path from the root to a leaf. We have:

Lemma 3: Let N_T be the number of vertices in the auxiliary graph T. The BFS tree constructed in Step 2 of the algorithm **Separator** is of depth at most $\sqrt{N_T}$.

Proof: Let d be the depth of the BFS tree. Because of the triangulation operation enforced on the graph T' during the BFS process, for $i = 1, 2, \cdots, d-1$, the vertices at level i (if $i = 1$, include the root as well) of the BFS tree actually contain all the vertices on the outer face of the subgraph induced by the vertices at levels $i, i+1, \cdots, d$. So it suffices to show that if we "peel off" one outer face from T' at each step, T' becomes an empty graph after $t \leq \sqrt{N_T}$ steps.

Let n_x be the number of vertices remaining in the graph T' after x steps. (By convention, define $n_0 = N_T$.) By Lemma 2, we know that in the x-th step we have "peeled off" at least $\lceil \sqrt{2\pi N_x} \rceil$ vertices. So $n_{t-1} \geq 1$, $n_i \geq n_{i+1} + \lceil \sqrt{2\pi n_{i+1}} \rceil$ for $i = t-2, t-3, \cdots, 0$. Now let us prove that $n_{t-j} \geq j^2$ by induction: when $j = 1$, we have $n_{t-1} \geq 1$ and when $j = 2$, we have $n_{t-2} \geq 4$; suppose our claim is true for $2 \leq j \leq i$; consider the case $j = i+1$, where $n_{t-(i+1)} \geq n_{t-i} + \lceil \sqrt{2\pi n_{t-i}} \rceil \geq i^2 + \lceil \sqrt{2\pi} \rceil i \geq i^2 + 2i + 1 = (i+1)^2$.

We have $N_T = n_0 = n_{t-t} \geq t^2$. So $t \leq \sqrt{N_T}$. ■

By Lemma 2 in [11], if a vertex-weighted planar graph has a spanning tree of depth h, then there exists a fundamental cycle of size at most $2h+1$ that separates the graph into two non-adjacent subgraphs each of which weighs no more than $2/3$ of the total weight of the graph. As the BFS tree obtained in Step 2 of Algorithm **Separator** is of depth at most $\sqrt{N_T}$, we have the following theorem immediately.

Theorem 2: Let N_T be the number of vertices in the auxiliary graph T, and let N_H be the number of vertices in H. Then, the total weight of the vertices of T is N_H, and the set S_T obtained in Algorithm **Separator** contains at most $2\sqrt{N_T} + 1$ vertices and separates T into two non-adjacent subgraphs each of which weighs no more than $\frac{2N_H}{3}$.

We now prove that the algorithm **Separator** also finds a small balanced separator for the grid graph H.

Theorem 3: Let N_H be the number of vertices in the grid graph H. Then, the algorithm **Separator** constructs a separator S_H of size $O(\sqrt{N_H})$ that separates H into two non-adjacent subgraphs each of which has no more than $\frac{2N_H}{3}$ vertices. Moreover, the grid graph H is $(\sqrt{n'}, \frac{2}{3})$-separable when the weights of all the vertices of H are set to be 1.

Proof: Let N' be the number of black nodes in T. Clearly $N' \leq N_H$; and it is straightforward that each cell corresponding to a black vertex of T contains at most $\lceil \frac{2(R+\sqrt{2}r/2)^2}{r^2} \rceil$ vertices of H. Hence we have $N' = \Theta(N_H)$. From lemma 1 we know that the number of red vertices is no more than N', and the total weight of vertices in T is N_H. Hence the separator S_T for T contains no more than $2\sqrt{2N'} + 1$ vertices

whose weights sum up to $O(\sqrt{N_H})$, and separates T into two parts each of which weighs no more than $\frac{2N_H}{3}$.

Now we show that after Step 4 of Algorithm **Separator**, S'_T is still a separator for T of size $O(\sqrt{N'})$, and A_1 and B_1 are still of weight no more than $\frac{2N_H}{3}$. Consider any red vertex $u \in S'_T$ in Step 4, in the case where all of u's neighbors are either in S_T or A_1 (respectively, B_1), $S'_T \setminus \{u\}$ can separate T into $A_1 \cup \{u\}$ and B_1 (respectively, A_1 and $B_1 \cup \{u\}$). Note that u has weight 0, so moving u from S'_T to A_1 (or, B_1) does not change their weights. In the complimentary case, the algorithm moves all u's neighbors into S'_T and moves u into A_1; clearly S'_T still separates A_1 and B_1. And by doing that, we decrease the weights of both A_1 and B_1. The size of S'_T increases by at most 3 for each red vertex.

Hence after Step 4, we have replaced all red vertices in S'_T by black ones, increasing the size of S'_T by at most three times, not increasing the weights of A_1 and B_1. Most importantly, S'_T still separates A_1 and B_1. Therefore S'_T is still of size $O(\sqrt{N'}) = O(\sqrt{N_H})$, and the weights of A_1 and B_1 are no more than $\frac{2N_H}{3}$. Each cell corresponding to a black vertex of T contains a bounded number of vertices of H, so S_H is of size $O(\sqrt{N_H})$. Also, the number of vertices in A_2 (resp., B_2) equals the weight of A_1 (resp., B_1) (at most $\frac{2N_H}{3}$).

By the construction of the auxiliary graph T, if no two black vertices are joined by an edge or two edges with a red vertex in the middle, there is no edge connecting vertices of H in those two corresponding cells. A_1 and B_1 are not adjacent in T, and S'_T has no red vertex. So A_2 and B_2 obtained in Step 5 are not adjacent in H, and S_H separates A_2 and B_2 in H.

It is simple to see that any subgraph of H can be used as the input of Algorithm **Separator**, and the above arguments still hold. Hence H is $(\sqrt{n'}, \frac{2}{3})$-separable. ■

For some applications, a perfectly balanced separator is desirable. By using the same technique described in [11], we can construct a separator of size $O(\sqrt{N_H})$ that separates H into two parts each of which has no more than $\frac{N_H}{2}$ vertices. The idea is to separate the larger part of the outcome of the algorithm recursively. Hence we have

Corollary 1: Let N_H be the number of vertices in the grid graph H. H is $(\sqrt{n'}, 0.5)$-separable.

For the grid graph, we can develop a shortest path routing scheme based on its separators, using the idea of distance labelling [7]. We can then transform it into a compact routing scheme for the underlying quasi-UDG G with a small stretch factor. The following theorem summarizes the result. We leave the details of the routing algorithm, the proof of Theorem 4 and the extended results to section IV.

Theorem 4: For any quasi-UDG G of N_G vertices, let $h(u, v)$ be the minimum hop distance between vertices u, v. There is a routing protocol that guarantees the routing path from u to v to have at most $2h(u, v) + 1$ hops, for any two vertices u and v. The size of the routing table at each node and the message overhead are both $O(\sqrt{N_G} \log N_G)$.

III. Backbone with constant stretch factor

We denote by *backbone* of a given graph a subgraph that contains the same set of vertices but fewer edges. One example of backbones are spanning trees. Backbones, particularly those with small stretch factors and degrees, have very important applications in wireless communication because they can help reduce signal interferences and simplify algorithms.

In this section, we present a distributed construction of a backbone with constant stretch factor, constant node degree and a small number of edge crosses for quasi-UDGs. It is also an extension of the grid method described in Section II. We will show in Section IV that these backbones can also help reduce the routing table size in our routing scheme.

A. Algorithm constructing the backbone

Energy is a major limitation in wireless networks. Accordingly, the stretch factor of backbones is often defined based on energy consumption. We start with its formal definition.

Definition 3: Let $u = u_1 \to u_2 \to \cdots u_k = v$ be a path from u to v in the graph G. Denote by $|ab|_G$ the Euclidian distance between any two vertices a and b. The **communication cost** between u, v following the given path is defined as:

$$c_G(u, v) = \sum_{i=1}^{k-1} \alpha |u_i u_{i+1}|_G^\beta,$$

where β is the path loss exponent, $2 \le \beta \le 5$, and α is a scaling factor linear in the number of sent bits. If there is no path from u to v, $c_G(u, v)$ is defined as $+\infty$.

Definition 4: Given a graph $G = (V, E)$ and a backbone B of G, the **stretch factor** of B is defined as:

$$\max_{u,v \in V} \left\{ \frac{c_{B,min}(u, v)}{c_{G,min}(u, v)} \right\},$$

where $c_{B,min}(u, v)$ and $c_{G,min}(u, v)$ denote the *minimum* communication cost (over all the paths) between u, v in graph B and G, respectively.

The stretch factor defined above is also called the *power stretch factor*. We say that a backbone is *energy efficient* if its power stretch factor is bounded by a constant.

We next present a distributed localized algorithm that, when given a quasi-UDG G, constructs a backbone where the maximum degree of a node is bounded by $O(\frac{R^2}{r^2})$, the average number of crossings of an edge is bounded by $O(\frac{R^4}{r^4})$ and the power stretch factor is bounded by $3+\epsilon$, where ϵ is a constant that can be made arbitrarily small. To run the algorithm, we classify the edges in the quasi-UDG G into two types: **short edges** whose lengths are no greater than r; and **long edges** whose lengths are strictly larger than r.

In our algorithm, we first reduce the number of short edges in the graph by applying an operation similar to Gabriel Planarization [5] to make the subgraph induced by all short edges of G a planar graph. In the second step, we apply an operation described in [8] to bound the number of short edges incident to any node. Finally, we apply a grid operation to reduce the number of long edges in the graph. Figure 5 contains the details for our algorithm.

Algorithm **QuasiUDG-Backbone**

INPUT: *G: a quasi-UDG with parameters R and r*
OUTPUT: *B: a backbone of G*

1. **Planarize the subgraph induced by short edges of G**
   The subgraph B will contain the same vertex set as G. Initially, the edge set of B is set to empty. For each edge $e\{u, v\}$ in G, if there is no common neighbor of u and v in G residing in the disk whose diameter is the edge $e\{u, v\}$, we add $e\{u, v\}$ into B. Similar to the Algorithm 1 described in [14], this process can be done in a distributed manner by exchanging no more than $O(m)$ messages where m is the number of edges in G.
2. **Reduce the number of short edges incident to each vertex**
   Let G' be the subgraph of B that includes all the vertices and short edges of B. Note that here G' is in fact the Gabriel graph constructed from a UDG (with communication range r); so G' is planar. We apply the algorithm described in [8] on G'. Here is a brief description of the algorithm that is performed by each vertex: Direct the edges in G' (using the classical acyclic orientation of a planar graph) so that every vertex in G' has at most 5 incoming edges; Perform a standard Yao step [8] on the set of outgoing edges; Select certain edges that form large angles with consecutive edges (see [8] for details); Finally, communicate with all the neighbors of the vertex and keep edges that have been selected by least one of their ends.
   When the above algorithm ends, we remove from B those edges that have been removed by the algorithm from G'. This step will reduce the number of short edges incident to every vertex to a constant $k + 5$, where k is a selectable parameter, and it can be done locally. Compared to the subgraph of G that contains all the short edges of G, B increases the minimum communication cost between any two vertices by a factor of at most $1+(2\sin(\pi/k))^\beta$, where k is a parameter, and β is the path loss exponent.
3. **Reduce the number of long edges incident to each vertex**
   Add all the long edges of G to B. We impose a grid of cell-size $\frac{r}{\sqrt{2}} \times \frac{r}{\sqrt{2}}$ on the plane. Clearly, any long edge must be connecting vertices in two different cells. For each pair of cells, we remove from B all the long edges between them except for the shortest one.

Fig. 5. Construct a backbone for a given quasi-UDG

Theorem 5: The algorithm **QuasiUDG-Backbone** constructs a backbone of the given quasi-UDG G such that its maximum degree is $O(\frac{R^2}{r^2})$, the average number of edges crossing an edge is $O(\frac{R^4}{r^4})$, and the power stretch factor is $3+\epsilon$ (where ϵ is a constant that can be made arbitrarily small).

Proof: Let G' be the subgraph of G that includes all the vertices and short edges of G. It is easy to see that G' is a UDG. Therefore after Step 1 and Step 2 of the algorithm, we have removed the crossings between shortest edges, and reduced the number of short edges incident to any vertex to no more than $k + 5$, where $k > 8$ is the parameter to the algorithm [8]. Note that in Step 3, we keep at most one edge between any two cells, and the number of cells reachable from any vertex is bounded by $O(\frac{R^2}{r^2})$. The total number of long edges incident to any vertex is then bounded by the same constant. Thus in the final backbone, the degree of a node is bounded by $O(\frac{R^2}{r^2})$.

On the other hand, any edge crossing in the final backbone has to involve a long edge since the subgraph induced by short edges is planar. For an arbitrary edge e, we will bound the number of long edges that can cross it. Any long edge that crosses e must connect one cell at one side of e to another cell on the other side. We can verify that the number of cells on one side of e that can connect to a cell on the other side is $\omega = O(\frac{R^2}{r^2})$. Therefore, the number of long edges that can cross e is at most $\omega^2 = O(\frac{R^4}{r^4})$. Suppose that the total number of edges in the final backbone is m. Then the total number of edge crossings is bounded by $O(\frac{R^4}{r^4})m$. Therefore the average number of edges crossing an edge is bounded by $O(\frac{R^4}{r^4})$.

After Step 1 and 2, we have constructed a planar power spanner for G' of stretch factor bounded by $1 + 2^\beta \sin^\beta(\pi/k)$ [8]. In Step 3, by removing all the edges between any two cells C_1 and C_2 except the shortest among them, the stretch factor is increased but still bounded by $3+2^{\beta+1}\sin^\beta(\pi/k)$. To prove this bound, we only need to prove that for any edge $\{x, y\}$ of G that is removed, there is a path from u to v in the final backbone such that the ratio of the communication cost of the path and that of the edge $\{u, v\}$ is at most $3+2^{\beta+1}\sin^\beta(\pi/k)$.

If the edge $\{u, v\}$ is removed in the Step 1, we know that the communication cost between u, v did not change (because $\beta \geq 2$). Otherwise we distinguish two cases:

Case 1, the edge $\{u, v\}$ is removed in step 2. In this case, [8] guarantees that by the end of step 2, there is a path from u to v consisting of edges of length at most r and the stretch factor of the path is bounded by $1 + 2^\beta \sin^\beta(\pi/k)$. Since step 3 only removes edges of length greater than r, the above path from u to v is preserved in the backbone and the stretch of the path is bounded by $1 + 2^\beta \sin^\beta(\pi/k) < 3 + 2^{\beta+1} \sin^\beta(\pi/k)$.

Case 2, the edge $\{u, v\}$ is removed in Step 3. In this case, the length of $\{u, v\}$ is greater than r and there is another edge $\{u', v'\}$ in the final backbone such that u and u' belong to the same cell, v and v' belong to the same cell, and $d(u', v') \leq d(u, v)$. By an argument similar to that in case 1, there must exit a path between u and u' in the final backbone whose communication cost is at most $(1 + 2^\beta \sin^\beta(\pi/k))d(u, u')^\beta \leq (1 + 2^\beta \sin^\beta(\pi/k))r^\beta < (1+2^\beta \sin^\beta(\pi/k))d(u, v)^\beta$. Similarly, there is a path between v and v' in the final backbone whose communication cost is at most $(1 + 2^\beta \sin^\beta(\pi/k))d(u, v)^\beta$. Since $d(u', v') \leq d(u, v)$, the stretch factor of the path (u, u', v', v) is at most $2(1 + 2^\beta \sin^\beta(\pi/k)) + 1 = 3 + 2^{\beta+1} \sin^\beta(\pi/k)$. Note that $2^{\beta+1} \sin^\beta(\pi/k)$ can be made arbitrarily small by choosing a sufficiently large parameter k. This completes the proof. ■

IV. Applications and Performance Evaluation

In this section, we first present out routing algorithm based on the separators, then prove the bound for the path stretch factor of our routing protocol. As the second part of the section, we show the simulation results of the backbone constructions and the routing performance of our routing algorithms to verify the theoretical bounds we prove.

A. A routing scheme based on the separators

As one of the applications of the small separators of the grid graphs, we present a routing scheme for quasi-UDG based on the grid graph and analyze its performance. Our routing scheme is suitable for systems in which the size of the messages itself is relatively large. We will give the simulation results later in this section.

Our routing scheme is based on the distance labelling scheme described in [7]. The basic idea of distance labelling is to give each vertex a label such that the distance between two vertices can be computed using only their labels. A straightforward labelling scheme is to store in each node a full table of the distances to all the other vertices. The goal of the distance labelling scheme in [7] is to find the labels of minimum length. The separability of the underlying graph is a key factor of how good a distance labelling scheme is available for the network. In [7] the authors proved that for a graph which has a separator of size k, there is a distance labelling scheme of label size $O(k \log n + \log^2 n)$, and the distance between two nodes can be computed in time $O(\log n)$, where n is the number of nodes in the network.

Although a quasi-UDG G may not possess a small separator, we have proved that the grid graph H with n vertices constructed for G does have a balanced separator of size $O(\sqrt{n})$. Conceptually, our routing protocol utilizes two-level routing: virtually, the message is sent in the grid graph from the cell containing the source to the cell containing the destination, via the shortest path in the grid graph; in reality, the routing is implemented in the underlying quasi-UDG to route from cell to cell. (Note that in each cell, the quasi-UDG vertices are fully connected, so routing from one cell to the next takes at most two hops.) The basic idea to achieve shortest path routing in the grid graph is to split H into two non-adjacent parts using the small separator. Each vertex of H remembers the distance to all separator vertices. Thus, two vertices in the two parts (or the separator) can compute their shortest path distance using that information, because their shortest path must go through a separator vertex. We recursively apply the same process to partition each part into small parts, to enable any two vertices to compute their shortest path distance using their stored information (their labels). We stop partitioning a part when its size is below a certain constant. (We call such a part a *basic block*.) Since we use balanced separators, the process ends after $O(\log n)$ levels of partitioning.

For a vertex W of H, let $v(W)$ be the set of quasi-UDG vertices of G that reside in the cell corresponding to W. The following list contains the information that each vertex $u \in v(W)$ in G stores in our protocol.

- the minimum distances (in H) to all the separator vertices that are on the boundaries of all the partitions W is in;
- the neighboring quasi-UDG vertex through which it can get to other cells adjacent to W in H;
- a shortest path routing table for the vertices of H in the basic block where u resides.

The routing protocol assumes that the source knows the label of the destination. This piece of information can be obtained from location service. Since location service is not directly related to our topic, we skip the details here.

If the destination is not in the same cell as the source, the message will follow a shortest path in H from the source cell to the destination cell. By utilizing the second part of the list (label), a vertex can send a message to any of its neighboring cell in two hops. Within a basic block, the third part of the routing table points out the shortest path between cells directly. Our routing protocol compares favorably with shortest path routing algorithms and compact routing algorithms for general networks for its significantly smaller routing table size and maintained constant stretch factor.

Proof of Theorem 4

Proof: In the routing protocol described above, the first part of a node's routing table is of size $O(\sqrt{N} \log N)$. The second and third parts of the routing table both consist of a constant number of entries because the number of neighboring cells and the number of cells in each basic block are both constants. The size of the routing table is then $O(\sqrt{N} \log N)$. Inside each message we need only to carry the label of the destination vertex, so the overhead in the message size is also bounded by $O(\sqrt{N} \log N)$.

Given a path p from u to v, let $d(p)$ denote its number of hops, and let $c(p)$ denote the number of times the path p travels from one cell to another. Let p_{opt} be the shortest path from u to v, and let p' be the routing path of our protocol. Clearly, $c(p_{opt}) \leq d(p_{opt})$, and $c(p') \leq c(p_{opt})$ because our protocol uses shortest path routing in the grid graph. p' travels from one cell to the next in at most two hops, so $d(p') \leq 2c(p') + 1$. So $d(p') \leq 2d(p_{opt}) + 1$. ■

Sometimes we are more concerned about the energy consumption than the hop distance if the wireless nodes are able to adjust their communication range to save power. Let the communication cost be as defined in Section III. In reality, it is infeasible for a node to reduce its communication range to infinitely small. There is always a constant range δ below which the wireless node cannot reduce its communication range. With this assumption, we prove the following theorem.

Theorem 6: Let the communication cost be as defined in Section III, and assume that the minimum communication range is δ. (Therefore, the communication cost of an edge of Euclidean length d is $\alpha \cdot (\max\{d, \delta\})^\beta$.) Then, the communication cost of a routing path from u to v generated by our routing protocol is upper bounded by a constant times the minimum communication cost over all the paths from u to v.

Proof: Let p_{opt} be the optimal path from u to v with the minimum communication cost C_{opt}, and let p' be the routing path of our algorithm with cost C'. If u, v are in the same cell of the grid graph H, then $C_{opt} \geq \alpha\delta^\beta$, and $C' \leq \alpha r^\beta$ since vertices in the same cell form a clique. So $C' \leq (r^\beta/\delta^\beta)\alpha\delta^\beta \leq (r^\beta/\delta^\beta)C_{opt} = C_{opt} \cdot O(1)$.

Now assume that u, v are in different cells of H. Let l_{opt} and l' denote, respectively, the number of hops in p_{opt} and p'. By Theorem 4, $l' \leq 2l_{opt}+1$. So $C' \leq l'\alpha R^\beta \leq (2l_{opt}+1)\alpha R^\beta \leq \frac{2l_{opt}+1}{l_{opt}} \cdot \frac{R^\beta}{\delta^\beta} \cdot l_{opt}\alpha\delta^\beta \leq 3 \cdot \frac{R^\beta}{\delta^\beta} \cdot C_{opt} = C_{opt} \cdot O(1)$. ■

B. Simulations

We conducted extensive simulations to evaluate the performance of our backbone construction algorithm and routing protocol. The performance has been stable and consistent. In the following experiment, we randomly deploy N quasi-UDG nodes in a 2-D space of size 1500×1500. We increase the number of nodes, N, in the system from 1000, 1500 to 2000

to verify the effects of density change on the performance. We also increase the value R/r from $1, 1.5, 2, 3$ to 10 to see the performance of our algorithms for different wireless connectivity models. To mimic nontrivial network topologies, we randomly generate a big hole of radius randomly picked in the range $[R, 2R]$ and five small random holes of radius picked in the range $[0, R]$. The centers of the holes are uniformly randomly chosen in the plane. If the distance between two nodes is in the range $(r, R]$, we put a direct link between them probabilistically. For each configuration, we run the simulation 30 times and take the average of the performance metrics.

We would like to point out that the performances of our routing algorithm and backbone construction method are relatively independent of the size of the network. Our theoretical bounds and the simulation results both show that the quality of the backbone constructed and the stretch of the routing paths are closely related to the ratio of r to R.

1) Backbone construction: In the backbone construction simulations, we measure the power stretch factor, maximum degree, the average degree and the average number of edge crossings on an edge in the backbone constructed and compare them to the original graph. For node pairs whose distances are between r and R, we adjust the probability of their being connected to ensure an expected average degree of 10 in order to compare the results between different densities and values of R/r. The results shown in Table I and Fig. 6 are for backbones constructed by only performing the first step and the last step in Algorithm **Backbone**. We eliminated the results for the case when $R = r$ since there the backbones are known to be planar with power stretch factor being 1 because of the Gabriel operation in the first step of the algorithm. Our results showed that for all configurations the backbones have very small power stretch factors, much smaller maximum degree and in most cases, we can bring the average degree to below 6 (which is the upper bound of the average degree for planar graphs). Even when $R/r = 10$, the average degree of our backbones is no more than 8. As for the number of crossings, our algorithm reduced it by at least 60% in all cases.

TABLE I

POWER STRETCH FACTOR FOR THE BACKBONES($\beta = 2$)

	Stretch Factor			
N	R/r=10	R/r=3	R/r=2	R/r=1.5
1000	1.048	1.141	1.190	1.184
1500	1.044	1.155	1.198	1.129
2000	1.046	1.176	1.239	1.204

From the three bar graphs in Fig. 6, the reduction in the metrics is quite uniform. It implies that the performance of our algorithm is stable for different sizes of the network.

2) Routing performance: We apply our routing protocol not only to the original quasi-UDGs but also to the backbones we obtain. To study the performance, we measure the maximum label size, average label size and the stretch factor of routing path that is defined as the distance in the actual routing path over the shortest path between the source and the destination.

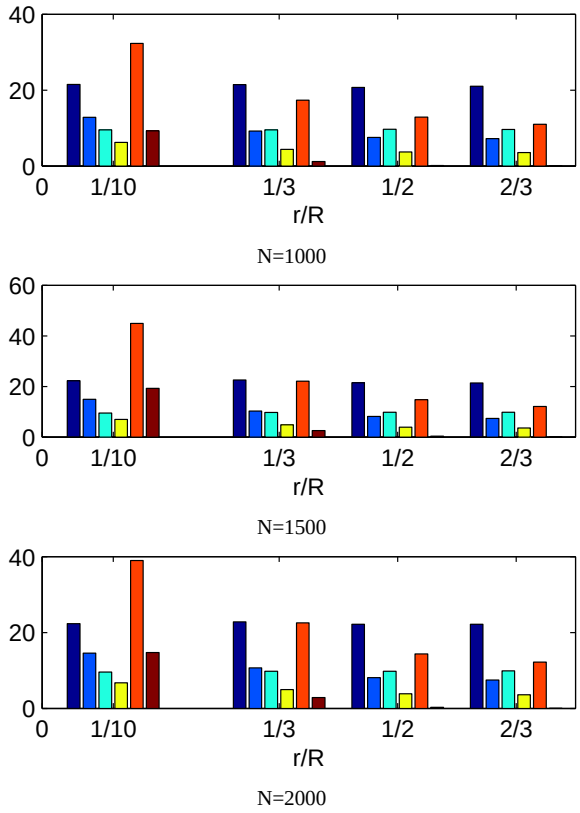


Fig. 6. The Maximum degree, the average degree, and the average number of edges crossing an edge for quasi-UDGs and their backbones. The 6 bars in each group are, from left to right (1) maximum degree in quasi-UDG; (2) maximum degree in backbone; (3) average degree in quasi-UDG; (4) average degree in backbone; (5) average number of crossings per edge in quasi-UDG; (6) average number of crossings per edge in backbone. Note that in some groups, the last bar is not shown, because the average number of crossings per edge in backbones equals 0 there.

The length of the path for routing in the original graphs is defined as the hop distance between two nodes, while in the backbones, we use the communication cost with $\beta = 2$ as the length of the path. In both cases we randomly pick 1000 source-destination pairs in the graph, simulate the routing process and compare the length of the path with the shortest. Due to the page limit we only present the results on the quasi-UDG with expected degree 10 and remark that the results are consistent for graphs with other edge densities.

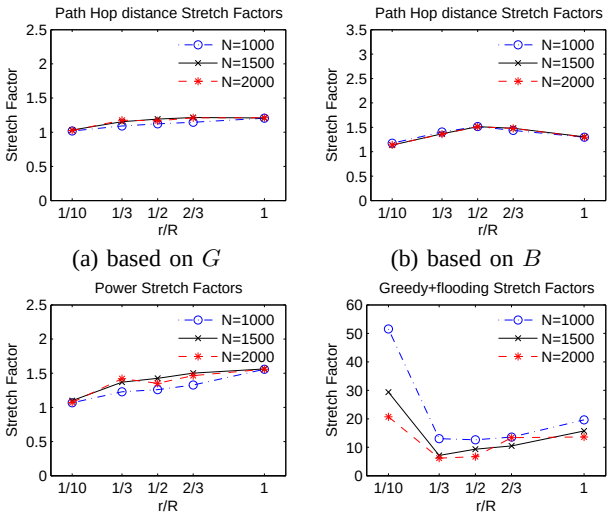
Table II shows the average values of the maximum label size and the average label size (with a node ID as a unit) over the experiments for two cases. We observe that the label sizes with the algorithm applied to the backbones are smaller than those to the original graphs. This is mainly because the backbones are sparser than the original quasi-UDGs, hence the grid graphs we get are also sparser and have smaller separators. We will see later that this advantage comes at a cost of slightly larger stretch factors.

Fig. 7(a) shows the average hop distance stretch factors of the routing path for the routing algorithm applied to the original graphs directly. In all cases we have the path stretch

TABLE II

LABEL SIZES OF ROUTING SCHEME BASED ON SEPARATORS

		On original		On backbone	
N	R/r	Max Size	Avg Size	Max Size	Avg Size
1000	10	220.667	155.911	184.800	131.614
1000	3	139.733	106.553	130.733	89.561
1000	2	129.933	91.318	97.434	68.825
1000	1.5	100.367	72.960	90.634	60.562
1000	1	75.834	55.876	72.434	51.231
1500	10	325.900	233.056	287.567	205.997
1500	3	218.933	152.448	166.067	121.646
1500	2	165.767	115.528	143.400	90.668
1500	1.5	133.900	91.548	122.967	78.739
1500	1	102.167	72.627	90.800	63.852
2000	10	320.033	243.665	292.733	219.245
2000	3	232.100	172.638	219.200	151.097
2000	2	196.500	142.079	151.400	102.410
2000	1.5	271.500	224.919	124.433	86.575
2000	1	115.867	84.759	108.933	74.227



(a) based on G (b) based on B

(c) based on B (d) Greedy+flooding

Fig. 7. Stretch factors for routing algorithms. G is the original graph, and B is the backbone.

factors no larger than 1.3.

Fig. 7(c) shows the power stretch factors and Fig. 7(b) shows the hop distance stretch factors of the routing paths when the algorithm is applied to the backbones. The hop stretch factors shown in Fig. 7(b) are moderately larger than the ones shown in Fig. 7(a). It is the price we paid for the reduction in the size of the routing tables.

It looks interesting from the figures that when R/r is large(10), the algorithm generally performs better than the other cases. This is because to maintain the same average node degree of the graphs we have to decrease the value of r. In that case a grid graph actually describes the original graph more accurately and with more details. Hence the sizes of the labels are larger(see Table II), but the paths we discovered are closer to the shortest ones.

We have also implemented the well known greedy-forwarding plus local-flooding (expanding ring search with doubling radius) routing algorithm, and performed the same number of experiments on the same set of graphs. The average stretch factors are shown in Figure 7(d). Our results indicate that compared to that algorithm, the routing protocol based on separators has a much better stretch factor because of its robustness to the existence of holes.

V. CONCLUSION

In this paper, we have studied two structural properties of quasi-UDGs: separability and the existence of power efficient spanners. Such results lead to a deeper understanding of the locality properties of quasi-UDG networks and an improvement in the development of networking protocols. As the future work, we will explore the separability of quasi-UDGs deployed in 3D space, other properties of quasi-UDGs, and their network applications.

REFERENCES

[1] K. ALZOUBI, X. LI, Y. WANG, P. WAN AND O. FRIEDER, Geometric spanners for wireless ad hoc networks, *IEEE Trans. Parallel and Distributed Systems*, vol. 14, no. 4, pp. 408-421, 2003.

[2] L. BARRIÈRE, P. FRAIGNIAUD AND L. NARAYANAN, Robust position-based routing in wireless ad hoc networks with unstable transmission ranges, *Proc. of the 5th international workshop on Discrete algorithms and methods for mobile computing and commnunications(DialM '01)*, pp. 19-27, (2001).

[3] P. BOSE, P. MORIN, I. STOJMENOVIC, AND J. URRUTIA, Routing with guaranteed delivery in ad hoc wireless networks, *Proc. of the 3rd international workshop on Discrete algorithms and methods for mobile computing and commnunications(DialM '99)*, pp. 48–55, 1999.

[4] Q. FANG, J. GAO AND L. J. GUIBAS, Landmark-based information storage and retrieval in sensor networks, *Proc. of INFOCOM'06*, 2006.

[5] K. GABRIEL, AND R. SOKAL, A new statistical approach to geographic variation analysis. *Systematic Zoology*, 18:259-278, 1969.

[6] D. GANESAN, B. KRISHNAMACHARI, A. WOO, D. CULLER, D. ESTRIN, AND S. WICKER, Complex behavior at scale: an experimental study of low-power wireless sensor networks. *Technical Report UCLA/CSD-TR 02-0013, UCLA, 2002*.

[7] C. GAVOILLE, D. PELEG, S. PÈRENNES, AND R. RAZ, Distance labeling in graphs, *Journal of Algorithms 53(1)*, pp.85-112, 2004.

[8] I. A. KANJ AND L. PERKOVIC, Improved stretch factor for bounded-degree planar power spanners of wireless ad-hoc networks. *To appear in the proceedings of ALGOSENSOR'06.*

[9] B. KARP AND H. T. KUNG, GPSR: Greedy perimeter stateless routing for wireless networks, *Proc. of the 6th annual international conference on Mobile computing and networking*, (2000), pp. 243-254.

[10] Y. KIM, R. GOVINDAN, B. KARP AND S. SHENKER, Geographic routing made practical, *Proc. of the 2nd USENIX/ACM Symposium on Networked System Design and Implementation (NSDI 2005), Boston, MA*, (May, 2005)

[11] R. J. LIPTON AND R. E. TARJAN, A separator theorem for planar graphs, *SIAM Journal on Applied Mathematics, Vol. 36, No. 2*, (1979), pp. 177-189.

[12] F. KUHN AND A. ZOLLINGER, Ad-hoc networks beyond unit disk graphs. *Proc. 2003 joint workshop on Foundations of mobile computing*, pages 69–78, 2003.

[13] K. SOHRABI, B. MANRIQUEZ AND G. POTTIE, Near ground wideband channel measurement. *IEEE Vehicular Technology Conference*, vol. 1, pp. 571–574, 1999.

[14] W.-Z. SONG, X.-Y. LI, Y. WANG, AND O. FRIEDER, Localized algorithms for energy efficient topology in wireless ad hoc networks, *Mobile Networks and Applications*, 10(6):911-923,2005.