Energy-Efficient Face Routing on the Virtual Spanner[⋆]

Héctor Tejeda[1], Edgar Chávez[1], Juan A. Sanchez[2], and Pedro M. Ruiz[2]

[1] Escuela de Ciencias Físico-Matemáticas, Universidad Michoacana, México
{elchavez, htejeda}@fismat.umich.mx
[2] Facultad de Informática, University of Murcia, Spain
{jlaguna, pedrom}@dif.um.es

Abstract. Geographic routing protocols are one of the most common routing schemes for sensor networks. These protocols consist of two different modes of operation: greedy routing to forward data to the destination using neighbors which are closer to the destination than current node and face routing to avoid voids in the network. Face routing requires the graph to be planar, which usually means that some crossing links of the original network cannot be considered when routing in face mode. In this paper we introduce a new localized scheme to build a virtual spanner which is planar by construction and is guaranteed to be connected if the underlying network is connected as well. Unlike previous works, by performing face routing over this spanner we can reduce energy consumption in face mode because the elimination of any of the original links in the network is not required. Thus, the most energy-efficient paths can be selected when the protocol enters face mode. The virtual spanner is easy-to-build and uses only local information, making it scalable to large-scale networks. Routing is always performed in real nodes; virtual nodes are used only as routing anchors when the agent is in face mode. In addition, our simulation results show that the proposed scheme outperforms the best energy-efficient geographic routing protocol for different network densities and energy models.

1 Introduction and Related Literature

Mobile ad hoc networks (often referred to as MANETs) as well as wireless sensor networks have become one of the hotest topics in communications. One of the main reasons is their wide applicability to many scenarios such as disaster relief, battlefield environments, etc. These networks consist of wireless nodes that communicate with each other in the absence of a fixed infrastructure. When a node needs to send a message to another host which is outside of its radio range, it uses other intermediate hosts as relay nodes. Those intermediate nodes are dynamically selected by the routing protocol according to some metric.

Among all routing protocols for these networks, geographic routing [BMSU01] has emerged recently as a very efficient way to provide guaranteed delivery routes

[⋆] Partially supported by CONACyT and the Spanish MEC by means of the Ramon y Cajal work program and the SAVIA (CIT-410000-2005-1) project.

without flooding the whole network with control messages. A node is only required to be able to know its position, the one of its neighbors (e.g. using periodic becons) and the position of the destination. When a node wants to send a message to some destination, each intermediate node selects locally its *best* neighbor to forward the message towards that destination. The best node depends on the particular routing metrics. For instance, in networks with battery-operated devices, energy consumption is one of the most important. This operation of selecting the best neighbor is called *greedy mode*.

There has been a number of energy-efficient geographic routing solutions reported in the literature. Stojmenovic and Lin [SL01] proposed localized power-aware algorithms based on the consideration of different neighbor selection schemes in greedy mode. For instance, one of the proposed schemes was selecting the nearest neighbor to current node among those providing advance towards destinations. More recently, those schemes were enhanced in [KNS04] by considering a cost over progress neighbor selection. The cost is determined by the energy to reach a candidate neighbor while the progress is the reduction in distance towards the destination by selecting that neighbor. The neibghbor with lowest ratio is selected at each step. The best solution among all alternatives reviewed is iPowerProgress, which uses the same idea but iteratively enhances the selection by trying to find another neighbor so that reaching the selected next hop through that neighbor reduces the overall energy required.

However, regardless of the greedy function used to select neighbors, all of them end up using face routing [BMSU01] to escape from local minima (e.g. a void zone). Face routing sends packets along the faces of a planar graph representing the underlying network graph following the right hand rule, and changing faces when the line between the node where face started and the destination is crossed. Correct operation of face routing requires the underlying graph to be planar. Thus, some of the original (crossing) edges are not considered while in face mode because they are eliminated by planarization tests.

There are several methods to extract a planar subgraph from a given Unit Disk Graph (UDG), which models the entire network. A UDG is a graph in which an edge $[u, v]$ exists only if $dist(u, v) \leq r$ being r the radio range. The *Relative Neighborhood graph*, RNG [Tou80] is obtained by applying the RNG test to every edge of the UDG: an edge $[u, v]$ is retained in $RNG(G)$ if there is no vertex z such that $\max\{d_G(u, z), d_G(v, z)\} < d_G(u, v)$. That is, if there is no vertex in the intersection of their disks. The *Gabriel graph*, GG [GS69], applies a slightly different test to every edge of the graph. It retains an edge $[u, v]$ in $GG(G)$ if there is no node in the disk with diameter $\overline{uv}$. Finally, the *Morelia test* [BCG$^+$04] manages to preserve some long edges by using a stronger condition for the removal of edges. An edge $[u, v]$ is not included in $MG(G)$ if there is a couple of points $[x, y]$ so that one of them (or both) is in the disk with diameter $\overline{uv}$ and $[x, y]$ crosses $[u, v]$. Given a UDG G we have $RNG(G) \subseteq GG(G) \subseteq MG(G)$.

Energy-efficient optimizations for geographic routing reported to date, have just focused on the greedy part of the protocol. In this paper, we propose the creation of a planar virtual spanner of the original graph using a tesselation planar by construction. Face routing will then be performed on the virtual graph. Once the next hop virtual neighbor is selected according to face routing, real nodes will route the message greedily towards the representative (e.g. centroid) of the selected neighboring tessel. Given that real nodes will route using all available links (no links are eliminated), it is possible to find an energy-efficient path towards that representative which enhances the performance in face mode of the protocol. In our case, we try both iterative PowerProgress [KNS04] and Dijkstra's shortest path (applied to the local neighborhood based on energy cost) to reach the representative of the next tessel using energy-efficient links. We shall see in our simulation results how the use of this virtual spanner manages to reduce energy consumption for all network desnsities tested.

The remainder of the paper is organized as follows: Section 2 presents our network model and the problem formulation. Section 3 illustrates how the virtual spanner is built. We explain how to route based on the virtual spanner in section 4. Finally we present some simulation results in section 5 and give some conclusions and future work in section 6.

2 Network and Energy Model

This section introduces the notation and the model we use throughout the paper. We consider routing algorithms on Euclidean graphs. As usual, a graph G is defined as a pair $G := (V, E)$ where V denotes the set of vertices and $E \subseteq V^2$ denotes the set of edges. The number of nodes is denoted by $n := |V|$.

In our evaluations we will use the general energy model proposed by Rodoplu and Meng [RM98] in which the power consumption between two nodes at distance d is $u(d) = d^\alpha + c$ for some constants $\alpha (2 \leq \alpha \leq 4)$ and c. We also consider a more realistic model for some devices which also accounts for the energy of reception. $u(d) = d^\alpha + c + c_r$, where c_r is usually $(r^\alpha + c)/3$ being r is the maximum transmision range.

In this paper we consider the standard UDG model for ad-hoc networks, but we assume that nodes can asjust their transmission range (r) to reach only a subset of their neighbors. Thus, given two nodes $v_1, v_2 \in V$, the edge $[v_1, v_2] \in E \Leftrightarrow ||[v_1, v_2]|| \leq r$, being $||[x, y]||$ the euclidean distance between vertices x and y.

As in previous geographic routing works in the literature, we assume that nodes know their positions and those of their neighbors. It is also assumed that sources of data packets know the position of the destination.

3 The Virtual Spanner

We explain in this section, how each node locally builds its partial view of the virtual spanner. For clarity, we first explain the overall idea and its topology.

We divide the plane in regions with a regular tessellation, which is a tessellation (or planar subdivision) made up of congruent regular polygons. The idea is that an entire region may be represented by a single virtual point, the centroid of the regular polygon. If we link the centers of the polygons we observe a peculiar behavior: the centers define a dual tessellation that is also planar. The dual of a triangle tessellation is a hexagonal tessellation, while a square tessellation is auto dual.

Only three regular polygons tessellate the Euclidean plane: triangles, squares or hexagons, from elementary geometry. They are depicted in figure 2.

The virtual node for a polygon is chosen as the centroid of the polygon. Two virtual nodes will share an edge if in their respective cells two real nodes are mutually reachable. Thus, we need to choose a suitable polygon size to build the virtual graph, so that we achieve a good trade-off between the simplicity to build the virtual graph (guaranteeing that is planar), and the number of cells to be checked for possible links to preserve connectivity. We have analyzed three options as we show in figure 1.

a) The transmission radius does not cover all the cell.
b) Any two points in the cell are within radio range.
c) A node in one cell can reach any other node in a neighboring cell.

Case a) complicates the design because it may require multihop routing within a cell. In case c) there may be a very big number of cells in which to look for neighbors. We decided to use case b) because it is the configuration which avoids multihop within a cell in which the number of cells to look for virtual neighbors is reduced.

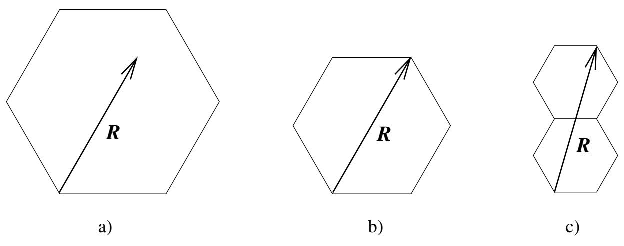
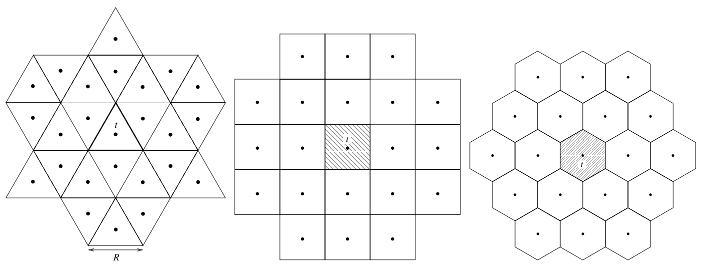


Fig. 1. Variation of polygon size with transmission radius fixed

If the graph is dense enough, there will be at least one node in each cell. Thus, each virtual node will be connected to all neighboring virtual nodes. The resulting virtual graph is exactly the dual of the graph, which is planar. In real situations we cannot guarantee that every cell will have a node. Thus, to preserve connectivity we must find all possible virtual neighbors. They may be in cells which are not contiguous to the current one. In figure 2 we show for each different tesselation (triangular, square and hexagonal) the possible cells that may contain real nodes which are neighbors of nodes in the current cell t. The cell t can reach more cells when using a triangular configuration (24 cells). With a square configuration 20 cells are candidates and when using hexagonal cells only 18 cells are candidates. Please note that adding virtual links to all those cells having real nodes which are neighbors of nodes in current cell may result in

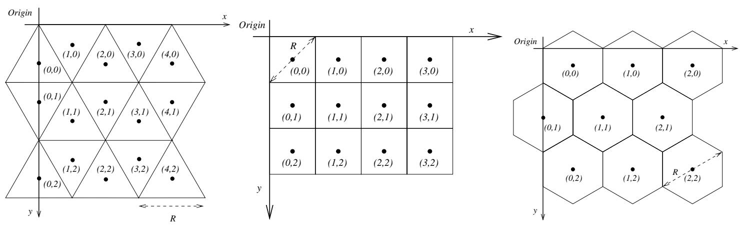


(a) Triangles (b) Squares (c) Hexagons

Fig. 2. Regular tessellations in the plane and cell centroids. Additionally, the cells shown are those reachable from t using a radius R equal to the diameter of the cell.

a non-plannar virtual spanner. Unlike other traditional planarity tests, crossing links can be detected and avoided using a static table containing all the possible configurations.

The grid with triangles, squares or hexagons is located arbitrarily in the plane. Each cell is identified by a coordinate pair as we show in Figure 3. Note that real nodes only need to know the type of tesselation and the communication radius at deployment time. Based on that, and given their current position they can easily compute the coordinates of their centroid. In addition, only with local information about the position of its neighbors they can compute their local view of the virtual graph (virtual edges). This has no additional overhead because position of real neighbors is already required for geographic routing to operate.



(a) Triangles (b) Squares (c) Hexagons

Fig. 3. Fixing the origin, the virtual coordinates are computed with elementary calculations

We show below some elementary calculations for the node to compute the coordinates of the virtual node for each real node and the tesselation to which it belongs. The node only needs the transmission radius R and its position (x, y).

Once a node computes its tessel and its centroid, it individually and locally computes its view of the virtual graph. The process consists of two different stages:

Type of tilling	Position of centroid	Tessel

Hexagonal	$y_c \leftarrow 3R(y+1/3)/4$
	if $y \bmod 2 = 0$
	then $x_c \leftarrow \sqrt{3}R(x+1/2)/2$
	else $x_c \leftarrow \sqrt{3}Rx/2$
		$y \leftarrow$**truncate** $(3y_n/4/R)$
		if $y \bmod 2 = 0$
		then $x \leftarrow$**truncate** $(2x_n/\sqrt{3}/R)$
		else $x \leftarrow$**truncate** $((2x_n+\sqrt{3}R/2)/\sqrt{3}/R)$

Triangular	$x_c \leftarrow Rx/2$
	if $(x+y) \bmod 2 = 0$
	then $y_c \leftarrow \sqrt{3}R(y+2/3)/2$
	else $y_c \leftarrow \sqrt{3}R(y+1/3)/2$
		$x \leftarrow$**truncate** $(x_n/2/R)$
		$y \leftarrow$**truncate** $(2y_n/\sqrt{3}/R)$

Square	$x_c \leftarrow (x+1/2)R/\sqrt{2}$ $y_c \leftarrow (y+1/2)R/\sqrt{2}$	$x \leftarrow$**truncate** $(\sqrt{2}x_n/R)$ $y \leftarrow$**truncate** $(\sqrt{2}y_n/R)$

Table 1. Formulas for a node to compute position of its centroid and its tessel

1. Test surrounding cells that are neighbors by their side.
2. Test all other cells that are reachable from current cell but are not neighbors by their side.

The first stage is easier than the second one because it always produces a planar graph. There are no edge crossings, as it is depicted in figure 4. In the first stage, a virtual edge is added between centroids of two cells adjacent by the side if there are two mutually reachable real nodes, one in each of those cells. Unfortunately, the virtual graph produced after the first stage may not be connected (see Fig. 4). Thus, we need to apply the second stage to obtain a connected graph without crossings of virtual edges.

For the second part, we start testing if we can add a virtual edge to the centroid of those cells (see figure 2) which are second degree neighbors (side neighbors of our side neighbors). If for one of those, we cannot add the virtual edge (i.e. there is no other real node in that cell directly reachable from any real node in current cell) then we try again with those cells being neighbors by side of this particular cell we couldn't find nodes to add the virtual edge. This condition guarantees that the resulting connected virtual graph will be planar, because each of those edges is tested not to cross previous edges before being added. Figure 5 shows the resulting virtual graph after both stages.

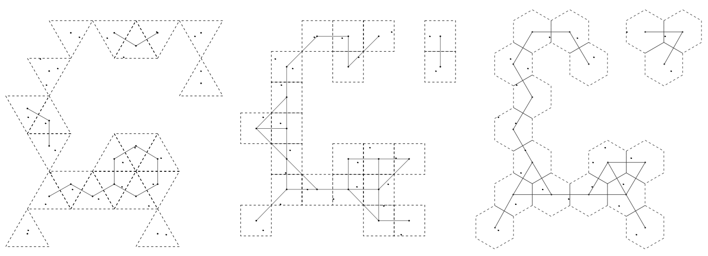

(a) Triangles (b) Squares (c) Hexagons

Fig. 4. First connectivity test. Natural neighbors.

As an example, we give the concrete algorithms used in each stage to add edges to the virtual spanner with an hexagonal tilling. The algorithms for squares and triangles are similar and are not included in here due to space limitations.

The algorithm for the first connectivity test using hexagons is given in algorithm 1.

Algorithm 1. Algorithm for the first stage with hexagons

```
procedure ReviewHexagonsStage1($I, t$)        ▷ $I$ are the neighbor cells
    $k \leftarrow 0$        ▷ by their side to $t$, they are enumerated from 0 to 5
    while $k < 6$ do
        if isThereEdge($t, I_k$) then        ▷ real edge among nodes of those tessels?
            addEdge($t, I_k$)
        end if
        $k \leftarrow k + 1$
    end while
end procedure
```

Algorithm 2. Algorithm for the second stage with hexagons

```
procedure ReviewHexagonsStage2($I, E, t$)        ▷ $E$ are the rest of the cells
    $k \leftarrow 0$        ▷ reachable by $t$, enumerated from 0 to 11
    while $k < 6$ do        ▷ Review for the odd cells from $E$
        $a \leftarrow 2k$
        $b0 \leftarrow$!isThereEdge($t, I_k$)
        $b1 \leftarrow$!isThereEdge($I_k, E_a$)
        $b2 \leftarrow$isThereEdge($t, E_a$)
        if $b0$ AND $b1$ AND $b2$ then
            addEdge($t, E_a$)
        end if
        $k \leftarrow k + 1$
    end while

    $k \leftarrow 0$
    while $k < 6$ do        ▷ Review for the even cells from $E$
        $a \leftarrow (k + 1)$ mod 6
        $b \leftarrow (2k + 1)$ mod 6
        $b0 \leftarrow$isThereEdge($t, I_k$)
        $b1 \leftarrow$isThereEdge($I_k, E_b$)
        $b2 \leftarrow$isThereEdge($t, I_a$)
        $b3 \leftarrow$isThereEdge($I_a, E_b$)
        $b4 \leftarrow$isThereEdge($t, E_b$)
        if !($b0$ AND $b1$) AND !($b2$ AND $b3$) AND $b4$ then
            addEdge($t, E_b$)
        end if
        $k \leftarrow k + 1$
    end while
end procedure
```

For the second stage with an hexagonal tilling, all reachable cells which are not side neighbors of the current cell (t) are tested. The test needs to take into account existing virtual links which have been added before, to avoid creating a non-planar virtual spanner. The detailed algorithm is given in algorithm 2.

As we stated, the goal of this virtual graph is enhancing the energy consumption of face routing. Thus, we will explain in the next section how real nodes perform energy-efficient face routing over the virtual graph.

4 Energy Efficient Face Routing on the Virtual Spanner

When the geographic routing protocol finds a void zone, we perform face routing over the virtual spanner. In this section we explain how real nodes route those messages in face mode to reduce energy consumption.

When a real node enters into face mode, it locally builds its view of the virtual spanner. Then, it applies face routing algorithm considering its centroid as the current node, and all other centroids connected through virtual edges as neighbors. The application of face routing gives a new virtual neighbor, representing the tessel to which the message needs to be sent in face mode. Now, real nodes are used to create a path towards that tessel. In addition, given that all edges in the original graph are preserved, we can try to find the path which minimizes energy consumption. In particular, we have considered two different alternatives to compute that path:

- Use greedy routing with an energy metric to go towards the closest node in the selected tessel.
- Compute Dijkstra's shortest weighted path algorithm considering energy as cost using local information.

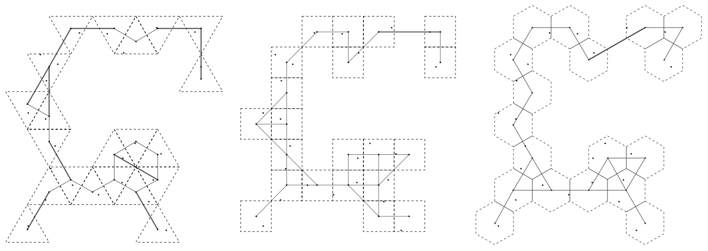
A brief description of the proposed algorithm is presented below. At each step of the algorithm the node currently trying to send the packet to the next neighbor in face mode performs the following operations:

1. Based solely on its coordinates, the node finds out his cell and corresponding virtual node according to expressions given before.
2. Using the information from neighbors (obtained by any geographic routing protocol using periodic beacons), the node finds out which virtual edges exist according to the procedures explained in the previous section. As we explained before, a virtual edge can only exist to a virtual node if there is a real neighboring node in the corresponding cell.
3. In *face mode* the current real node routing in face mode will use the virtual graph to select (according to face routing) the proper virtual edge to follow. Once it is selected, it uniquely defines the next cell that needs to be reached using *real nodes*. The node then computes the shortest path tree in energy consumption towards the nearest real node in the next cell. If real nodes of the selected cell are not directly reachable by current real node, it computes the the shortest energy consumption path towards the most distant real node

in its own cell in the direction of the next cell. So when the message reaches that node it can continue computing the shortest energy consumption path to reach some node from the next cell. To route along the lowest energy path, we use either iterative power progress or Dijkstra's shortest path tree.

4. Once a real node in the destination cell (the next cell in the path) received the packet it will forward the packet by repeating the process.
   Inside a cell the packets can be forwarded greedily because all nodes in a cell are mutually reachable.

Figure 6 shows an example of the application of the algorithm above. The source node and the target are labeled as 22 and 15 respectively. With greedy routing the packet can only travel until node 2. At that point the algorithm switches to face mode using the left hand rule. Therefore, the packet is sent to the nearest node in the virtual cell $(1, 0)$ using iterative power progress or finding the



(a) Triangles (b) Squares (c) Hexagons

Fig. 5. Second test. Reachable neighbors.

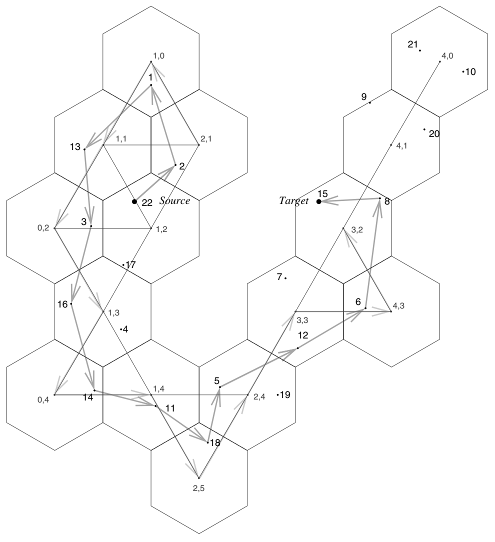


Fig. 6. Face routing using virtual hexagonal graph

local shortest weighted path tree. The face routing in the virtual nodes demands reaching cell $(1, 1)$, hence a real node in that cell must be selected. The rest of the example follows similarly. The sequence of virtual nodes, and the sequence of real nodes are depicted in figure 6.

5 Performance Evaluation

We used *connected* random unit disk graphs for our simulations. We considered scenarios consisting of 1000 nodes. The network density is defined as the average number of neighbors per node, which was varied between 4 and 18 with increments of 2. For each density the nodes were placed randomly in the simulation area and 100 different graphs were generated. The size of the simulation area was adapted to preserve the density of the network. Finally, for every graph, we selected 100 different (source,destination) pairs. Thus, each point in the graph represents the average over 10000 routing tasks.

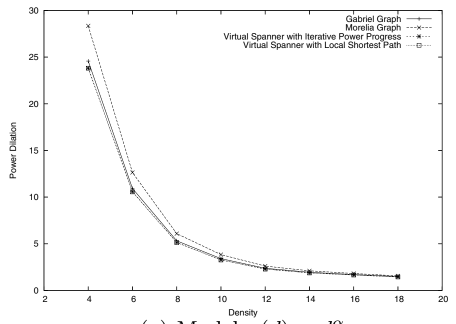
The performance metric we have used to compare our algorithm is the total amount of power required to successfully route a message from the source to the destination. We define *power dilation* as the ratio of the power requirement of the specific algorithm to that of Dijkstra's shortest weighted path algorithm applied over the whole network.

We present the results of our simulations for different densities and energy models based on the general model $u(d) = d^\alpha + c + c_r$, by just adjusting the values of constants α, c and c_r.

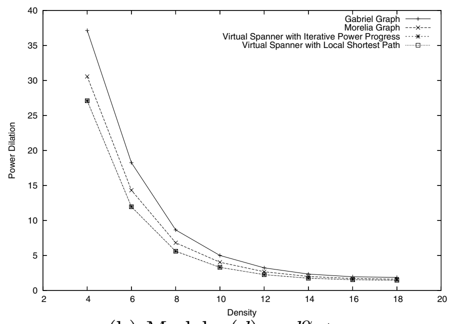
Figure 7 shows the power dilation of the proposed schemes as the density of the network increases. As we can see in all three graphs the lower density, the better the performance of our virtual spanner compared with the traditional Gabriel Graph (GG)[GS69] and the more recent Morelia Graph (MG)[BCG$^+$04] planar tests. Neither of the above graphs can guarantee energy optimality of the preserved links. Morelia Graph will preserve longest links to diminish hop distance (perhaps along with short links, but without certainty), while Gabriel Graph cannot guarantee neither long nor short links.

For the GG the density of the underlying unit disk graph will drive the preserved links, for the sparse case most of the links will be preserved. For the dense case only very short links are preserved which are suboptimal when $c > 0$ because a large number of relays are required. For $c = 0$ all the schemas will be nearly the same because short links will be present. Since MG cannot guarantee short edges the energy consumption in this model will be higher for all densities.

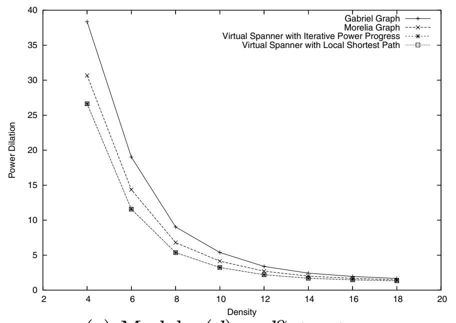
The virtual graph, by considering all possible links when finding the shortest path route to the selected node in face mode, is particularly better for low density networks because that is the case when energy can be reduced by using multiple short links rather than a single (eventually long) link. For higher densities, the spanner is also better, because it can use energy efficient links, without using too short edges. However, the overall improvement in dense networks is lower because when density increases, the amount of routing performed in face mode is very low.



(a) Model $u(d) = d^\alpha$



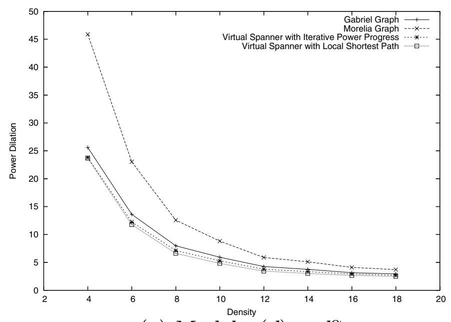
(b) Model $u(d) = d^\alpha + c$



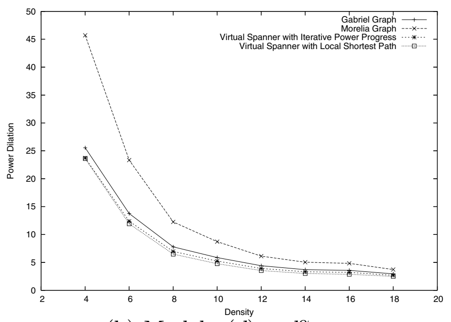
(c) Model $u(d) = d^\alpha + c + c_r$

Fig. 7. Comparing the efficiency of the Virtual Spanner vs Gabriel Test and Morelia Test when $r = 250$, $\alpha = 2$ and $c = 10000$. Lower is better.

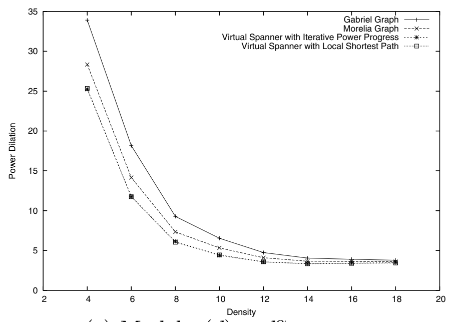
If we compare Fig. 7(a), Fig. 7(b) and Fig. 7(c) we see that the more detailed the energy model, the better the improvement achieved by the virtual spanner both with greedy routing or Dijkstra's algorithm. The reason is that when $c = 0$ and $c_r = 0$ the shorter the edges, the lower the energy consumption. Thus, GG performs particularly well for higher densities because only very short edges are preserved. In fact, in that case our protocols only manage to improve over GG a little bit in low density networks, where links preserved by GG may be longer and face routing may select a link whose cost is higher than the use of two or more links to reach the same node. However, when $c > 0$, GG is suboptimal at increasing density because the reduction of d does not compensates the additional number of transmissions that are needed to reach the destination (each of them having a cost of at least c units). We can see that for energy models in which $c > 0$ our proposed scheme is able to offer up to a 40% energy reduction compared to the same protocol which currently uses face routing based on GG and is reported in the literature as the best solution to date. MG is in disadvantage in the simplest model but as the model becomes mode accurate it outperforms GG because the longer edges preserved achieves a better trade-off in terms of energy.



(a) Model $u(d) = d^{\alpha}$



(b) Model $u(d) = d^{\alpha} + c$



(c) Model $u(d) = d^{\alpha} + c + c_r$

Fig. 8. Comparing the efficiency of the Virtual Spanner vs Gabriel Test and Morelia Test when $r = 250$, $\alpha = 4$ and $c = 10000$. Lower is better.

To analyze the impact of the radio attenuation factor (α) we conducted experiments in which $\alpha = 4$. As we see in figure 8, again a lower density produces a higher difference across approaches for the reasons given above. However, the difference in performance is reduced compared to the previous case. The reason is that with $\alpha = 4$, again reducing distance becomes more effective (i.e. relative importance of the constants c and c_r compared to d^{α} is lower). However, as we see in Fig. 8(a), 8(b) and 8(c), our proposed scheme still outperforms the GG and MG for all densities.

6 Conclusions

We have presented the first localized scheme to reduce energy consumption in face mode for geographic routing protocols. Our strategy is to retain all the links in the wireless network, giving the protocol the ability to select energy-efficient paths even in face mode. Links are retained by using a regular tesselation of the plane, and defining a virtual planar spanner superimposed to the underlying network graph. A virtual link is defined between two cells if there is at least a node in each of them so that those nodes are within radio range one of another. The virtual spanner is used to perform face routing, selecting virtual nodes according to the traditional face routing algorithm. Nevertheless the routing is performed in the underlying network all the time, virtual nodes are used only as routing anchors. Energy efficiency is achieved by selecting energy efficient paths between cells connected by the selected virtual edge.

This scheme does not require any additional control overhead compared to existing geographic routing protocols and all computations are performed based solely on local information. The best performance in all the experiments is achieved with the Local Shortest Path in our virtual spanner. This schema outperform iterative power progress with face routing in both Gabriel Graph and Morelia Graph. In particular, for low densities our experiments report that energy consumption can be reduced up to 40%.

References

[BCG$^+$04] P. Boone, E. Chavez, L. Gleitzky, E. Kranakis, J. Opartny, G. Salazar, and J. Urrutia. Morelia test: Improving the efficiency of the gabriel test and face routing in ad-hoc networks. *Lecture Notes in Computer Science*, 3104:23–24, January 2004.

[BMSU01] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad-hoc wireless networks. *ACM/Kluwer Wireless Networks*, 7(6):609–616, 2001.

[GS69] K. Gabriel and R. Sokal. A new statistical approach to geographic variation analysis. *Systematic Zoology*, 18:259–278, 1969.

[KNS04] J. Kuruvila, A Nayak, and I. Stojmenovic. Progress based localized power and cost aware routing algorithms for ad hoc and sensor wireless networks. *Lecture Notes in Computer Science*, 3158:294–299, January 2004.

[RM98] V. Rodoplu and T. Meng. Minimum energy mobile wireless networks. In *ICC'98: Proceedings of the 1998 IEEE International Conference on Communications*, volume 3, pages 1633–1639. ACM Press, June 1998.

[SL01] I. Stojmenovic and X. Lin. Power aware localized routing in ad hoc networks. *IEEE Trans. Paralled Distributed Systems*, 12(10):1023–1032, 2001.

[Tou80] G. Toussaint. The relative neighbourhood graph of a finite planar set. *Pattern Recognition*, 12(4):261–268, 1980.