Improved Topology Control Algorithms for Simple Mobile Networks*

Fei (Sophie) Che[1], Errol L. Lloyd[1], and Liang Zhao[2]

[1] University of Delaware, Newark DE 19716, USA
{fei,elloyd}@cis.udel.edu
[2] QUALCOMM Inc.
zhaol@qualcomm.com

Abstract. Topology control is the problem of assigning powers to the nodes of an ad hoc network so as to create a specified network topology while minimizing the energy consumed by the network nodes. Topology control in *mobile* wireless ad hoc networks under the *Simple Mobile Network* (SMN) Model was introduced in [6]. In that model, there is only one moving node. That node moves on a straight line segment throughout a unit time interval and no assumptions are made about the moving speed. In this paper we study the problem of minimizing the maximum power consumed by any network node to maintain a 1-Connected (i.e. connected) topology under the SMN model. For that problem, in [6], a decision algorithm that runs in time $O(n^2)$ and an optimization algorithm that runs in time $O(n^2 \log n)$ are described. In this paper we improve upon those results by taking advantage of Voronoi diagrams and Delaunay triangulations. We provide three main results: a decision algorithm that runs in time $O(n \log n)$; an optimization algorithm with an expected running time of $O(n^{7/4} \log n)$; and a constant factor approximation algorithm that runs in time $O(n \log n)$. Simulation results evaluating the approximation algorithm are also provided.

Keywords: topology control, mobile ad hoc networks, Delaunay triangulations, Voronoi diagrams, 1-Connected, power optimization.

1 Introduction

Wireless Ad hoc Networks are networks in which wireless nodes cooperate to establish network connectivity and perform routing functions in the absence of infrastructure using self-organization. Similarly, *Mobile wireless Ad hoc Networks (MANETs)* are networks in which nodes communicate through wireless links but move freely. Here, each node functions, when necessary, as a relay node so as to allow multihop *store-and-forward* communications. A key problem associated

* Prepared through collaborative participation in the Communications and Networks Consortium sponsored by the U. S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-2-0011. The U. S. Government is authorized to reproduce and distribute reprints for Government purposes not withstanding any copyright notation thereon.

with mobile wireless ad hoc networks is to conserve energy so as to prolong the battery life and to accommodate the movement of the network nodes.

In a mobile wireless ad hoc network, the *network topology* is formed based on the nodes' transmission ranges and routes of node movement. The objective in *topology control* is to reduce energy consumption so as to maintain a specified network topology such as 1-Connected (i.e. the network is connected).

This paper considers topology control using the *Simple Mobile Network (SMN)* model. As described in [6], in the SMN model, there is only one moving node. That node is assumed to move on a straight line segment throughout a unit time interval. No assumptions are made about the moving speed of the node. Thus, that node could move at a uniform speed throughout the unit time interval, or it could be changing speeds at arbitrary times, or could even stop for a period. Regardless, the 1-Connected topology should be maintained without the algorithm having any knowledge of the actual movement other than the start and end points in the unit time interval. The goal is to minimize the maximum power consumed by any network node so that the network topology is 1-Connected under the SMN model. A formal definition of the problem is as follows.

Definition 1. *Under the* Simple Mobile Network *(SMN) model, the given network $\mathcal{N}$, denoted as $\langle \mathcal{N}_s, B\rangle$, has n stationary nodes in $\mathcal{N}_s$ and one moving node B. Associated with each stationary node is its Euclidean position. Associated with the moving node are its starting and ending Euclidean positions in a unit time interval. Nothing is assumed about its speed. The goal of the problem*[1] ⟨SMN*,* 1-Connected*,* MaxP⟩ *is to minimize the maximum power (denoted as* MaxP*) uniformly assigned to every node such that the network $\mathcal{N}$ achieves the property* 1-Connected *throughout the unit time interval.*

Relative to the above definition, when a line segment specifies the movement of a node during a unit time interval, that segment is said to be the moving route of the node.

Note that, as in the case of stationary networks [5], a solution to the above problem does *not* imply that each node transmits at the same power. Rather, after solving the above problem of minimizing the maximum power, each node determines its connectivities resulting from using that power and then sets its power level to the minimum required to achieve those connectivities.

One scenario (Figure 1) for SMN is a *vehicular ad-hoc network* (VANET) that provides communications among nearby vehicles, and between vehicles and nearby fixed roadside equipment that utilizes wireless transmitters, so as to increase driving comfort and safety [4]. In such an ad hoc network, consider a collection of roadside equipment, such as traffic lights, road sign alarms, roadside base stations, etc., and a vehicle equipped with a VANET device. Since the vehicle always moves on its specified route (a straight line in most cases) with varying speeds, the SMN model can be applied for maintaining the network connectivity

[1] Using a notation similar to the one used in [2], we specify a topology control problem by a triple of the form $\langle \mathbb{M}, \mathbb{P}, \mathbb{O}\rangle$. In this notation, $\mathbb{M}$ represents the mobility model, $\mathbb{P}$ represents the desired graph property and $\mathbb{O}$ represents the minimization objective.

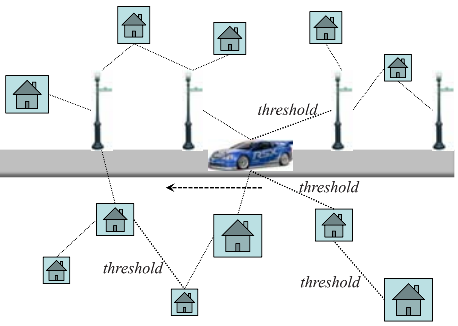

Fig. 1. An example scenario for SMN

throughout the vehicle movement. The goal is to reduce the transmission power of the VANET device while maintaining a connected network.

The primary contributions of this paper are providing polynomial time algorithms for ⟨SMN, 1-Connected, MaxP⟩. Our main results are:

– A decision algorithm for solving ⟨SMN, 1-Connected, MaxP⟩ with running time $O(n \log n)$. This improves upon the $O(n^2)$ algorithm provided in [6].
– An optimization algorithm for solving ⟨SMN, 1-Connected, MaxP⟩ with an *expected* running time $O(n^{7/4} \log n)$ and worst case time $O(n^2 \log n)$. This improves upon the algorithm given in [6] which runs in time $O(n^2 \log n)$ in both the worst and expected cases.
– A constant factor approximation algorithm for solving ⟨SMN, 1-Connected, MaxP⟩ with running time $O(n \log n)$. This running time matches the best running time for a stationary network.

The rest of the paper is organized as follows. In Section 2 we briefly review the key related work. In Section 3 we introduce certain graph definitions along with several definitions from computational geometry. We follow with our polynomial algorithms for SMN: the decision version (in Section 4), the optimization version (in section 5), and an approximation algorithm (in section 6). In Section 7 we present simulation results examining the quality of the solution provided by our approximation algorithm. Finally, Section 8 describes future directions.

2 Related Work

The general form of topology control in *stationary* wireless ad hoc networks was first proposed by Ramanathan and Rosales-Hain [5]. Among the several results in that paper, they presented an efficient algorithm for the problems of ⟨Undir, 1-Connected, MaxP⟩ (where Undir refers to the undirected graph model for stationary networks) that minimized the power uniformly assigned to any node in a stationary network such that the resulting network is 1-connected. As described in [5], the running time for that algorithm is $O(n^2 \log n)$ since there are at most $O(\log n)$ powers that need to be checked and each checking (for 1-Connected) takes time $O(n^2)$.

In [6], Zhao et.al. provided the first theoretical results for topology control in a mobile network model. Specifically, they provided a framework for solving $\langle$SMN, $\mathcal{P}$, MaxP$\rangle$, where $\mathcal{P}$ can be any monotone property [2]. When $\mathcal{P}$ is 1-Connected, their framework produces an algorithm that runs in time $O(n^3 \log n)$. Further, for the property k-Connected, they gave specialized algorithms improving upon their general results. Specifically, for 1-Connected, their decision algorithm runs in time $O(n^2)$ and their optimization algorithm runs in time $O(n^2 \log n)$. We improve upon both of those results.

3 Preliminaries

In this section we first provide definitions and notations concerning transmission powers and graphs related to SMN. In addition, we introduce certain terms and properties related to Voronoi diagrams and Delaunay triangulations.

3.1 Graph Definitions

Related to the definition of SMN, we present several definitions concerning transmission powers and related graphs relative to a stationary network $\mathcal{N}_s$ and a SMN $\mathcal{N}$. We use the term *MANET* when definitions apply to both stationary networks and SMNs.

Relative to lines and line segments, we define uv to be the line that goes through points u and v, $\overline{uv}$ to be the line segment with endpoints u and v, and $d(u, v)$ to be the Euclidean distance between u and v.

Definition 2. *For each ordered pair (u, v) of nodes in a MANET, at any given time instant, a* threshold *is a transmission power, denoted by $\pi(u, v)$, such that a signal transmitted by node u can be received by node v if and only if the transmission power of u is at least $\pi(u, v)$.*

In this paper we utilize the *geometric model* in which the threshold is determined by the Euclidean distance $d(u, v)$ between u and v. Typically, the threshold $\pi(u, v)$ is taken to be $d(u, v)^\alpha$, where α is the *attenuation constant* associated with path loss. The path loss is the ratio of the received power to the transmitted power of the signal. The value of α is generally less than 4. For simplicity of explanation we take α to be 1 throughout this paper, although, with the exception of the approximation ratio of the algorithm given in section 6, our results apply without modification for any value of α. Note that in the geometric model, threshold values are *symmetric*. That is, $\pi(u, v) = \pi(v, u)$. Thus, in the remainder of this paper, we let $\pi(u, v)$ denote both itself and $\pi(v, u)$.

Definition 3. *Let $G^p(\mathcal{N}_s)$ denote the undirected edge-weighted graph induced from a stationary network $\mathcal{N}_s$, when transmission power p is uniformly assigned to each node. That is, in $G^p(\mathcal{N}_s)$, an edge is present between nodes u and v if and only if $\pi(u, v) \leq p$, and the weight of edge $\overline{uv}$ is its threshold $\pi(u, v)$.*

Definition 4. *Let $G^p(\mathcal{N})$ denote the undirected edge-weighted graph induced from a SMN $\mathcal{N}$ by freezing the moving node B at some time instant and uniformly assigning the transmission power p to each node (including the moving node). That is, in $G^p(\mathcal{N})$: the moving node is located at some point on the moving route; an edge is present between nodes u and v if and only if $\pi(u,v) \leq p$; and, the weight of edge $\overline{uv}$ is its threshold $\pi(u,v)$, where u and v can be either two stationary nodes or a stationary node and the moving node.*

3.2 Voronoi Diagram and Delaunay Triangulation

The algorithms we give in this paper rely heavily on Voronoi diagrams and Delaunay triangulations. In this subsection we provide a brief overview of these two topics and related concepts. More detailed descriptions may be found in [8,9].

Let $Q = \{X_1, \ldots, X_n\}$ be a set of n distinct points in the Euclidean plane. VORONOI DIAGRAM $\mathcal{VD}$ of Q is the subdivision of the plane into n cells, one for each point X_i, with the property that a point u lies in the cell corresponding to a point X_i if and only if $d(u, X_i) < d(u, X_j)$ for each X_j with $j \neq i$ [8]. The cell that corresponds to a point X_i is denoted as the V-CELL DEFINED by X_i.

An important property of Voronoi diagrams is given in [8]:

Theorem 1. *The size of a Voronoi diagram on n points is linear. That is, the number of vertices in the Voronoi diagram is $O(n)$ and the number of edges in the Voronoi diagram is also $O(n)$.*

We next consider the dual graph of a Voronoi diagram $\mathcal{VD}$, as shown in Figure 2. This graph $\mathcal{G}$ has a node for every V-cell in $\mathcal{VD}$, specifically, the point defining the V-cell in the plane, and it has an arc between two points u and v if the corresponding V-cells share an edge. We call this graph the DELAUNAY GRAPH $\mathcal{DG}$ of Q. We further define a DELAUNAY TRIANGULATION $\mathcal{DT}$ to be a Delaunay graph of Q if no four points are co-circular. Then all bounded faces of $\mathcal{DT}$ are triangles. Otherwise, we define a DELAUNAY TRIANGULATION $\mathcal{DT}$ to be any triangulation obtained by adding edges to $\mathcal{DG}$ [8]. As a dual graph of a Voronoi diagram, the size of a Delaunay triangulation on n points is also linear [8]. We are interested in $\mathcal{VD}$ and $\mathcal{DT}$ not only because of their linear sizes but because they can be computed in time $O(n \log n)$ [8] (*cf.* $O(n^2)$ time to construct a complete graph for n points).

Given a Delaunay triangulation $\mathcal{DT}$, we denote an edge of $\mathcal{DT}$ as a D-EDGE. For any two endpoints u and v of a D-edge, we refer to u as v's D-NEIGHBOR and vice versa. Note that the V-cells defined by u and v must share a Voronoi edge. We denote a triangle of $\mathcal{DT}$ as a D-TRIANGLE. It is known from [8] that $\mathcal{DT}$ is a Delaunay triangulation of Q if and only if the circumcircle of any D-triangle is an EMPTY CIRCLE (i.e. a circle containing no point of Q in its interior). We refer to a circle $\mathcal{C}$ that is a circumcircle of a D-triangle as a D-CIRCLE and say that $\mathcal{C}$ is DEFINED by the three points which are the vertices of the triangle.

Definition 5. *A* D-THRESHOLD $\pi_D(u, v)$ *is the threshold of the D-edge between a pair of D-neighbors u and v.*

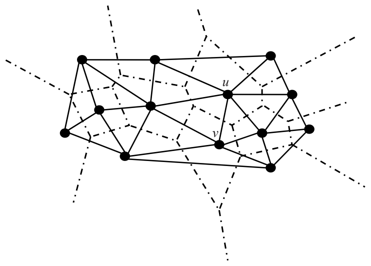

Fig. 2. Delaunay triangulation (in solid lines) on top of the Voronoi diagram

Definition 6. *A* D-BISECTOR *w of node u is a bisector point between any two nodes x and y both of which are D-neighbors of node u. We refer to x (or y) as w's* PARENT, *and we say that w is* INDUCED *by x and y.*

Definition 7. *A* B-THRESHOLD *$\pi_B(x, y)$ of node u is the threshold of an edge between x (or y) and the D-bisector w induced by x and y, where both x and y are D-neighbors of node u.*

Note that we denote a B-threshold by using the two parent nodes x and y instead of their D-bisector w, since when computing a B-threshold in our algorithm, it is natural to start with the two parent nodes.

Definition 8. *Consider a Delaunay triangulation $\mathcal{D}_n$ and a straight line l passing through $\mathcal{D}_n$. Let $\mathcal{C}$ be a D-circle in $\mathcal{D}_n$. We refer to any intersection point u between $\mathcal{C}$ and l as a* D-BREAKPOINT *of $\mathcal{C}$. For the three nodes which define the D-circle $\mathcal{C}$, we refer to the two nodes which are the endpoints of the nearest edge to u (or v) as* NEARER NEIGHBORS *of u (or v); and we refer to the other node as the* FARTHEST NEIGHBOR *of u (or v).*

This definition relates to the SMN model in the following way: Suppose we construct a Delaunay triangulation $\mathcal{DT}$ for the stationary nodes and B is located at the start point of the moving route. Then, as B is moving along the moving route, we incrementally maintain the Delaunay triangulation. A structural change to $\mathcal{DT}$, such as the introduction or removal of a D-neighbor of B, can only occur when B becomes co-circular with 3 other Voronoi vertices [3].

Observation 1. *Consider any two distinct points x and y on the line segment $[d_i, d_{i+1})$ where d_i and d_{i+1} are adjacent D-breakpoints on the moving route. Let Nb_x be the set of D-neighbors of B when B is at x and let Nb_y be the set at y. Then $Nb_x = Nb_y$. That is, when B is moving between two adjacent D-breakpoints, the set of D-neighbors of B will not change.*

Observation 2. *If B is entering a D-circle $\mathcal{C}$ at D-breakpoint u of $\mathcal{C}$, then the farthest neighbor of u becomes the new D-neighbor of B; similarly, if B is exiting the D-circle $\mathcal{C}$ at D-breakpoint v, then the farthest neighbor of v will cease to be a D-neighbor of B.*

Definition 9. *For two points u and v in the plane we define the* BISECTOR *of u and v as the perpendicular bisector of the line segment $\overline{uv}$. This bisector splits the plane into two* HALF-PLANES. *We denote the open half-plane that contains u by $h(u, v)$ and the open half-plane that contains v by $h(v, u)$. Note that $r \in h(u, v)$ if and only if $d(r, u) < d(r, v)$.*

Note that a Voronoi edge between two V-cells defined by u and v is a part of the bisector of u and v.

4 Decision Version of SMN

In this subsection, we provide an algorithm for the decision version of ⟨SMN, 1-CONNECTED, MAXP⟩. Recall that given a power p the goal here is to determine whether the network $\mathcal{N}$ achieves property 1-CONNECTED under p throughout the unit time interval. Our algorithm improves upon the results in [6], where an $O(n^2)$ algorithm was given for this decision version. Here, we present an $O(n \log n)$ algorithm utilizing Delaunay triangulations.

The algorithm proceeds by first finding the connected components of the stationary network and then checking the connectivity along the entire moving route between each connected component and the moving node. That is, at each point along the moving route, to check whether or not there exists a connection between B and each connected component of the stationary network. To do the connectivity checking we use an incremental method based on *constant-connectivity segments*. A CONSTANT-CONNECTIVITY SEGMENT [6] is a line segment formed by the adjacent intersection points of the moving route and the circles of radius p with their centers at some stationary node(s), where p is the given power value. The most important characteristic of such a segment is that there are no connectivity changes when the moving node moves within the segment. Such changes occur only at segment endpoints.

Algorithm 1 provides the details of the method. The time-dominating step is to compute the connected components of the $O(n)$ stationary nodes. To do this we first construct the Delaunay triangulation $\mathcal{DT}_s$ for the stationary network $\mathcal{N}_s$ and then compute the connected components of the graph $G^p(\mathcal{N}_s)$ induced with only D-edges. By doing this, the total number of edges in $G^p(\mathcal{N}_s)$ is only $O(n)$, from which it follows that the running time for computing the connected components of $G^p(\mathcal{N}_s)$ is only $O(n)$, achieving an overall running time of $O(n \log n)$ since the construction of $\mathcal{DT}_s$ then dominates. Thus,

Theorem 2. *Algorithm 1 runs in time $O(n \log n)$.*

To establish the correctness of Algorithm 1 we begin with:

Lemma 1. *For any two nodes u and v in a stationary network $\mathcal{N}_s$, if u and v are connected under power p, then there must exist a path between them such that the path consists of only D-edges and the threshold of each such D-edge is no more than p.*

Algorithm 1. for ⟨SMN, 1-Connected, MaxP⟩ – CheckSMNFor1Conn

Input: An instance of SMN $\mathcal{N} = \langle \mathcal{N}_s, B \rangle$ and a power p.
Output: If $\mathcal{N}$ achieves 1-Connected under p, return true; else return false.

```
 1: Construct the Delaunay triangulation $\mathcal{DT}_s$ of the nodes in $\mathcal{N}_s$;
 2: Induce $G^p(\mathcal{N}_s)$ with only D-edges;
 3: Compute the connected components $C_1, C_2, \ldots, C_k$ of $G^p(\mathcal{N}_s)$, and label each node
    with the connected component to which it belongs;
 4: Partition the moving route into constant-connectivity segments under $p$;
 5: for the first constant-connectivity segment do
 6:     Record the number of connectivities between $B$ and each $C_k$;
 7:     if $B$ is not connected to every $C_k$ then return false;
 8: for each successive segment do
 9:     Update the number of connectivities between $B$ and each $C_k$;
10:     if $B$ is not connected to every $C_k$ then return false;
11: return true;
```

Proof. Given power value p, we induce an undirected graph $G^p(\mathcal{N}_s)$ from $\mathcal{N}_s$ under power p. Recall that in $G^p(\mathcal{N}_s)$ an edge is present between u and v if and only if $\pi(u, v) \leq p$. There could be multiple paths between u and v. We need to show that there exists at least one path that consists of only D-edges.

By way of contradiction, suppose each of the paths between u and v has at least one edge that is not a D-edge. Arbitrarily consider one such path and let $\overline{xy}$ be an edge on that path that is not a D-edge.

Since $\overline{xy}$ is not a D-edge, the V-cells defined by x and y must not be adjacent to each other. That is, when moving from x to y, we must encounter at least one V-cell that is defined by some node other than x or y. Let w be such a node whose corresponding V-cell is the first encountered V-cell when moving from x to y (Figure 3). Thus, w must be a D-neighbor of x (while w may or may not be a D-neighbor of y).

Thus, the midpoint of x and y, say M, must not lie in either of the V-cells defined by x or y (including the boundaries). That is, M lies in the half-plane $h(w, x)$. Hence, $d(x, M) > d(x, w)/2$, i.e. $d(x, w) < 2 \times d(x, M) = d(x, y)$. Note also that $d(w, M) < d(x, M) = d(y, M)$, which implies that M lies in the

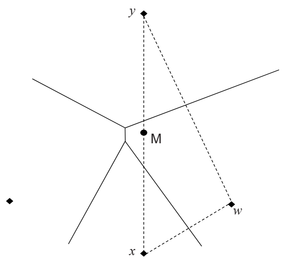

Fig. 3. non D-edge $\overline{AB}$

half-plane $h(w, y)$. Using the same argument as above, it follows that $d(y, w) < d(x, y)$. Hence, $\pi(x, w) < \pi(x, y)$ and $\pi(y, w) < \pi(x, y)$, which implies that $\overline{xw}$ and $\overline{yw}$ must be edges in $G^p(\mathcal{N}_s)$. Therefore, we can remove $\overline{xy}$ from the path and insert $\overline{xw}$ and $\overline{yw}$ in its place on the path. Note that while $\overline{yw}$ is shorter than $\overline{xy}$, it may not be a D-edge.

If we apply this replacement recursively, each such action removes a non D-edge from the path (perhaps as noted above we replace it with a shorter non D-edge). Since there are only a finite number of edges, the process terminates at a path between u and v that consists of only D-edges – a contradiction. □

With the above lemma in hand, the correctness of Algorithm 1 follows:

Theorem 3. *If Algorithm 1 returns true, the value p is such that network $\mathcal{N}$ achieves property* 1-CONNECTED *under power p; if Algorithm 1 returns false, $\mathcal{N}$ can not achieve* 1-CONNECTED *under power p.*

5 Optimization Version of SMN

In this section, we give an algorithm for the optimization version of ⟨SMN, 1-CONNECTED, MAXP⟩. Recall that the goal is to minimize the power uniformly assigned to each node (both stationary nodes and the moving node) such that the network achieves property 1-CONNECTED throughout the unit time interval. In [6], two algorithms are provided for this problem: one runs in time $O(n^3 \log n)$ and the other runs in time $O(n^2 \log n)$. We note that the better of those two algorithms also runs in *expected* time $O(n^2 \log n)$ since the algorithm includes a step that sorts a set of $O(n^2)$ values.

The algorithm we present in this section takes advantage of Delaunay triangulations to yield an expected running time of $O(n^{7/4} \log n)$, with an $O(n^2 \log n)$ worst case time.

Our algorithm is best understood in the context of the $O(n^3 \log n)$ approach in [6] mentioned above. That approach is to collect certain candidate thresholds into a set P, sort those thresholds, and do binary search within P to locate the minimum overall power value for $\mathcal{N}$. In [6], the thresholds collected are *stationary thresholds* (a threshold between two stationary nodes) and *bisector thresholds* (a threshold between a stationary node and a bisector point) and it is shown there that the optimal power must be one of those thresholds.

In our approach, we introduce two new types of thresholds: *D-thresholds* and *B-thresholds*, which are particular types of stationary thresholds and bisector thresholds respectively. We utilize $O(n)$ D-thresholds (compared with $O(n^2)$ stationary thresholds in [6]) and $O(n^2)$ B-thresholds (compared with $O(n^3)$ bisector thresholds in [6]). After collecting all of the D-thresholds and B-thresholds, we sort them, and do binary search using our new algorithm CHECKSMN-FOR1CONN from Subsection 4 to check the connectivity of $\mathcal{N}$ in each iteration. Algorithm 2 provides the details of the method as outlined above.

Key to our approach is a fast method for computing the B-thresholds. In our algorithm, we begin by constructing a Delaunay triangulation $\mathcal{DT}_s$ for $\mathcal{N}_s$. Next, the starting point V_{start} of the moving route is temporarily inserted and $\mathcal{DT}$ is incrementally constructed. Then we compute D-bisectors of V_{start} for every two D-neighbors of V_{start}, and place the B-thresholds of those D-bisectors into our candidate set P.

To collect the other D-bisectors and their B-thresholds incrementally, we perform the following steps:

- Compute and sort all of the D-breakpoints[2];
- For each successive D-breakpoint, we maintain a set S of current D-neighbors of the moving node B - when B arrives at a D-breakpoint entering a D-circle, we include the farthest neighbor into set S, and when exiting a D-circle, we remove the farthest neighbor from set S.
- For each successive D-breakpoint, to update P, we only need to compute the bisector points between the new D-neighbor and all of the nodes in set S and include those B-thresholds into our set P.

5.1 Running Times

Theorem 4. *Algorithm 2 runs in time $O(n^2 \log n)$ in the worst case.*

The proof of this theorem is given in [1].

In the remainder of this subsection we establish that the *expected* running time of Algorithm 2 is $O(n^{7/4} \log n)$. We begin by taking a closer look at Algorithm 2. The worst case running time is dominated by steps II (lines 5 – 19) and III (line 20). Specifically, for step II, the running time analysis is based on the observation that there are *at most* $O(n)$ D-neighbors of B at the starting point and at each D-breakpoint. But this is a worst case analysis. In the following Lemma we prove that the *expected* number of D-neighbors of B at any point on the moving route is much less, specifically, the expected number is only $O(n^{3/4})$.

Lemma 2. *Consider n nodes, $X_1, \ldots, X_n$, which are i.i.d. (independent and identically distributed) and uniformly distributed in a rectangular area $\mathcal{R}$. Let l be a straight line going through the area $\mathcal{R}$, let B be a node placed at any arbitrary point on l, and let $\mathcal{DT}$ be a Delaunay triangulation on $X_1, \ldots, X_n$ and B. The expected number of D-neighbors of B is $O(n^{3/4})$.*

Proof. Without loss of generality, we orient the rectangular area $\mathcal{R}$ to the $x-$ and $y-$ axes, and let a and b be the width and length respectively. We begin by assuming that l runs horizontally through the exact middle[3] of $\mathcal{R}$. Let m and n be, respectively, horizontal lines lying a distance of d above and below l, where $d = a/\sqrt[4]{n}$. We refer to the position of $\mathcal{R}$ between m and n as the ZONE.

Let x be an arbitrary point location on l in $\mathcal{R}$. We separately compute the expected number of D-neighbors of x inside and outside the zone and then sum together those expected values.

[2] When adding a D-breakpoint u to a list L, we say that $L \leftarrow L + u$.

[3] With some additional discussion, the proof holds for completely general positions of l. Due to space limitation, the discussion is omitted.

Algorithm 2. for ⟨SMN, 1-Connected, MaxP⟩ – FastSMNFor1Conn

Input: An instance of SMN $\mathcal{N} = \langle \mathcal{N}_s, B\rangle$.
Output: The minimum power p_{min} such that $\mathcal{N}$ achieves 1-Connected under p_{min}.

```
 1: Construct the Delaunay triangulation $\mathcal{DT}_s$ of the nodes in $\mathcal{N}_s$;
 2: $P \leftarrow \emptyset$;
            /* Step I: Collect D-thresholds */
 3: for each pair of D-neighbors $V_i$ and $V_j$ in $\mathcal{DT}_s$ do
 4:     $P \leftarrow P \cup \{\pi_D(V_i, V_j)\}$;
            /* Step II: Collect B-thresholds */
 5: $L \leftarrow \{V_{start}\}$;     /* where $L$ is a list and $V_{start}$ is the start point of the moving
    route */
 6: for each D-triangle $dt_i$ in $\mathcal{DT}_s$ do
 7:     Compute D-circle $\mathcal{C}_i$ of $dt_i$;
 8:     for each D-breakpoint $u$ of $\mathcal{C}_i$ do
 9:         $L \leftarrow L + u$;     /* Add $u$ and $v$ to $L$ even if there already exist points in $L$
    with the same location as $u$ or $v$. */
10: Sort $L$ by the locations of points, from left to right;     /* After sorting, the
    relative order of points sharing the same location is not significant. */
11: Incrementally construct a Delaunay Triangulation $\mathcal{DT}'$ for $\mathcal{DT}_s \cup V_{start}$, and ini-
    tialize a set $S$ with the current D-neighbors of $V_{start}$;
12: for any two nodes $u$ and $v$ in set $S$ do
13:     $P \leftarrow P \cup \{\pi_B(u, v)\}$;
14: for each successive D-breakpoint $w$ in $L$ do
15:     if entering a D-circle then
16:         for each node $v$ in $S$ do
17:             $P \leftarrow P \cup \{\pi_B(u, v)\}$, where $u$ is the farthest neighbor of $w$;
18:         $S \leftarrow S \cup u$;
19:     else Remove $u$ from $S$;
            /* Step III: Sort threshold values and do binary search */
20: Sort $P$ by threshold values, and use binary search to locate the least $p_{min} \in P$
    such that CheckSMNFor1Conn($\mathcal{N}, p_{min}$) is true;
21: return$p_{min}$;
```

1. The expected number of D-neighbors of x outside the zone.
   We refer to a node which is not in the zone as an O-Node. Associate an indicator random variable z_i with each node $X_i, 1 \leq i \leq n$. The value of z_i is 1 if X_i is an O-node and a D-neighbor of x and 0 otherwise. Let the random variable Z give the total number of O-nodes that are D-neighbors of x. By linearity of expectation (see [7]) and since z_i is an indicator random variable,

$$\mathbb{E}[Z] = \sum_{i=1}^{n} \mathbb{E}[z_i] = \mathbb{P}[z_i = 1]$$

   Note that in a Delaunay triangulation, two nodes X_i and x are D-neighbors if and only if there is a closed circle $\mathcal{C}_i$ with X_i and x on its boundary and no other nodes are contained in $\mathcal{C}_i$ (see [8]). Note also that the smallest empty circle with X_i and x on its boundary is the circle with $\overline{xX_i}$ being its diameter. It follows that the circle $\mathcal{C}_i$ has diameter at least $d(x, X_i)$. Thus,

$$\mathbb{P}[z_i = 1] \leq (1 - \frac{\pi(d(x, X_i)/2)^2}{ab})^n = (1 - \frac{\pi a}{4b\sqrt{n}})^n$$

Thus,

$$\mathbb{E}[z_i] = \mathbb{P}[z_i = 1] \leq (1 - \frac{\pi a}{4b\sqrt{n}})^n$$
$$< e^{-\frac{\pi a}{4b}\sqrt{n}} < \frac{32b^2}{\pi^2 a^2} \cdot \frac{1}{n}$$

using the inequalities (from [7]) $\frac{x}{1+x} \leq ln(1 + x) \leq x$ for $x > -1$, where equality holds only for $x = 0$. Hence,

$$\mathbb{E}[Z] = \sum_{i=1}^{n} \mathbb{E}[z_i] < \frac{32b^2}{\pi^2 a^2} = O(1)$$

2. The expected number of D-neighbors of x inside the zone.
   Similarly to the above, the expected number of D-neighbors of x inside the zone is no more than $2n^{3/4}$.

Finally, combining the results of parts (1) and (2), the expected total number of D-neighbors of x at any point location on l is $O(n^{3/4})$. □

With Lemma 2 in hand, an analysis of the steps in Algorithm 2 yields:

Theorem 5. *The expected running time of Algorithm 2 is $O(n^{7/4} \log n)$.*

The proof of this theorem appears in [1]. Likewise in [1] we prove the correctness of the algorithm:

Theorem 6. *The value p_{min} returned by Algorithm 2 is the minimum power such that network $\mathcal{N}$ achieves* 1-Connected *under power p_{min}.*

6 An Approximation Algorithm for SMN

In this section, we present a constant factor approximation algorithm with an $O(n \log n)$ running time for SMN. The constant factor depends on the attenuation constant α. In the explanation given here, we assume that $\alpha = 1$, in which case the algorithm yields a 2-approximation.

Our approach to achieve a 1-connected network under SMN is to first create a 1-connected network of the stationary nodes without considering the moving node and then determine the minimum power such that the moving node is continuously connected to that stationary network. Note that in this approach the moving node never explicitly provides connectivity between two stationary nodes. This is in sharp contrast to the algorithms that find an optimal solution.

The key ingredient in our algorithm is the use of a Voronoi diagram of just the stationary nodes. Recall that in a Voronoi diagram $\mathcal{VD}$ any point u in a V-cell defined by X_i is closer to X_i than to any other X_j with $j \neq i$. What does that mean for SMN? In SMN, with the Voronoi diagram $\mathcal{VD}_s$ for $\mathcal{N}_s$ in hand, when the moving node B is inside the V-cell defined by stationary node X_i, it follows that X_i is the closest stationary node to B. Thus, to maintain the connectivity of the entire network, the moving node B must remain connected with X_i when B is inside the V-cell defined by X_i. Hence, we segment the moving route into line segments by the intersection points of that moving route with V-cells, and compute the *V-thresholds* defined below.

Definition 10. *A* V-THRESHOLD $\pi_V(A, B)$ *is the threshold that can be assigned to both A and B such that nodes A and B are continuously connected as B moves inside the V-cell defined by A. If the moving route of B does not intersect with the V-cell defined by A, we let $\pi_V(A, B) = 0$.*

The following lemma can be easily verified.

Lemma 3. *Given a V-cell defined by a stationary node A and a moving node B whose moving route intersects with the V-cell at B_0 and B_1,*

$$\pi_V(A, B) = MAX(\pi(A, B_0), \pi(A, B_1))$$

To achieve 1-CONNECTED for the entire network, the optimal power for must be at least equal to the maximum of those V-thresholds so that the moving node is continuously connected to the stationary network. An algorithm based on the above idea is given in Algorithm 3.

Algorithm 3. for ⟨SMN, 1-CONNECTED, MAXP⟩ – APPROXSMN

Input: An instance of SMN $\mathcal{N} = \langle \mathcal{N}_s, B \rangle$.

Output: The power p such that $\mathcal{N}$ achieves 1-CONNECTED under p and $p \leq 2 \times p_{min}$, where p_{min} is the minimum power for 1-CONNECTED.

```
1: Construct a Voronoi diagram $\mathcal{VD}_s$ for $\mathcal{N}_s$;
2: Find a MST $T$ for $\mathcal{N}_s$ based on $\mathcal{VD}_s$;
3: $p_M \leftarrow$ the largest edge threshold in $T$;
4: $P \leftarrow \emptyset$;
5: for each V-cell $c_i$ defined by a node $V_i$ do
6:     $P \leftarrow P \cup \{\pi_V(V_i, B)\}$;
7: $p_V \leftarrow$ the largest V-threshold in $P$;
8: return$p \leftarrow MAX(p_M, p_V)$;
```

In regard to the running time of Algorithm 3, note that it takes time $O(n \log n)$ to both construct the Voronoi diagram in line 1 and to find a MST in line 2. Those two steps dominate the running time. Thus,

Theorem 7. *Algorithm 3 runs in time $O(n \log n)$.*

In regard to the quality of the approximation:

Theorem 8. *When $\alpha = 1$, the power value p returned by Algorithm 3 is less than $2 \times p_{min}$, where p_{min} is the minimum power for* ⟨SMN, 1-Connected, MaxP⟩.

The proof may be found in the appendix.

Corollary 1. *For general values of α, the power value p returned by Algorithm 3 is less than $2^{\alpha} \times p_{min}$, where p_{min} is the minimum power for* ⟨SMN, 1-Connected, MaxP⟩.

The bounds of 2 and 2^{α} we proved above are tight. An example is shown in Figure 4 for the bound of 2 (i.e. $\alpha = 1$. In this case, the result returned by Algorithm 3 is $2p$ which connects all of the stationary nodes without using the moving node as an intermediate node. The optimal solution is $p + \epsilon$. Thus, the bound of 2 is tight.

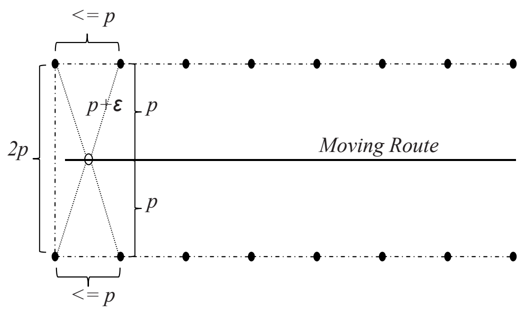

Fig. 4. An example shows that the bound of 2 is tight

7 Simulation Results

In this section we describe simulations used to evaluate ApproxSMN and compare the power value it returns to the optimal power value. This was done by implementing ApproxSMN and an algorithm producing the optimal value using C++ with the CGAL [10] and Boost [11] libraries. We simulated a network with stationary nodes randomly distributed in a square field. The moving route was a horizontal line running through the middle of the field. The x-coordinate of the start (end) point of the moving route was the left-most (right-most) x-coordinate of the stationary nodes. The threshold of each edge was its Euclidean length. The simulation was run 200 times each for 5, 10, 20, 50, and 100 nodes. The following table shows for each number of nodes the number of instances where ApproxSMN returned a non-optimal value.

number of nodes	5	10	20	50	100
number of non-opt values	41	6	2	0	0

Surprisingly, in most cases the power value returned by ApproxSMN is the optimal value. Why? Note that the moving node may work as a "connector" between stationary nodes in the optimal solution but will never do so in the approximate solution. Further, when the moving node is working as a connector, it must do so along the entire moving route. That is, there must exist stationary nodes in each connected component that are close enough to be connected to the moving node. At the same time, those nodes can not be directly connected to analogous nodes in other connected components. Such cases are unlikely to occur when a large number of nodes are randomly distributed.

Beyond that, also notice that by using the moving node as a connector, the node degrees (both maximal and average) in the optimal solution are less than that of the approximate solution (Figure 5). Note that generally speaking, a higher degree will lead to higher congestion and also consume more energy.

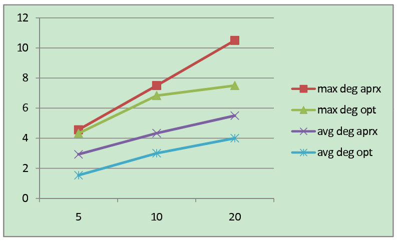

Fig. 5. Difference between node degrees in approximate and optimal results

8 Conclusions and Future Work

In this paper, we explored geometric approaches to the problem $\langle$SMN, 1-Connected, MaxP$\rangle$. In the SMN model, there is one moving node and n stationary nodes. The moving node is assumed to move on a straight line during a unit time interval without any limit of its moving speed. The goal is to find the minimum power that can be assigned to all nodes such that the network achieves the property 1-Connected. By taking advantage of the properties of Voronoi diagrams and Delaunay triangulations, we provided an optimization algorithm for solving $\langle$SMN, 1-Connected, MaxP$\rangle$ with an expected running time $O(n^{7/4} \log n)$ and an $O(n \log n)$ algorithm for the corresponding decision version. Beyond that, we also provided a constant factor approximation algorithm with running time $O(n \log n)$.

There are many open problems for further research. First, it would be interesting to develop algorithms under a more general mobile model, in which all nodes are mobile and/or the moving route can be any arbitrary curve instead of a straight line segment. Another direction is to develop distributed algorithms based on our algorithms for the SMN model and its generalizations.

Acknowledgement. We thank S.S. Ravi of SUNY Albany for his insights.

Disclaimer: The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government.

References

1. Full version, `http://www.cis.udel.edu/~fei/papers/ImprovedSMN.pdf`
2. Lloyd, E.L., Liu, R., Marathe, M.V., Ramanathan, R., Ravi, S.S.: Algorithmic Aspects of Topology Control Problems for Ad Hoc Networks. J. Mobile Networks and Applications (MONET) 10, 19–34 (2005)
3. Icking, C., Klein, R., Köllner, P., Ma, L.: Java Applets for the Dynamic Visualization of Voronoi Diagrams. In: Klein, R., Six, H.-W., Wegner, L. (eds.) Computer Science in Perspective. LNCS, vol. 2598, pp. 191–205. Springer, Heidelberg (2003)
4. Yousefi, S., Mousavi, M.S., Fathy, M.: Vehicular Ad Hoc Networks (VANETs): Challenges and Perspectives. In: 6th International Conference on ITS Telecommunications Proceedings, pp. 761–766. IEEE Press, Los Alamitos (2006)
5. Ramanathan, R., Rosales-Hain, R.: Topology Control of Multihop Wireless Networks Using Transmit Power Adjustment. In: IEEE INFOCOM 2000, pp. 404–413. IEEE Press, Los Alamitos (2000)
6. Zhao, L., Lloyd, E.L., Ravi, S.S.: Topology Control for Simple Mobile Networks. In: IEEE GLOBECOM 2006-WASNet. IEEE Press, Los Alamitos (2006)
7. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. MIT Press, Cambridge (2003)
8. de Berg, M., Van Kreveld, M., Overmars, M., Schwarzkopf, O.: Computational Geometry: Algorithms and Applications. Springer, Heidelberg (2000)
9. Preparata, F.P., Shamos, M.I.: Computational Geometry: An Introduction. Monographs in Computer Science. Springer, Heidelberg (1985)
10. CGAL, Computational Geometry Algorithms Library, `http://www.cgal.org`
11. Boost C++ Libraries, `http://www.boost.org`