Topology Control in Constant Rate Mobile Ad Hoc Networks [1]

Liang Zhao [2] Errol L. Lloyd [2] S. S. Ravi [3]

June 26, 2008

Abstract

Topology control is the problem of assigning power levels 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. While considerable theoretical attention has been given to the issue of topology control in wireless ad hoc networks, all of that prior work has concerned *stationary* networks. When the nodes are mobile, there is no algorithm that can guarantee a graph property (such as network connectivity) throughout the node movement.

In this paper we study topology control in *mobile* wireless ad hoc networks (MANETs). We define a mobility model, namely the *constant rate mobile network model* (CRMN), in which we assume that the speed and direction of each moving node are known. The goal of topology control under this model is to minimize the maximum power used by any network node in maintaining a specified *monotone* graph property. Network connectivity is one of the most fundamental monotone properties.

Under the CRMN model, we develop general frameworks for solving both the *decision* version (i.e. for a given value $p > 0$, will a specified monotone property hold for the network induced by assigning the power value p to every node?) and the *optimization* version (i.e. find the minimum value p such that the specified monotone property holds for the network induced by assigning the power value p to every node) of the topology control problems. Efficient algorithms for specific monotone properties can be derived from these frameworks. For example, when the monotone property is network connectivity, our algorithms for the decision and optimization versions have running times of $O(n^2 \log^2 n)$ and $O(n^4 \log^2 n)$ respectively. Our results represent a step towards the development of efficient and provably good distributed algorithms for topology control problems for MANETs.

1 Introduction

Two of the most critical issues associated with wireless ad hoc networks used in military and search-and-rescue operations are to conserve energy so as to prolong the battery life and to accommodate the movement of the network nodes. This paper considers these issues in the context of *topology control*.

A *wireless ad hoc network* consists of a collection of nodes which self-organize using communication based on radio propagation, since there is no pre-existing infrastructure. In communicating through *wireless links*, each node functions, when necessary, as a relay so as to allow multihop communications. In wireless ad hoc networks, battery power is a precious resource.

In a *mobile* wireless ad hoc network *(MANET)*, where nodes move freely, the *network topology* is formed based on the nodes' transmission ranges and routes of node movement. The objective in *topology control* is to maintain a specified network topology (i.e. graph property) such as connected, biconnected or diameter at most d. The desired effect of topology control is to reduce energy consumption, reduce MAC layer interference between adjacent nodes, and to increase the effective network capacity. Note that the network performance can be impacted in a major way by the network topology. A dense topology may induce high interference, which, in turn, reduces the effective network capacity due to limited spatial reuse and may cause unnecessarily high energy consumption. In contrast, a sparse topology is vulnerable to network partitioning due to node or link failures.

The primary method of accomplishing topology control is by adjusting the transmission powers of the network nodes. That is, each node is assigned a transmission power level so that the induced graph of the

[1] Preliminary results regarding connectivity only were presented at the GLOBECOM'06-WirelessComm [22] conference in San Francisco, California, November 2006.

[2] Department of Computer and Information Sciences, University of Delaware, Newark, DE 19716. Email: {zhaol, elloyd}@cis.udel.edu. 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.

[3] Department of Computer Science, University at Albany - SUNY, Albany, NY 12222. Email: ravi@cs.albany.edu.

network satisfies a prespecified topology. Further, the assignment of transmission powers to nodes aims to optimize some function of those powers. The most common objective, and the one utilized throughout this paper, is to minimize the *maximum power* utilized by any node.

While over the past several years considerable theoretical attention [13, 7, 8, 18, 20, 9, 4, 14, 17, 16, 1, 15, 11] has been given to the issue of topology control in wireless ad hoc networks, all of that work has concerned stationary networks. Such results are useful in the context of ad hoc networks where nodes are generally stationary (e.g. sensor networks). However, in other applications of ad hoc networks such as search-and-rescue operations and mobile robots patrolling an area, most of the nodes in the system are mobile. (Additional discussion regarding these applications is provided in Section 2.3.) In such contexts, it is important to study topology control problems for mobile ad hoc networks. To our knowledge, the impact of node mobility on topology control problems has not received attention in the literature. In this paper we are the first to provide polynomial time algorithms for optimally solving topology control problems in *mobile* wireless ad hoc networks. These topology control problems are studied under a generic mobile network model, *Constant Rate Mobile Networks*, in which we assume the speed and direction of each moving node is known. As will be explained in Section 2, this generic model underlies several mobility models considered in the literature.

Our primary contribution is a collection of polynomial time algorithms and frameworks[4] for topology control problems for maintaining specified graph properties in mobile ad hoc networks where certain information is known about node movement. The centralized algorithms that we provide are immediately applicable in networks where node movements are predictable and/or periodic, hence transmission powers can be computed in advance by a central authority. Two example scenarios of this variety are provided in a later section. Relative to *distributed* topology control algorithms for mobile ad hoc networks, our results are useful in two ways. First, these first centralized algorithms that we provide will serve as a critical baseline for the performance evaluation of future distributed heuristics. Second, developing *any* topology control algorithm when nodes are mobile is a difficult task, regardless of whether the algorithm is distributed or centralized. Part of this process is to uncover basic properties and characteristics of topology control problems when nodes are mobile. Our centralized results provide an important first step in this understanding that can be built upon in future work to develop distributed algorithms.

[4]In the context of this paper, a framework is a collection of concrete algorithms with each algorithm solving the topology control problem for a particular graph property.

2 Key Definitions & Summary of Main Results

In this section we first introduce the fundamentals of topology control, and provide the formal definitions of monotone properties. Then we establish a generic mobility model and specify topology control problems that incorporate mobility. To conclude this section, we provide a summary of the main results obtained in this paper.

2.1 Topology Control Fundamentals

In studying topology control for MANETs, we are given a network $\mathcal{N}$ consisting of a set V of nodes in the plane, each of which may move as time progresses[5]. At any given time instant, for each ordered pair (u, v) of nodes, there exists a transmission power *threshold*, denoted by $\pi(u, v)$, with the following significance: 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. Throughout this paper, 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 [19]. The value of α is typically between 2 and 4. Note that in the geometric model, threshold values are *symmetric*. That is, $\pi(u, v) = \pi(v, u)$. In the remainder of this paper, we let $\pi(u, v)$ denote both itself and $\pi(v, u)$.

At any instant in time, given the transmission powers and the positions of the nodes, an ad hoc network can be represented by an undirected graph over the nodes of the network. An edge (u, v) is in this *induced* graph if and only if the transmission powers of both u and v are at least the threshold $\pi(u, v)$.

In the context of MANETs, the main goal of *topology control* is to assign transmission powers to nodes so that the mobile network $\mathcal{N}$ achieves a specified property $\mathcal{P}$ (i.e. the graph induced by those transmission powers has the property $\mathcal{P}$) and the transmission

[5]We let the node name represent both itself and the Euclidean position of the node.

powers are optimal. Two common optimization objectives considered in the literature are to minimize the maximum power (denoted by MaxP) and to minimize the total power (denoted by TotalP) assigned to the nodes. Under MaxP, it is standard to assume that the same power is assigned to all nodes[6]. In this case, we say that *network $\mathcal{N}$ achieves a property $\mathcal{P}$ under such power*.

It is clear that the difficulty of finding the optimal power may vary with the property $\mathcal{P}$. Note that given a general property $\mathcal{P}$, the corresponding topology control problem may be **NP**-hard. For example, letting G be the graph induced from network $\mathcal{N}$ under power p, consider the property that G is a Tree. Simply determining whether there is *any* power assignment (without any optimization objective) such that the resulting undirected graph G is a tree is **NP**-complete [15]. Hence, there is no efficient way to find a power p such that network $\mathcal{N}$ achieves any general property $\mathcal{P}$ under power p, unless $\mathbf{P} = \mathbf{NP}$. To efficiently deal with graph properties in a general manner, the properties we consider in this paper are *monotone properties* [15]. Fortunately, the commonly used properties in networks are in this class.

2.2 Monotone Properties

Definition 2.1 *A property $\mathcal{P}$ of the undirected graph induced by a power assignment is* ***monotone*** *if the property continues to hold even when the powers assigned to some (perhaps all) nodes are increased while the powers assigned to the other nodes remain unchanged.*

This definition implies that for any monotone property $\mathcal{P}$, if G is a graph satisfying $\mathcal{P}$ and one or more additional edges are added to G, then the resulting graph also satisfies $\mathcal{P}$.

Since a stationary network is in one-to-one correspondence with the graph which is induced by the powers assigned to the nodes, Definition 2.1 applies directly to a stationary network. For a mobile network, at any instant in time, there exists a stationary network. Hence, a mobile network achieves a monotone property $\mathcal{P}$ if $\mathcal{P}$ is achieved by all stationary networks that exist during the lifetime of the mobile network.

For example, if $\mathcal{P}$ is Connected, then at every instant in time, the undirected graph induced by the transmission powers of the nodes is *connected*. Following this convention, we extend all monotone properties from stationary networks to mobile networks.

Other examples of monotone properties include k-Connected (including both k-Node-Connected and k-Edge-Connected, where $k \geq 1$), Bounded Diameter, and Minimum Degree Constraint. Here, k-Node/Edge-Connected means that the graph remains connected with the removal of any set of $k-1$ nodes/edges; Bounded Diameter means that the diameter of a graph cannot exceed a prespecified constant c ($c \geq 1$), where the diameter is the maximum shortest path distance between any pair of nodes in a graph; and Minimum Degree Constraint means that the node degree of every node in the graph is at least a prespecified constant c ($c \geq 1$). Note that Minimum Degree Constraint does not require the graph to be connected.

2.3 Mobility Models & Problem Specification

Ideally, one would solve the topology control problem just once, thereby assigning powers to the nodes so that the MANET achieves a monotone property (such as Connected) throughout the network lifetime. Unfortunately even for monotone properties, this requirement is not realistic in that nodes are moving, and with that movement the power requirements needed to achieve the property may change dramatically. A more realistic approach is to assign power levels so that the network achieves the property throughout some prespecified interval of time. At the start of each interval, the power levels of the nodes are recomputed based on the current location of the nodes and taking into account any additional information that is known about the movement of the nodes. Thus, the general problem that we consider in this paper is to minimize the maximum power assigned to any node in the MANET such that the network achieves a specific monotone property throughout a *unit time interval*. Since this interval would generally be quite short, we can assume that during each unit time interval the movement of each node can be represented by a line segment. That is, the movement of the node is on a straight line without any bends or curves. If a node does not move at all during a unit time interval then it is *stationary*, otherwise the node is *moving*. A *stationary network* is one in which all of the nodes are stationary.

Using a notation similar to the one used in [15], 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 prop-

[6] In particular, this assumption can be made without loss of generality for monotone properties, as will become clear from the next section.

erty and $\mathbb{O}$ represents the minimization objective. In this paper, $\mathbb{M} \in \{\text{CRMN}\}$, which will be explained shortly. Generally, $\mathbb{O} \in \{\text{MaxP}, \text{TotalP}\}$ (abbreviations of Max Power and Total Power). While the transmission powers must be assigned to the nodes so that the resulting graph (induced by the assigned powers in relation to the edges of the threshold graph) achieves $\mathbb{P}$, MaxP represents the maximum power among all assigned powers is minimized and TotalP represents the total power of all assigned powers is minimized. Note that for the problems studied in this paper, we only consider MaxP. For monotone graph properties, it is easy to see that the MaxP objective is equivalent to minimizing the power which is uniformly assigned to all nodes. Therefore, we will use this approach for MaxP throughout the remainder of this paper.

The model and problems we study are:

Definition 2.2 Constant Rate Mobile Network *(CRMN): In this model, each of the n nodes in the given network $\mathcal{N}$ is mobile. Associated with each node are its starting and ending Euclidean[7] positions in the unit time interval. It is assumed that each node moves at its own uniform rate and direction throughout the time interval. The goal is to minimize the power uniformly assigned to every node such that the network $\mathcal{N}$ achieves monotone property $\mathcal{P}$ throughout the unit time interval. Note that in the above, $\mathcal{P}$ could be any monotone property.*

For convenience, the CRMN problems are represented by $\langle$CRMN, $\mathcal{P}$, MaxP$\rangle$ in the remaining of this paper.

As can be seen from Definition 2.2, the CRMN model attempts to represent in a uniform manner the mobility patterns of all the nodes in the system. The underlying assumption is that each node moves along a straight line during each unit time interval; however, different nodes may move with different speeds and along different directions. We now point out how the CRMN model arises in the context of two of the mobility models proposed in the literature [5]. As a first example, consider the random way point mobility model [12] where the movement of each node consists of segments with a pause between successive segments. In each segment, an individual node is assumed to move with a uniform velocity along a straight line, as in the CRMN model. A second example is provided by the column mobility model, which is known to be useful in modeling the search or scan of a region [5]. In this model, each mobile node is assigned a reference point along a line (column). (The orientation of the line is changed periodically to move the search to a different part of the region.) For each time period, a mobile node moves along line segments around its reference point. For each line segment, the the node is assumed to move with a constant velocity. Thus, the node movement along each segment is again captured by the CRMN model.

An example scenario for the CRMN model arises in the context of communication among a group of satellites. In such a MANET, since each satellite typically moves at a constant speed and direction within a unit time interval (e.g. a few hours), the minimum signal transmission power for maintaining connectivity can be pre-computed on the ground by mission control. To do this, the ground center collects all the necessary information from each of the satellites, computes the power values centrally and sends the power value to the satellites. This ensures that the satellite network remains connected throughout its mission lifetime. This is one context in which our centralized algorithms (presented in later sections) can be used.

As another scenario that can be captured by the CRMN model, consider a set of robotic guards patrolling an area too dangerous (e.g. due to environmental issues, or in a war zone situation, enemy action) for humans. It is likely that a) the placement of a large number of fixed stations is prohibitively difficult so standard cellular communication is not possible, hence the use of a mobile ad hoc network is required, and b) the robots patrol in a fixed pattern on a periodic basis (e.g. if the region being patrolled is polygonal and each robot is going around the polygon). Using the fixed robot patrol patterns, a central station can pre-compute the transmission powers that the robots should use at each point along their routes. This pattern of robot mobility is precisely captured by the CRMN model.

In subsequent sections of the paper, we present centralized algorithms for topology control problems under the CRMN model. We discussed two example scenarios (namely communication among a group of satellites and communication among robots patrolling a region) where mobility can be captured using the CRMN model. We note that in these two scenarios, one can use centralized algorithms for topology control since the mobility patterns are known in advance. Thus, optimal power values can be precomputed and the nodes can be programmed to use appropriate power values at different times to minimize power consumption.

[7] Henceforth, when the meaning is clear, we omit the word "Euclidean".

2.4 Summary of Main Results

The main contributions of this paper are polynomial time algorithms and frameworks for each of the specified topology control problems in MANETs. Our main results are:

- A framework for solving ⟨CRMN, $\mathcal{P}$, MaxP⟩, where $\mathcal{P}$ can be any monotone property. For example, when $\mathcal{P}$ is Minimum Degree Constraint or Connected, the resulting algorithm runs in time $O(n^6 \log n)$.
- An improved algorithm for ⟨CRMN, Connected, MaxP⟩, which runs in time $O(n^4 \log^2 n)$.
- A framework for solving the decision version[8] of ⟨CRMN, $\mathcal{P}$, MaxP⟩, where $\mathcal{P}$ can be any monotone property. When $\mathcal{P}$ is Connected, the resulting algorithm runs in time $O(n^2 \log^2 n)$.

3 Additional Definitions & Stationary Networks

In this section we provide additional definitions and notations, and discuss some prior work for stationary networks.

3.1 Additional Definitions & Notation

We begin with three definitions concerning transmission powers and related graphs relative to stationary networks (later we will extend these to MANETs):

- We let $G^p(\mathcal{N})$ denote the undirected graph induced from a stationary network $\mathcal{N}$, when transmission power p is uniformly assigned to each node. That is, in $G^p(\mathcal{N})$, an edge is present between nodes u and v if and only if $p \geq \pi(u, v)$.
- A *threshold graph* is a complete undirected edge-weighted graph where each edge is of positive weight. The weight $weight(u, v)$ of each edge (u, v) is its *threshold* $\pi(u, v)$. A *threshold graph for a stationary network* $\mathcal{N}$ is a threshold graph with the same node set as $\mathcal{N}$ and where, for each edge (u, v), $weight(u, v) = \pi(u, v)$.
- Given an undirected edge-weighted graph $G = (V, E)$ and a power p, the *graph* $G^p(V)$ *induced from* G *under* p is a graph with the same node set as G, and with an edge (u, v) if and only if $p \geq weight(u, v)$.

[8] For an optimization problem, its decision version is: Given any proposed solution to the optimization problem, determine if that solution satisfies the objective (without optimization) of the problem. Specifically, the decision version of ⟨CRMN, $\mathcal{P}$, MaxP⟩ is to determine whether for a given a power p, the CRMN $\mathcal{N}$ achieves monotone property $\mathcal{P}$ under p.

3.2 Stationary Networks

The general form of topology control for stationary networks was first proposed by Ramanathan and Rosales-Hain [18]. Among the several results in that paper, they presented an algorithm for ⟨Undir, Connected, MaxP⟩ (where Undir refers to the undirected graph model for stationary networks) that minimizes the power uniformly assigned to any node in a stationary network such that the resulting network is connected. Subsequently, [15] extended that algorithm and provided a general polynomial framework for ⟨Undir, $\mathcal{P}$, MaxP⟩, where $\mathcal{P}$ is any monotone graph property. In this paper we make extensive use of the framework given in [18, 15] for ⟨Undir, $\mathcal{P}$, MaxP⟩ and we outline that algorithm and related concepts in the next several paragraphs. Please note that while we use some ideas from [18, 15], the results of [18, 15] are only for systems with stationary nodes. New ideas are needed to handle mobile nodes and all the necessary definitions and proofs are presented in Section 4.

Recall that for MaxP, it is standard to assign the same power to all nodes. In this context, the topology control problem for stationary networks is to induce a graph $G^p(\mathcal{N})$ from the network under a uniform power p, such that $G^p(\mathcal{N})$ achieves a monotone property $\mathcal{P}$ (i.e. $\mathcal{N}$ achieves monotone property $\mathcal{P}$ under power p). Considering Connected as an example monotone property, note that given a stationary network $\mathcal{N}$ consisting of n nodes and a power p, the graph $G^p(\mathcal{N})$ is easily constructed in time $O(n^2)$. Likewise, given $G^p(\mathcal{N})$, it can be determined in time $O(n^2)$ if that graph is connected. It follows that given network $\mathcal{N}$ and power p, $O(n^2)$ time is sufficient to check if $\mathcal{N}$ is connected under power p.

With these preliminaries concluded, the framework described in [15] is shown in Figure 1. That framework is based on the insight that p must come from among the threshold values associated with node pairs in $\mathcal{N}$. This permits the framework to do a binary search over those threshold values searching for the least p such that $\mathcal{N}$ achieves a monotone property $\mathcal{P}$ under power p.

Thus, we solve ⟨Undir, Connected, MaxP⟩ by replacing $\mathcal{P}$ with Connected in Framework 1. Subsequently, the running time is $O(n^2 \log n)$ as de-

Input: A stationary network $\mathcal{N}$ with node set V, and a monotone property $\mathcal{P}$.

Output: The minimum power p_{min} such that $\mathcal{N}$ achieves $\mathcal{P}$ under p_{min}.

Steps:

1. Construct threshold graph $G_{th} = (V, E)$ from $\mathcal{N}$.
2. Sort E by threshold values, and use binary search to locate the least $p_{min} \in E$ such that $\mathcal{N}$ achieves $\mathcal{P}$ under power p_{min}.
3. Return p_{min}.

Figure 1: Framework 1 for $\langle$Undir, $\mathcal{P}$, MaxP$\rangle$ — MinMaxGraphFor$\mathcal{P}$

scribed in [18], since there are at most $O(\log n)$ powers that need to be checked and each checking (for Connected) takes time $O(n^2)$. Establishing a convention of naming the framework, we let Framework 1 be called MinMaxGraphFor$\mathcal{P}$. When $\mathcal{P}$ is Connected, the corresponding algorithm is called MinMaxGraphForConn.

We will make use of MinMaxGraphFor$\mathcal{P}$ as a subroutine throughout this paper. Additionally, for the convenience of future discussion, we introduce a meta function CheckGraphFor$\mathcal{P}$ with the following significance: Given property $\mathcal{P}$ and power p, CheckGraphFor$\mathcal{P}(\mathcal{N}, p)$ is called to solve the decision version of $\langle$Undir, $\mathcal{P}$, MaxP$\rangle$, that is, checking whether the stationary network $\mathcal{N}$ achieves $\mathcal{P}$ under p. Note that CheckGraphFor$\mathcal{P}$ is implied in step 2 of Framework 1. It simply goes through all the edges to check whether $\mathcal{P}$ is satisfied. When $\mathcal{P}$ is Connected, the algorithm is CheckGraphForConn($\mathcal{N}$, p) and runs in time $O(n^2)$.

4 Constant Rate Mobile Networks

In this section we consider topology control for monotone properties in constant rate mobile networks (CRMN). An instance $\mathcal{N}$ of a CRMN is a MANET in which all the n nodes are moving and each node moves at its own uniform rate and direction independently throughout a unit time interval. Associated with each moving node V_i are its starting and ending Euclidean positions in the unit time interval. The goal is to minimize the power uniformly assigned to any node in the MANET such that the network achieves a monotone property throughout the unit time interval.

In this section, for a moving node V_i, we refer to its starting position as V_i, to its ending position as V_i', and to the vector $\vec{v}_i = \overrightarrow{V_iV_i'}$ [9] as its moving route.

Our approach to solving an instance of CRMN is based on partitioning the unit time interval into *time slots*. We refer to this as *time slicing*. Note that at any time *instant*, since the positions of all of the moving nodes will be known, a threshold graph can be constructed. Our goal is to use time slicing to produce constant-order time slots such that *the ordering of the sorted list of threshold graph edges is invariant within each time slot*. Because of the invariance of this ordering, there is a specific edge which determines for every instant in the time slot the minimum power p such that the induced graph $G^p(\mathcal{N})$ (at that time instant) achieves the specified monotone property. Our framework will find this edge for every time slot, then compute the largest threshold value for each such edge within its time slot, and finally select the largest of these values as the overall solution to the instance of CRMN.

This section is organized as follows. In the first subsection, we introduce notation and terminology related to distance and threshold functions. In subsection 4.2, we provide the specifics of constant-connectivity time slots (a weaker version of constant-order time slots) and present a framework for solving the decision version of the CRMN problem. Then, in subsection 4.3, we provide the specifics of constant-order time slots and present a framework for solving the optimization version of the CRMN problem. Finally in subsection 4.4, we provide a faster algorithm to solve an instance of CRMN when property $\mathcal{P}$ is Connected.

4.1 Distance Functions & Threshold Functions

Since $|\vec{v}_i|$ is the length of the moving route of node V_i in a unit time interval, it follows that $|\vec{v}_i|$ is also the moving speed of V_i. Further, at any instant t in $[0, 1]$, the position of node V_i is given by $\vec{v}_i \cdot t$.

Consider two moving nodes A and B. We define the *distance function* between nodes A and B to be $d_{AB}(t)$. At any time instant t of the unit time interval, this function gives the distance between moving nodes A and B. Recall that these two nodes each

[9] $\vec{v}_i$ represents the vector $\overrightarrow{V_iV_i'}$ starting at V_i and ending at V_i'.

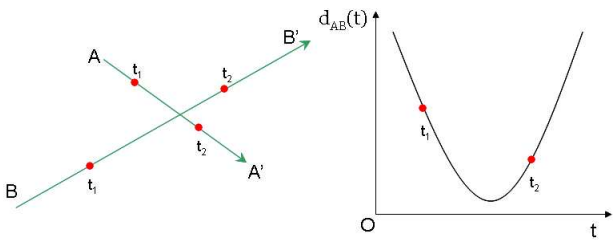

Figure 2: A Distance Example — Parabola

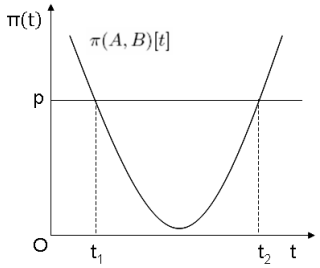

Figure 3: An Example of Constant-Connectivity Time Slots

moves at a constant direction and rate throughout the unit time interval. Note that these rates need not be the same. In a coordinate system with the X-axis being time t and the Y-axis being the value of distance function $d_{AB}(t)$, that distance function is a parabola. An example is shown in Figure 2.

The *threshold function* between A and B is $\pi(A,B)[t] = d^{\alpha}_{AB}(t)$, where α is again the attenuation constant associated with path loss. Hence, the threshold function is also drawn as a parabola in a coordinate system with the X-axis being time t and the Y-axis being the value of threshold function $\pi(A,B)[t]$. At any time instant t, this function gives the threshold between moving nodes A and B. The *threshold between A and B in a time slot* $[t_g, t_h]$ is given by $\pi(A,B)[t_g,t_h] = \text{MAX}(\pi(A,B)[t_g], \pi(A,B)[t_h])$. Here it is easy to see that for any $t \in [t_g, t_h]$, $\pi(A,B)[t] \leq \pi(A,B)[t_g,t_h]$.

Finally, a distance function is the square root of a quadratic function. Since there are at most two real solutions[10] to a quadratic equation, the following lemma is obvious:

Lemma 4.1 *Given any two non-identical distance functions $d_{AB}(t)$ and $d_{CD}(t)$, there are at most two real solutions for t to the equation $d_{AB}(t) = d_{CD}(t)$.*

Since a threshold function is $\pi(A,B)[t] = d^{\alpha}_{AB}(t)$, it follows that:

Corollary 4.2 *Given any two non-identical threshold functions $\pi(A,B)[t]$ and $\pi(C,D)[t]$, there are at most two real solutions for t to the equation $\pi(A,B)[t] = \pi(C,D)[t]$.*

4.2 A Framework for the Decision Version of ⟨CRMN, $\mathcal{P}$, MAXP⟩

In this subsection, we describe a framework to solve the *decision* version of CRMN problems. That is, given a power p and an instance $\mathcal{N}$ of CRMN, to decide whether $\mathcal{N}$ achieves a monotone property $\mathcal{P}$ under p.

As noted in the overview, our framework for the decision version of CRMN is based on time slicing. For each pair of distinct nodes V_i and V_j in $\mathcal{N}$, we calculate the real solutions (if any) to the equation $\pi(V_i,V_j)[t] = p$. The resulting solutions are called *slicing points*. Note that if $\pi(V_i,V_j)[t] \equiv p$, then V_i is constantly connected to V_j, and no slicing point will be recorded for use in the following. We collect all of the $O(n^2)$ distinct slicing points in a sorted list. If there are identical slicing points, they are represented by a single point in this list. Adjacent points in this sorted list define a time slot. An important fact is that connectivity of two moving nodes changes only when the edge between them either comes into, or goes out of, existence. This can only occur around a slicing point. It follows that each time slot is a *constant-connectivity time slot*. More precisely, a time slot $[t_x, t_y]$ is a constant-connectivity time slot if given a power p and a graph $G = (V,E)$, the following conditions hold:

- $\forall (V_i,V_j) \in E, \forall t \in (t_x,t_y) : \pi(V_i,V_j)[t] \leq p$
- $\forall (V_i,V_j) \notin E, \nexists t' \in (t_x,t_y) : \pi(V_i,V_j)[t'] \leq p.$

Note that the slicing points defining the time slot are not included in the above calculations. To see why these points must be excluded, consider consecutive time slots $[t_{i-1},t_i]$ and $[t_i,t_{i+1}]$, and assume some edge (V_j,V_k) does not exist at an interior point of $[t_{i-1},t_i]$, but does exist at an interior point of $[t_i,t_{i+1}]$. Thus, this edge comes into existence at slicing point t_i. Clearly, t_i cannot be included in the constant-connectivity calculation for $[t_{i-1},t_i]$. Fortunately, as we will show later there is no need to explicitly check connectivity at the slicing points.

An example of constant-connectivity time slots is shown in Figure 3. There are three constant-

[10]Imaginary solutions do not have any physical significance in this context.

connectivity time slots: $[t_1, t_2]$, the time ending at t_1 and the time beginning with t_2.

Having partitioned the unit time interval into constant-connectivity time slots, our framework simply checks the monotone property $\mathcal{P}$ for each such time slot. Network $\mathcal{N}$ achieves $\mathcal{P}$ under power p if and only if it achieves $\mathcal{P}$ under p in each such time slot. Since the connections among nodes in network $\mathcal{N}$ under power p are interpreted as the edges in the induced graph $G^p(\mathcal{N})$, it follows that it is only necessary to check $\mathcal{P}$ in the induced graph for each constant-connectivity time slot. The details are given in Framework 2 (Figure 4).

The remainder of this subsection is devoted to a proof of correctness and running time analysis for Framework 2. To prove the correctness, we begin with the following lemma.

Lemma 4.3 *Given a constant-connectivity time slot $[t_k, t_{k+1}]$, any edge present at an instant $t'_k \in (t_k, t_{k+1})$ is also present at any instant $t''_k \in [t_k, t_{k+1}]$.*

Proof: Since $[t_k, t_{k+1}]$ is a constant-connectivity time slot, edge set E of $G_k = (V, E)$ at $t'_k \in (t_k, t_{k+1})$ is invariant in (t_k, t_{k+1}). Note that both t_k and t_{k+1} are either the first or last instant that some edge is present in E. Thus, any edge present at an instant $t'_k \in (t_k, t_{k+1})$ is also present at any instant $t''_k \in [t_k, t_{k+1}]$. ■

Theorem 4.4 *Framework 2 returns true if and only if network $\mathcal{N}$ achieves the monotone property $\mathcal{P}$ under power p.*

Proof: We show that network $\mathcal{N}$ achieves $\mathcal{P}$ under power p if and only if every graph G_k (constructed at step 4.*b* in Framework 2) achieves $\mathcal{P}$.

If some G_k does not achieve $\mathcal{P}$, then clearly $\mathcal{N}$ does not achieve $\mathcal{P}$ under power p.

Conversely, if network $\mathcal{N}$ does not achieve $\mathcal{P}$ under power p, then at some instant $t \in [0, 1]$ the induced graph at t under power p does not achieve $\mathcal{P}$. If t is not a slicing point, then t is in some (t_k, t_{k+1}). Since (t_k, t_{k+1}) is a constant-connectivity time slot, it follows that G_k does not achieve $\mathcal{P}$. If t is a slicing point, say t_k, then recall that t_k is either the first or last instant at which some edge is present. In either case, it follows that the edge set of G_k as constructed at step 4.*b* in Framework 2 is a subset of the edge set present at t_k. Hence, G_k does not achieve $\mathcal{P}$. The theorem follows. ■

Theorem 4.5 *If* CheckGraphFor$\mathcal{P}$ *runs in time R_c, then Framework 2 runs in time $O(\text{MAX}(n^4, n^2 R_c))$.*

Input: An instance $\mathcal{N}$ of $CRMN$, a power p, and a monotone property $\mathcal{P}$.

Output: If $\mathcal{N}$ achieves $\mathcal{P}$ under power p, return true; else return false.

Steps:

1. $T \leftarrow \emptyset$: a set of slicing points.
2. For each two distinct nodes V_i, V_j in $\mathcal{N}$ do,
   (a) If the equation $\pi(V_i, V_j)[t] = p$ is a linear equation,
      i. Compute the solution t_g to the equation $\pi(V_i, V_j)[t] = p$.
      ii. If $t_g \in (0, 1)$, then $T \leftarrow T \cup \{t_g\}$.
   (b) If the equation $\pi(V_i, V_j)[t] = p$ is a quadratic equation,
      i. Compute the real solutions t_g and t_h to the equation $\pi(V_i, V_j)[t] = p$.
      ii. If $t_g \in (0, 1)$, then $T \leftarrow T \cup \{t_g\}$.
      iii. If $t_h \in (0, 1)$, then $T \leftarrow T \cup \{t_h\}$.
3. Let ST be the sorted list of distinct values in T, and let $\mathcal{T}_k = [t_k, t_{k+1}]$ be the k^{th} time slot of the unit time interval as defined by the adjacent values in ST.
4. For each $\mathcal{T}_k$ do,
   (a) $t'_k \leftarrow$ an interior time instant of $\mathcal{T}_k$.
   (b) Construct a graph $G_k = (V, E)$, where V is the set of nodes in $\mathcal{N}$ and $E = \{(V_i, V_j) : \pi(V_i, V_j)[t'_k] \leq p\}$.
   (c) If CheckGraphFor$\mathcal{P}(G_k)$ is false, then return false.
5. Return true.

Figure 4: Framework 2 for the Decision Version of $\langle$CRMN, $\mathcal{P}$, MaxP$\rangle$ — CheckCRMNFor$\mathcal{P}$

Proof: In Framework 2, step 2 takes $O(n^2)$ time, since there are $O(n^2)$ equations and each can be solved in time $O(1)$. Obviously, step 3 takes $O(n^2 \log n)$ time. Now, consider the running time of step 4. Implementing step 4 as described in Framework 2 requires $O(\text{MAX}(n^4, n^2 R_c))$ time, since there are $O(n^2)$ time slots, and for each k, constructing G_k also takes $O(n^2)$ time. The theorem follows. ■

Corollary 4.6 *If $\mathcal{P}$ is* Connected*, then Frame-*

work 2 runs in worst case time $O(n^2 \log^2 n)$ and expected time $O(n^2 \log n (\log \log n)^3)$.

Proof: From Theorem 4.5, the running time of Framework 2 for CONNECTED is $O(n^4)$. However, that running time can be reduced by observing that G_k and G_{k+1} for the consecutive time slots (t_k, t_{k+1}) and (t_{k+1}, t_{k+2}) differ only by the insertion or deletion of an edge [11]. This allows us to implement step 4 using dynamic graph algorithms [10, 21]. Such algorithms process a sequence of update and query operations interspersed in any order, where the *update* operations include the insertion and deletion of edges, and the *query* operation is a query about some property of the graph[12]. Using [10], these operations for CONNECTED can be implemented in $O(\log^2 n)$ amortized time per update and $O(\log n / \log \log n)$ amortized time per query. Thus, over all $O(n^2)$ iterations, step 4.*b* can be implemented in $O(n^2 \log^2 n)$ worst case time and step 4.*c* can be implemented in $O(n^2 \log n / \log \log n)$ worst case time. Likewise from a result of [21] in which queries require $O(\log n / \log \log \log n)$ time while updates require $O(\log n (\log \log n)^3)$ *expected* amortized time, it follows that over all the $O(n^2)$ iterations, steps 4.*b* and 4.*c* can be implemented in $O(n^2 \log n (\log \log n)^3)$ expected amortized time and $O(n^2 \log n / \log \log \log n)$ time. Note that step 4.*c* uses these dynamic graph operations in place of CHECKGRAPHFOR$\mathcal{P}$. Hence, for CONNECTED, the worst case running time is $O(n^2 \log^2 n)$ and the expected running time is $O(n^2 \log n (\log \log n)^3)$. ■

4.3 A Framework for ⟨CRMN, $\mathcal{P}$, MAXP⟩

In this subsection, we give a framework for solving the *optimization* version of CRMN problems. That is, given an instance $\mathcal{N}$ of CRMN, to find the minimum power p_{min} uniformly assigned to all nodes such that $\mathcal{N}$ achieves a monotone property $\mathcal{P}$ under p_{min}. We begin with the following lemma:

[11] This is assuming that there is only one slicing point at t_{k+1}. When there are multiple slicing points at t_{k+1}, the update operations described in this proof are applied to each of those slicing points.

[12] In [10], there are only two graph properties (related to this paper) handled by their dynamic graph algorithms. These properties are CONNECTED and 2-NODE/EDGE-CONNECTED. However, it is not hard to see that MINIMUM DEGREE CONSTRAINT can be processed incrementally in constant time as follows. Let every node be associated with a counter that records the number of incident edges on that node. The update operation is that when an edge comes into or goes out of existence, the degree of each node is accounted for by increasing or decreasing the counter associated with each endpoint of that edge.

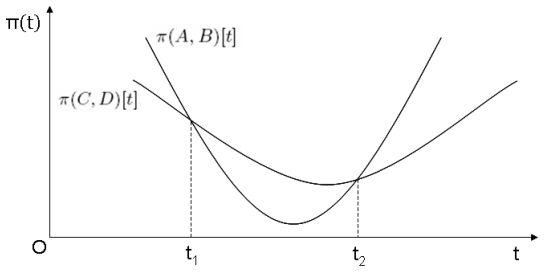

Figure 5: An Example of Constant-Order Time Slots

Lemma 4.7 *Given moving nodes A (which moves from A to A′) and B (which moves from B to B′),* MAX*$(\pi(A,B), \pi(A',B'))$ is the minimum power that can be assigned to both A and B such that nodes A and B are continuously connected as they move simultaneously from their respective starting positions to ending positions at their respective constant rates.*

Proof: Since the distance function $d_{AB}(t)$ between nodes A and B is the square root of a quadratic function, the maximum distance between nodes A and B is MAX$(|\overline{AB}|, |\overline{A'B'}|)$. Hence, the lemma follows. ■

Next, letting $G_t = (V, E_t)$ be the threshold graph at time instant t, we state two definitions. (We use $|e_i|$ to denote the length of an edge e_i in G_t.)

Edge $e_m \in E_t$ is a *MinMax edge* at t, if $G'_t = (V, E'_t)$ achieves a monotone property $\mathcal{P}$, where $E'_t = \{e_i \in E_t : |e_i| \leq |e_m|\}$, and $G''_t = (V, E''_t)$ does not achieve $\mathcal{P}$, where $E''_t = E_t \setminus \{e_i \in E_t : |e_i| \geq |e_m|\}$.

Time slot $[t_x, t_y]$ is a *constant-order time slot* if there exists an ordered list E^* of the network edges such that at any time instant $t \in [t_x, t_y]$, E^* is an ordered list by length of the edges in G_t.

An example of constant-order time slots is shown in Figure 5. Since $\pi(A,B)[t]$ is always greater than or equal to $\pi(C,D)[t]$ before t_1 and after t_2, and since $\pi(A,B)[t]$ is smaller than or equal to $\pi(C,D)[t]$ throughout $[t_1, t_2]$, there are three constant-order time slots: $[t_1, t_2]$, the time ending at t_1 and the time beginning with t_2.

With these definitions in hand, we have the following lemma.

Lemma 4.8 *Given a constant-order time slot $[t_x, t_y]$ and a time instant $t' \in (t_x, t_y)$, if edge e_m is a MinMax edge at t', then e_m is a MinMax edge at any instant $t'' \in [t_x, t_y]$.*

Proof: By way of contradiction, assume that there exists an instant $t'' \in [t_x, t_y]$ such that e_m is not a MinMax edge at t''. Let e_p be a MinMax edge at t'',

and note that $|e_p|$ cannot be equal to $|e_m|$ at t'', since then e_m is also a MinMax edge at t''.

Let $G = (V, E)$ be the threshold graph for an instance $\mathcal{N}$ of CRMN. There are two cases.

Case 1: $|e_p| > |e_m|$.

Here, let $G'' = (V, E'')$ be the graph at t'' with $E'' = E \setminus \{e_i : |e_i| \geq |e_p|\}$. Since e_p is a MinMax edge at t'', G'' does not achieve a monotone property $\mathcal{P}$. Let $G' = (V, E')$ be the graph at t' with $E' = \{e_i : |e_i| \leq |e_m|\}$. Since e_m is a MinMax edge at t', G' achieves $\mathcal{P}$. Since the ordering of the sorted edge list is invariant and $|e_p| > |e_m|$, E' is a subset of E''. Hence, G'' does not achieve $\mathcal{P}$ while G' achieves $\mathcal{P}$. This is a contradiction.

Case 2: $|e_p| < |e_m|$.

Here, let $G'' = (V, E'')$ be the graph at t'' with $E'' = \{e_i : |e_i| \leq |e_p|\}$. Since e_p is a MinMax edge at t'', G'' achieves a monotone property $\mathcal{P}$. Let $G' = (V, E')$ be the graph at t' with $E' = E \setminus \{e_i : |e_i| \geq |e_m|\}$. Since e_m is a MinMax edge at t', G' does not achieve $\mathcal{P}$. Since the ordering of the sorted edge list is invariant and $|e_p| < |e_m|$, E' is a superset of E''. Hence, G'' achieves $\mathcal{P}$ while G' does not achieve $\mathcal{P}$. This is also a contradiction.

Thus, e_m is a MinMax edge in G at any instant $t'' \in [t_x, t_y]$. ■

Based on Lemma 4.8, the framework for solving an instance of CRMN will be presented in two phases. We first give an algorithm for time slicing that produces the constant-order time slots, then we provide the framework optimally solving an instance of CRMN.

Our algorithm for time slicing is as follows. For each pair of non-identical threshold functions $\pi(V_a, V_b)[t]$ and $\pi(V_c, V_d)[t]$, we calculate the real solutions (if any) to the equation $\pi(V_a, V_b)[t] = \pi(V_c, V_d)[t]$ (equivalently, to the equation $d_{V_a V_b}(t) = d_{V_c V_d}(t)$). The resulting solutions are called *slicing points*, and the edges (V_a, V_b) and (V_c, V_d) are said to define the slicing points. Note that any pair of distinct edges define at most two slicing points. We collect all $O(n^4)$ distinct slicing points into a sorted list. Adjacent points in this sorted list define a time slot. Here, given a threshold graph $G = (V, E)$ at an instant in a time slot $[t_x, t_y]$, the ordering of a sorted edge list for E is invariant in $[t_x, t_y]$. It follows that each such time slot is a constant-order time slot. Algorithm 3 presents the details for time slicing.

Lemma 4.9 *Algorithm 3 runs in time $O(n^4 \log n)$.*

Proof: There are $O(n^2)$ edges and $O(n^2)$ threshold functions. Hence, there are $O(n^4)$ equations to be solved. Since there are at most two solutions for an equation and each can be solved in time $O(1)$, step 2 takes time $O(n^4)$. However, since step 3 takes $O(n^4 \log n)$ time for sorting the $O(n^4)$ slicing points, Algorithm 3 runs in time $O(n^4 \log n)$. ■

Input: An instance $\mathcal{N}$ of $CRMN$.

Output: A set of constant-order time slots $\mathcal{T}_k$.

Steps:

1. $T \leftarrow \emptyset$: a set of slicing points.
2. For each two distinct threshold functions $\pi(V_a, V_b)[t]$ and $\pi(V_c, V_d)[t]$ do,
   (a) If the equation $\pi(V_a, V_b)[t] = \pi(V_c, V_d)[t]$ is a linear equation,
      i. Compute the solution t_g to the equation $\pi(V_a, V_b)[t] = \pi(V_c, V_d)[t]$.
      ii. If $t_g \in (0, 1)$, then $T \leftarrow T \cup \{t_g\}$.
   (b) If the equation $\pi(V_a, V_b)[t] = \pi(V_c, V_d)[t]$ is a quadratic equation,
      i. Compute the real solutions t_g and t_h to the equation $\pi(V_a, V_b)[t] = \pi(V_c, V_d)[t]$.
      ii. If $t_g \in (0, 1)$, then $T \leftarrow T \cup \{t_g\}$.
      iii. If $t_h \in (0, 1)$, then $T \leftarrow T \cup \{t_h\}$.
3. Let ST be a sorted list of the values in T, and let $\mathcal{T}_k = [t_k, t_{k+1}]$ be the k^{th} time slot of the unit time interval as defined by the adjacent values in ST, where $1 \leq k \leq w$.
4. Return $\{\mathcal{T}_k : 1 \leq k \leq w\}$.

Figure 6: Algorithm 3 — TIMESLICING

Making use of TIMESLICING, we present our Framework 4 for solving an instance of CRMN in Figure 7. In Framework 4, step 1 slices the unit time interval into constant-order time slots. For each constant-order time slot, step 2 builds a threshold graph at an internal instant of the time slot, finds a MinMax edge in that threshold graph, and calculates the largest threshold value of the MinMax edge in the constant-order time slot. Step 3 returns the largest threshold value among all of the time slots as the solution to the instance of CRMN. We now establish the correctness and the running time of Framework 4.

Theorem 4.10 *Framework 4 computes the optimal solution to an instance of CRMN,*

Input: An instance $\mathcal{N}$ of $CRMN$, and a monotone property $\mathcal{P}$.

Output: The minimum power p_{min} such that $\mathcal{N}$ achieves $\mathcal{P}$ under p_{min}.

Steps:

1. $\{\mathcal{T}_k : 1 \leq k \leq w\} \leftarrow$ TimeSlicing($\mathcal{N}$).
2. For k from 1 to w do,
   (a) Let t'_k be an interior time instant in $\mathcal{T}_k$, and construct a threshold graph $G_k = (V, E)$ at t'_k.
   (b) $p_{m_k} \leftarrow$ MinMaxGraphFor$\mathcal{P}$(G_k).
   Construct $G_k^{p_{m_k}}(V)$, the graph induced from G_k by p_{m_k}. Select from G_k an edge $e_{m_k} = (V_{m_k}, V_{m'_k})$ such that $\pi(V_{m_k}, V_{m'_k})[t'_k] = p_{m_k}$.
   Let $p_{m_k} \leftarrow \pi(V_{m_k}, V_{m'_k})[t_k, t_{k+1}]$.
   (c) Put p_{m_k} into set P.
3. $p_{min} \leftarrow$ the largest value in P.
4. Return p_{min}.

Figure 7: Framework 4 for ⟨CRMN, $\mathcal{P}$, MaxP⟩ — MinMaxCRMNFor$\mathcal{P}$

and the running time is $O(\text{MAX}(n^6, n^4 R_o))$*, if* MinMaxCRMNFor$\mathcal{P}$ *runs in time* R_o*.*

Proof: Since constant-order time slots are produced by TimeSlicing, it follows directly from Lemma 4.8 that a MinMax edge is found in the induced graph within each constant-order time slot. Hence, Framework 4 computes the optimal solution for each constant-order time slot. The overall optimal solution is clearly the largest of these values.

In Framework 4, step 1 runs in time $O(n^4 \log n)$. There are $O(n^4)$ iterations in step 2, and step 2.a takes time $O(n^2)$ to construct a threshold graph, while step 2.b takes time R_o to find a MinMax edge. Hence, step 2 runs in time $O(\text{MAX}(n^6, n^4 R_o))$. Step 3 takes time $O(n^4)$ to find the largest value.

The theorem follows. ■

When $\mathcal{P}$ is specified as Connected, since MinMaxGraphForConn runs in time $O(n^2 \log n)$, Framework 4 runs in time $O(n^6 \log n)$. In the next subsection, we provide a faster algorithm for Connected.

4.4 A Faster Algorithm for Connectivity

To improve upon the running time of Framework 4 when $\mathcal{P}$ is Connected, rather than independently building a threshold graph and finding a MinMax edge for each constant-order time slot, we instead produce the MinMax edges incrementally by using dynamic graph algorithms.

Here, the two keys to improving the running time are (1) avoiding an explicit construction of threshold graphs, and (2) computing the MinMax edge incrementally. To help achieve these two objectives, we incrementally construct the *induced* graph $G_k^{p_{m_k}}(V)$, where p_{m_k} is the threshold value of a MinMax edge for threshold graph G_k.

The key step of this implementation is as follows. Assume that for constant-order time slot $[t_{k-1}, t_k]$ we have $G_{k-1}^{p_{m_{k-1}}}(V)$ and the MinMax edge for G_{k-1}. For $[t_k, t_{k+1}]$, we update these two items as follows.

Let e_x and e_y be the two edges that define the slicing point at t_k [13]. It follows that sorted lists of the edges during $[t_{k-1}, t_k]$ and during $[t_k, t_{k+1}]$ differ only in that e_x and e_y interchange their positions, since they define a slicing point at t_k. Note that e_x and e_y are adjacent in both sorted edge lists.

Assume $|e_x| < |e_y|$ in (t_{k-1}, t_k). Since every time slot is a constant-order time slot, $G_{k-1}^{p_{m_{k-1}}}(V)$ for $[t_{k-1}, t_k]$ differs from $G_k^{p_{m_k}}(V)$ of $[t_k, t_{k+1}]$ only in whether or not e_x and e_y are present. To see this clearly, we consider three cases.

Case 1: Let $e_{m_{k-1}}$ be a MinMax edge for $[t_{k-1}, t_k]$. If $|e_x|$ and $|e_y|$ are both less than or both greater than $|e_{m_{k-1}}|$ in $[t_{k-1}, t_k]$, then $e_{m_{k-1}}$ is also a MinMax edge for $[t_k, t_{k+1}]$. That is $e_{m_k} = e_{m_{k-1}}$, hence $G_k^{p_{m_k}}(V) = G_{k-1}^{p_{m_{k-1}}}(V)$.

Case 2: If $|e_x|$ is less than $|e_{m_{k-1}}|$ and $|e_y|$ is equal to $|e_{m_{k-1}}|$ (i.e., e_y is a MinMax edge) in $[t_{k-1}, t_k]$, then $G_{k-1}^{p_{m_{k-1}}}(V)$ is updated to obtain $G_k^{p_{m_k}}(V)$ as follows: delete e_x and check whether the resulting graph G' is connected. If so, e_y is a MinMax edge for $[t_k, t_{k+1}]$ and $G_k^{p_{m_k}}(V) = G'$. If G' is not connected, then e_x is inserted into G'. The resulting graph is $G_k^{p_{m_k}}(V)$ and e_x is a MinMax edge for $[t_k, t_{k+1}]$.

Case 3: If $|e_x|$ is equal to $|e_{m_{k-1}}|$ (i.e., e_x is a MinMax edge) in $[t_{k-1}, t_k]$, then $|e_y|$ is greater than $|e_{m_{k-1}}|$ in $[t_{k-1}, t_k]$, in which case $G_{k-1}^{p_{m_{k-1}}}(V)$ is updated to obtain $G_k^{p_{m_k}}(V)$ as follows: delete e_x, insert e_y, and

[13] Note that when there are identical threshold functions, multiple pairs of edges might define the same slicing points. In that case, the procedure described in the remainder of the proof is carried out for each such pair of edges.

Input: An instance $\mathcal{N}$ of $CRMN$, and $\mathcal{P}$ is CONNECTED.

Output: The minimum power p_{min} such that $\mathcal{N}$ achieves CONNECTED under p_{min}.

Steps:

1. $\{\mathcal{T}_k : 1 \leq k \leq w\} \leftarrow$ TIMESLICING($\mathcal{N}$).
2. Let t'_1 be an interior time instant in $\mathcal{T}_1$, and construct a threshold graph $G_1 = (V, E)$ at t'_1.
3. $p_{m_1} \leftarrow$ MINMAXGRAPHFOR$\mathcal{P}(G_1)$.
   Construct $G_1^{p_{m_1}}(V)$, the graph induced from G_1 by p_{m_1}. Select from G_1 an edge $e_{m_1} = (V_{m_1}, V_{m'_1})$ such that $\pi(V_{m_1}, V_{m'_1})[t'_1] = p_{m_1}$.
   Let $p_{m_1} \leftarrow \pi(V_{m_1}, V_{m'_1})[t_1, t_2]$.
4. $P \leftarrow \{p_{m_1}\}$.
5. For k from 2 to w do,
   (a) UPDATE($\mathcal{T}_k$, $e_{m_{k-1}}$, $G_{k-1}^{p_{m_{k-1}}}(V)$), which returns e_{m_k} and $G_k^{p_{m_k}}(V)$.
   (b) $p_{m_k} \leftarrow \pi(V_{m_k}, V_{m'_k})[t_k, t_{k+1}]$, where $e_{m_k} = (V_{m_k}, V_{m'_k})$.
   (c) $P \leftarrow P \cup \{p_{m_k}\}$.
6. $p_{min} \leftarrow$ the largest value in P.
7. Return p_{min}.

Figure 8: Algorithm 5 for ⟨CRMN, CONNECTED, MAXP⟩ — MINMAXCRMNFORCONN

check whether the resulting graph G' is connected. If so, e_y is a MinMax edge for $[t_k, t_{k+1}]$ and $G_k^{p_{m_k}}(V) = G'$. If G' is not connected, then e_x is inserted into G'. The resulting graph is $G_k^{p_{m_k}}(V)$ and e_x is a MinMax edge for $[t_k, t_{k+1}]$.

The improved algorithm is presented as Algorithm 5 in Figure 8. The steps described above in cases 1, 2 and 3 are assumed to be carried out by a procedure UPDATE which takes as parameters $\mathcal{T}_k$, $e_{m_{k-1}}$ and $G_{k-1}^{p_{m_{k-1}}}(V)$ and which returns e_{m_k} and $G_k^{p_{m_k}}(V)$.

Steps 2 - 5 of Algorithm 5 correspond to step 2 of Framework 4. Our approach for these steps is to utilize dynamic graph algorithms to construct the induced graphs $G_k^{p_{m_k}}(V)$. Specifically, steps 2 - 4 in Algorithm 5 provide the initial data structures and step 5 provides the incremental construction.

Since Algorithm 5 finds the MinMax edges for all constant-order time slots, we have:

Theorem 4.11 *The value p_{min} returned by Algorithm 5 is such that network $\mathcal{N}$ achieves* CONNECTED *under power p_{min}, and p_{min} is the minimum such power.*

Finally, we consider the running time of Algorithm 5.

Theorem 4.12 *Algorithm 5 runs in worst case time $O(n^4 \log^2 n)$.*

Proof: From Lemma 4.9, step 1 requires $O(n^4 \log n)$ time. Step 2 constructs a threshold graph in a straightforward fashion in $O(n^2)$. Since $\mathcal{P}$ is CONNECTED, step 3 takes time $O(n^2 \log n)$ for the call to MINMAXGRAPHFORCONN and time $O(n^2)$ to construct $G_1^{p_{m_1}}(V)$ and find edge e_{m_1}. For step 5, there are $O(n^4)$ iterations. In each iteration, step 5.a dominates the running time. By using dynamic graph algorithms, step 5.a is completed in time $O(\log^2 n)$. Specifically, for CONNECTED, in each iteration of that step, the UPDATE procedure does at most two edge insertions/deletions and one query of connectivity in constructing $G_k^{p_{mk}}(V)$ from $G_{k-1}^{p_{m(k-1)}}(V)$. Recall that the algorithm of Holm et al. [10] utilizes $O(\log^2 n)$ amortized time per insertion/deletion and $O(\log n / \log\log n)$ time per connectivity query. Since there are totally $O(n^4)$ constant-order time slots, the total running time of step 5 over all iterations is $O(n^4 \log^2 n)$. The theorem follows. ■

More generally, for any property $\mathcal{P}$ for which there exists a dynamic graph algorithm, we have:

Theorem 4.13 *If* MINMAXGRAPHFOR$\mathcal{P}$ *runs in time R_o and* UPDATE *runs in time R_u, then Algorithm 5 runs in worst case time $O(\max(n^4 \log n, n^4 R_u, R_o))$.*

Since the dynamic graph algorithm in [10] can also deal with the properties 2-NODE-CONNECTED and 2-EDGE-CONNECTED, in $O(\log^5 n)$ and $O(\log^4 n)$ respectively, it follows that:

Corollary 4.14 *When monotone property $\mathcal{P}$ is specified as* 2-NODE/EDGE-CONNECTED*, Algorithm 5 runs in worst case time $O(n^4 \log^5 n)$ for* 2-NODE-CONNECTED*, and runs in worst case time $O(n^4 \log^4 n)$ for* 2-EDGE-CONNECTED*.*

Using a result of [21] for dynamic graph connectivity in which connectivity queries require

$O(\log n/\log\log\log n)$ time while updates require $O(\log n(\log\log n)^3)$ expected amortized time, it follows that:

Corollary 4.15 *Algorithm 5 runs in expected time* $O(n^4 \log n(\log\log n)^3)$.

4.5 Example Problems

To gain some perspective on the various CRMN results, in this subsection we summarize the worst case running times for solving the decision and optimization CRMN problems for sample monotone properties: (1) CONNECTED, (2) 2-NODE-CONNECTED, (3) MINIMUM DEGREE CONSTRAINT OF l_d, and (4) BOUNDED DIAMETER OF 2. Since there are multiple algorithms and frameworks proposed in Section 4, we list the fastest running time for each property. Note that for the decision problems, when a fast dynamic graph algorithm is applicable in Framework 2, the resulting algorithm typically runs faster than the straightforward implementation. Likewise, for the optimization problems, when there is a fast dynamic graph algorithm for the property, Algorithm 5 typically runs faster than Framework 4.

Property	Decision	Optimization
CONNECTED	$O(n^2 \log^2 n)$	$O(n^4 \log^2 n)$
2-NODE-CONNECTED	$O(n^2 \log^5 n)$	$O(n^4 \log^5 n)$
MINIMUM DEGREE CONSTRAINT	$O(n^2 \log n)$	$O(n^4 \log n)$
BOUNDED DIAMETER OF 2	$O(n^5)$	$O(n^7 \log n)$

Since CONNECTED, 2-NODE-CONNECTED, and MINIMUM DEGREE CONSTRAINT OF l_d can be computed incrementally by using dynamic graph algorithms, the algorithms introduced in [10] are used. Specifically, we use an $O(\log^2 n)$ UPDATE algorithm for CONNECTED and an $O(\log^5 n)$ UPDATE algorithm for 2-NODE-CONNECTED. According to Theorem 4.12 and Theorem 4.14, the running times of the optimization algorithms for rows 1 and 2 are given. Similarly, by using dynamic graph algorithms, the running times of the decision algorithms for rows 1 and 2 are derived in a manner that is virtually identical to that for the optimization algorithms. In particular, the running time of the decision algorithm for row 1 is given by Corollary 4.6.

The dynamic graph algorithm for MINIMUM DEGREE CONSTRAINT was not presented in [10]. Recall however from Section 4.2 that its UPDATE algorithm can be easily implemented in time $O(1)$. It follows that the running times $O(n^2 \log n)$ for the decision algorithm and $O(n^4 \log n)$ for the optimization algorithm are determined by sorting the constant-connectivity or constant-order time slots. Specifically, the running time of the decision algorithm for row 3 is derived analogously to that for CONNECTED from Corollary 4.6 by using dynamic graph algorithms, and the running time of the optimization algorithm for row 3 is given by Theorem 4.13.

Since there is no known algorithm for incrementally implementing BOUNDED DIAMETER OF 2, the running times for row 4 are determined by Framework 2 and Framework 4. Since the naive decision algorithm for row 4 in stationary networks runs in time $O(n^3)$, according to Theorem 4.5, the running time of the decision algorithm for row 4 is $O(n^5)$. Likewise, since the naive optimization algorithm for row 4 in stationary networks runs in time $O(n^3 \log n)$, according to Theorem 4.10, the running time of the optimization algorithm for row 4 is $O(n^7 \log n)$.

5 Related Work

Recall that there are no previous theoretical results on topology control when there are moving nodes. In this section, we provide a brief overview for stationary networks.

As noted in Section 3.2, the general form of topology control considered in this paper was first proposed by Ramanathan and Rosales-Hain [18]. They presented efficient algorithms for the problems ⟨UNDIR, CONNECTED, MAXP⟩ and ⟨UNDIR, 2-NODE CONNECTED, MAXP⟩. In addition, they presented efficient distributed heuristics for those problems. Subsequently, [15] provided a general polynomial algorithm for minimizing maximum power for a range of properties.

Considerable work has been reported in the literature on a variety of other topology control problems. For instance, several groups of researchers have studied ⟨UNDIR, CONNECTED, TOTALP⟩, ⟨UNDIR, 2-NODE CONNECTED, TOTALP⟩ and ⟨UNDIR, DIAMETER K, MAXP⟩ [15, 14]. Here, TOTALP is the optimization objective for minimizing the total power used by the nodes. Likewise, work on the ⟨DIR, STRONGLY CONNECTED, TOTALP⟩ problem may be found in [6, 13, 8]. In most instances, the problems are shown to be **NP**-hard and the focus is on the development of approximation algorithms having either $O(\log n)$ or constant approximation ratios. Further, two new approximation algorithms for ⟨UNDIR, 2-NODE CONNECTED, TOTALP⟩ and ⟨UNDIR, 2-EDGE CONNECTED, TOTALP⟩ with an asymptotic approximation ratio of 8 are presented in [15]. Both of the approximation ratios are improved to 4 in [4]. References [2, 3] show the **NP**-hardness of ⟨UNDIR, CONNECTED, TOTALP⟩ and [3] presents a $(1 + \ln 2)$-approximation algorithm for that problem. The ap-

proximation ratio is improved to 5/3 in a journal submission of [3]. There are many other results using the TotalP objective and the reader is referred to [14, 17] for a thorough discussion.

6 Conclusions

In this paper, we described a simple model called CRMN for incorporating mobility into topology control problems for MANETs. In this model, for every node we are given the starting and ending positions, the moving rate, and direction of movement. Our focus was on finding the minimum power that can be assigned to all nodes such that the network achieves a monotone property under such power. Under the CRMN model, we developed polynomial time frameworks for solving both decision and optimization versions of topology control problems for monotone properties. For some specific monotone properties (e.g. Connected, 2-Node/Edge-Connected) we presented faster algorithms using results from fully dynamic graph algorithms.

The work reported in this paper provides a foundation for the development of distributed algorithms for the topology control problems in mobile ad hoc networks. A possible approach for developing distributed heuristics is to let each node v collect information about the speed and direction of the nodes which are within a small number of hops from v and use that information and our centralized algorithms to compute an optimal power value for an appropriate subnetwork. The computed values for various subnetworks must then be combined in a careful manner to obtain a power value which ensures that the network has the desired properties and the computed power value is reasonably close to the optimal value. Clearly, the key challenge in such an approach is the development of techniques needed to combine the information from various subnetworks in a distributed manner. The ideas developed in the context of centralized algorithms presented in this paper may be further helpful in achieving that goal.

Since this paper is among the first group of theoretical results for topology control incorporating mobility, there are many other directions for further research. A second direction for future work is the development of algorithms and frameworks for topology control problems under more general versions of CRMN. A third direction is to study topology control problems for minimizing the total power objective under the CRMN model and its generalizations.

Acknowledgments: We thank the reviewers for reading the manuscript and providing helpful suggestions.

References

[1] E. Althaus, G. Calinescu, I. Mandoiu, S. Prasad, N. Tchervenski, and A. Zelikovksy. Power efficient range assignment for symmetric connectivity in static ad hoc wireless networks. *Wireless Networks*, 12(3):287–299, May 2006.

[2] D. M. Blough, M. Leoncini, G. Resta, and P. Santi. On the symmetric range assignment problem in wireless ad hoc networks. *Proc. 2nd IFIP International Conference on Theoretical Computer Science*, 223:71–82, 2002.

[3] G. Calinescu, I. Mandoiu, and A. Zelikovsky. Symmetric connectivity with minimum power consumption in radio networks. *Proc. 2nd IFIP International Conference on Theoretical Computer Science*, 223:119–130, 2002.

[4] G. Calinescu and P.-J. Wan. Range assignment for high connectivity in wireless ad hoc networks. *Proc. International Conference on Ad hoc and Wireless Networks*, pages 235–246, 2003.

[5] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communication and Mobile Computing (WCMC)*, 2(5):483–502, February 2002.

[6] W. Chen and N. Huang. The strongly connecting problem on multihop packet radio networks. *IEEE Trans. Communication*, 37(3):293–295, March 1989.

[7] A. E. F. Clementi, P. Penna, and R. Silvestri. Hardness results for the power range assignment problem in packet radio networks. *Proc. Third International Workshop on Randomization and Approximation in Computer Science (APPROX 1999)*, 1671:195–208, July 1999.

[8] A. E. F. Clementi, P. Penna, and R. Silvestri. The power range assignment problem in packet radio networks in the plane. *Proc. 17th Annual Symposium on Theoretical Aspects of Computer Science (STACS 2000)*, pages 651–660, February 2000.

[9] M. T. Hajiaghayi, N. Immorlica, and V. Mirrokni. Power optimization in fault-tolerant topology control algorithms for wireless multihop networks. *MobiCom'03*, pages 300–312, September 2003.

[10] J. Holm, K. de Lichtenberg, and M. Thorup. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. *Journal of the ACM*, 48(4):723–760, July 2001.

[11] X. Jia, D. Kim, S. Makki, P. Wan, and C. Yi. Power assignment for k-connectivity in wireless ad hoc networks. *J. Combinatorial Optimization*, 9(2):213–222, March 2005.

[12] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In T. Imielinski and H. Korth, editors, *Mobile Computing*, volume 353, pages 153–181. Kluwer Academic Publishers, 1996.

[13] L. M. Kirousis, E. Kranakis, D. Krizanc, and A. Pelc. Power consumption in packet radio networks. *Proc. 14th Annual Symposium on Theoretical Aspects of Computer Science*, 1200:363–374, February 1997.

[14] S. O. Krumke, R. Liu, E. L. Lloyd, M. V. Marathe, R. Ramanathan, and S. S. Ravi. Topology control problems under symmetric and asymmetric power thresholds. *Proc. International Conference on Ad hoc and Wireless Networks*, 2865:187–198, October 2003.

[15] E. L. Lloyd, R. Liu, M. V. Marathe, R. Ramanathan, and S. S. Ravi. Algorithmic aspects of topology control problems for ad hoc networks. *Mobile Networks and Applications (MONET)*, 10(1-2):19–34, February-April 2005. (An earlier version appeared in MobiHoc 2002.).

[16] E. L. Lloyd, R. Liu, and S. S. Ravi. Approximating the minimum number of maximum power users in ad hoc networks. *Mobile Networks and Applications (MONET)*, 11(2):129–142, April 2006. (An earlier version appeared in ADHOC-NOW 2004.).

[17] E. L. Lloyd and S. S. Ravi. Topology control problems for wireless networks. In T. Gonzalez, editor, *Approximation Algorithms and Metaheuristics*, pages 67.1–67.20. Chapman & Hall/CRC, 2007.

[18] R. Ramanathan and R. Rosales-Hain. Topology control of multihop wireless networks using transmit power adjustment. *Proc. IEEE INFOCOM 2000*, pages 404–413, March 2000.

[19] T. S. Rappaport. *Wireless Communications: Principles and Practice*. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1996.

[20] G. Robins and A. Zelikovsky. Improved steiner tree approximation in graphs. *Proc. 11th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA 2000)*, pages 770–779, 2000.

[21] M. Thorup. Near-optimal fully-dynamic graph connectivity. *Proc. Thirty-second Annual ACM Symposium on Theory of Computing*, pages 343–350, 2000.

[22] L. Zhao, E. L. Lloyd, and S. S. Ravi. Topology control for constant rate mobile networks. *Proc. IEEE GLOBECOM'06-WirelessComm (6 pages)*, November 2006.