UNIVERSITY OF CALIFORNIA,
IRVINE

Connectivity Establishment and Maintenance for Sparse Network Graphs

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Mechanical and Aerospace Engineering

by

Kai Ding

Dissertation Committee:
Professor Homayoun Yousefi'zadeh, Advisor
Professor Faryar Jabbari, Chair
Professor Kenneth D. Mease
Professor Solmaz S. Kia

2018

DEDICATION

To my parents and their naive boy 5 years ago.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ACKNOWLEDGMENTS

First of all, I would like to thank Prof. Homayoun Yousefi'zadeh and Prof. Faryar Jabbari for their invaluable guidance and mentorship. In preparation for the Ph.D preliminary exam in 2013, Prof. Jabbari advised me to take Prof. Yousefi'zadeh's Linear Programming class in EECS department, which happened to ignite this dissertation. One year after the class ended, I got an email from Prof. Yousefi'zadeh, when I was in a grocery store with my mom, informing me that he could offer me an RA position related to a project funded by DARPA, and I could work on it toward the completion of my Ph.D degree. The project was fascinating and I started the journey without much of relevant background. Prof. Yousefi'zadeh pointed me to relevant papers to read and taught me how to read those papers. He shaped my scattered ideas and put me on a clear path formed by my sporadic thoughts. Prof. Jabbari helped me find TA positions after the DARPA project ended, guided me through administration issues, and offered insight on practical applications of theoretical research topics. It was hard at the beginning. Doing research at that time felt like a daunting effort in finding my way out of a swamp. Fortunately, their guidance allowed me to achieve my goal without drowning in the swamp.

Second, I'm grateful to the support and encouragement provided by my family. My father worked hard to support my first year study abroad and my mother traveled abroad a couple of times to relieve my homesickness. Three of us endeavored to build a better future in different ways. This is the spirit of my family - to always do whatever we can and hope for the best. We live in different sides of the Pacific Ocean but our hearts beat as one.

Third, I want to thank my fiancee, Ms. Jingshu Ma. Our story started at UCI. She altered the track of her life for our relationship. It was difficult seeking dreams of our own while keeping a relationship. We worked it out. What we've been through these years, is a priceless experience testing our endurance under server situations and then entertaining stories about it after a homemade dinner, when no one wants to wash the dishes.

Apart from my advisors and family, I also want to thank my friends, who helped me settle down in Irvine, provided assistance and important information when I needed them most, accompanied me when I was alone, as well as those who showed me different life styles.

Last but not least, I want to thank our sponsor, DARPA GRAPHS program Award N66001-14-1-4061, for supporting my research, as well as IEEE for giving me the permision to incorporate some published works into this dissertation.

CURRICULUM VITAE

Kai Ding

EDUCATION

Doctor of Philosophy in Mechanical and Aerospace Engineering **Mar. 2018**
University of California, Irvine *Irvine, CA*

Master of Science in Mechanical and Aerospace Engineering **Dec. 2014**
University of California, Irvine *Irvine, CA*

Bachelor of Engineering in Detection Guidance and Control Technology **Jun. 2012**
Beijing University of Aeronautics and Astronautics *Beijing*

RESEARCH EXPERIENCE

Graduate Student Researcher **2013–2017**
University of California, Irvine *Irvine, California*

Project: Network Connectivity Establishment and Maintenance

TEACHING EXPERIENCE

Teaching Assistant
ENGRMAE 80 Dynamics **Jan. 2017 - Aug. 2017**

ENGRMAE 150L Mechanics of Structures Laboratory **Sep. 2015 - Dec. 2015**

Peer Mentor in Graduate Resource Center **Summer 2014 & 2016**

REFEREED JOURNAL PUBLICATIONS

A Robust Advantaged Node Placement Strategy for Sparse Network Graphs **2017**
IEEE Transactions on Network Science and Engineering

REFEREED CONFERENCE PUBLICATIONS

A systematic node placement strategy for multi-tier heterogeneous network graphs **Apr. 2016**
IEEE 2016 Wireless Communications and Networking Conference (WCNC'2016)

ABSTRACT OF THE DISSERTATION

Connectivity Establishment and Maintenance for Sparse Network Graphs

By

Kai Ding

Doctor of Philosophy in Mechanical and Aerospace Engineering

University of California, Irvine, 2018

Professor Faryar Jabbari, Chair

Establishing robust connectivity in heterogeneous networks (HetNets) is an important yet challenging problem. For a heterogeneous network accommodating a large number of nodes, establishing perturbation-invulnerable connectivity is of utmost importance. This dissertation provides a robust advantaged node placement strategy best suited for sparse network graphs. In order to offer connectivity robustness, this work models the communication range of an advantaged node with a hexagon embedded within a circle representing the physical range of a node. Consequently, the proposed node placement method in this dissertation is based on a so-called hexagonal coordinate system (HCS) in which we develop an extended algebra. We formulate a class of geometric distance optimization problems aiming at establishing robust connectivity of a graph of multiple clusters of nodes. After showing that our formulated problem is NP-hard, we provide a heuristic algorithm based on HCS that efficiently solves the problem. Performance evaluation on different aspects of the algorithm is given. The results show that our algorithm is most effective in sparse networks for which we derive classification thresholds.

After connectivity is established, we consider the problem that when some nodes are exposed to mobility, how to maintain connectivity. In this part, the network of interest consists of two types of nodes, pre-deployed (client) and intermediate nodes. We assume full control

on the intermediate nodes but not the pre-deployed nodes. In such networks, on which we have only partial control, two types of node mobility scenarios are investigated. The first scenario analyzes the bounds of mobility when pre-deployed nodes move at small scales. The bounds of node mobility preserving connectivity are derived through analysis and verified by simulations. The second scenario considers the movement of pre-deployed nodes beyond the bounds of the first scenario thereby breaking connected links and partitioning the connected network. This scenario then considers relocating the existing intermediate nodes in order to reestablish connectivity. A general formulation is proposed in the form of an optimization problem. We prove that the general formulation of the problem is NP-hard. Next, we turn our attention to a practical scenario in which the location of nodes is made available using GPS signals. We solve the problem of the practical scenario in polynomial time and analyze the complexity of our solution. We also present comprehensive performance evaluation results of our proposed algorithm.

Chapter 1

Introduction

In this dissertation, we study the problem of network connectivity establishment and maintenance. The first part of the research aims at developing a systematic strategy of establishing robust network connectivity in the presence of perturbations. This task is fulfilled by placement of extra nodes in the network. We seek a node placement strategy that aid establish the initial connectivity and carries a certain level of robustness therein. The second part of the research studies the problem that when some of the nodes in the network are exposed to mobility, how to relocate the intermediate nodes placed in the first part, in order to maintain connectivity. We formulate the problem in the form of an optimization problem, which is later proved to be NP-hard, and seek solutions with reasonable time complexity. In the following sections, an overview of each problem is given, respectively, followed by relative literature review.

1.1 Overview of Connectivity Establishment in Heterogeneous Networks

Establishing connectivity in heterogeneous networks (HetNets) has been of high significance in the studies of MANET [1], wireless sensor networks [2], multi-facility locations [3], and traffic flow problems [4]. HetNets are typically composed of nodes with different capabilities and are formed by a collection of clusters. Generally, each cluster contains several standard nodes with short communication ranges and a cluster head node [5]. The cluster head node is an advantaged node serving as the gateway of this cluster in communication with other cluster heads. Connectivity scenarios of multi-tier networks have found extensive applications in different disciplines including but not limited to health surveillance, environment monitoring, earthquake detection, and Internet of Things (IoT). In all these applications, a large number of low-capability standard nodes (SNs) rely on a small number of advantaged nodes (ANs) to form a communication network.

Similar to literature work of [6, 7], we assume HetNets are formed by SNs arranged in clusters with each cluster designated an AN gateway. AN gateways are assumed to have much longer communication ranges and able to simultaneously connect to multiple nodes [8]. While the assumption guarantees intra-cluster connectivity inter-cluster connectivity still needs to be established by placement of additional intermediate ANs. Lin and Xue [9] abstract this problem in the form of a *Steiner minimum tree problem with minimum number of Steiner points and bounded edge length*, which we refer to this algorithm as SMT not to be confused with MST used to represent minimum spanning trees. Lin and Xue provide an approximation algorithm to the original NP-complete problem with a polynomial time complexity and a performance ratio of 5. This algorithm lays the groundwork of several other approximation algorithms with smaller (better) performance ratios [10, 11, 12, 13]. In [8], the authors develop a node placement algorithm for clustered ad-hoc networks subject to capacity

constraints. Other related works, albeit at small scale sensor networks, include [14, 4, 7, 15] in which energy and network lifetime constraints are emphasized in node placement.

All of the above algorithms use the Gilbert disk connectivity model [16, 17, 18] representing the communication range of an AN as a circle. One disadvantage of this model is lack of boundary connectivity robustness where the distance between two centers is close to the distance threshold of connectivity d. In such cases, a pair of connected nodes can easily become disconnected as the result of small position perturbations, a phenomenon occurring frequently and unpredictably, especially in harsh environments. To compensate against these cases, fault-tolerant k-connectivity ($k \geq 2$) node placement algorithms have been developed [19, 14, 20, 21, 22]. By using a much larger number of ANs, these algorithms guarantee there are always k different paths between each pair of ANs.

In addition to the disadvantage above, SMT-based methods are subject to a second yet major disadvantage. Since the minimum spanning tree is formed statically once, to represent the topology of the network graph, SMT-based methods do not consider the effects of changes to minimum spanning tree as the result of placing ANs in subsequent iterations. This can lead to potentially over utilizing AN resources, since it is possible to establish connectivity with a smaller number of ANs.

As detailed in Section 2.2 and Section 3.3, this work provides a dynamic strategy for AN placement capable of dynamically considering the effects of changes to minimum spanning tree while offering robust network connectivity in the presence of perturbations. In essence, we seek an AN placement strategy that carries a certain level of robustness therein. To avoid the inherent problem of Gilbert disk model in boundary connectivity cases, we model the communication ranges of nodes as hexagons embedded within the circles representing the actual communication ranges of nodes. Two nodes are considered connected only when their associated hexagons have a common edge. Consequently, a pair of connected nodes actually have a margin of perturbation conserving connectivity. Projecting the node placement prob-

lem into HCS with integer coordinates allows us to utilize the higher computation efficiency of HCS compared to a conventional Cartesian Coordinate System (CCS) in minimizing the number of intermediate ANs, identifying their positions, and accounting for topology perturbations.

In our work, we consider a two-tier graph of nodes in which clusters of SNs are to be connected with a minimum number of ANs. ANs are distinguished from SNs by their higher ranges of communication and ability to simultaneously connect to a large number of standard nodes. Each cluster of SNs is assumed to be equipped with an AN allowing full connectivity of the nodes within the cluster. Multiple clusters of SNs may or may not be connected depending on their separation distance. It is important to note that inter-cluster connectivity as facilitated by ANs is mostly a function of distance as opposed to interference because of the much larger separation distances of ANs and much stronger power profiles compared to SNs.

In this dissertation, connectivity establishment problem is studied in Chapter 2, Chapter 3 and Chapter 4. The main contributions of this part of work are as follows. First and for the purpose of offering robustness, we introduce a hexagonal coordinate system and develop associated extended algebra. Relying on the proposed HCS, we then formulate a class of geometric distance optimization problems aiming at finding the minimum number of ANs and their positions to guarantee robust connectivity of a given HetNet. We prove that our formulated problem is NP-hard and offer an exhaustive search algorithm for solving this NP-hard problem as well as a low complexity algorithm for solving an approximation of this problem. We show our heuristic solution closely tracks the exhaustive search algorithm while enjoying excellent node cost, runtime, and robustness characteristics compared to other alternatives. Our proposed approximation algorithm utilizes far fewer ANs than a k-connected network. This is because establishing a k-connected network requires many additional edges to a graph so as to preserve connectivity under $(k-1)$ edge or vertex cuts. Naturally, adding edges will increase the number of intermediate ANs.

1.2 Overview of Mobile Network Connectivity Maintenance

With the emergence of high speed networks, the concept of "connecting everything" has drawn major attention in the recent years [23]. Accordingly, a vast array of stationary and mobile devices have become members of large connected components. In the case of mobile devices, connectivity is negatively affected by wireless network environment effects such as fading [24], interference [25], and congestion [26]. Consequently, the challenge of "keeping everything connected" is introduced after initially succeeding in "connecting everything". The problem space of this connectivity maintenance falls under the paradigm of keeping everything connected in presence of mobility. The subject of keeping everything connected is well studied in the literature of mobile ad hoc networks (MANET) [27, 1] and multi-agent systems [28, 29].

One way of persisting connectivity is to preserve the quality of wireless channels in dynamic networks [2, 30]. Accordingly, several MANET protocols have been developed [31, 32, 33] to take advantage of preserving link qualities. Particularly useful in long range networks mostly with line of sight links, another way of persisting connectivity is through topology control. In such cases, the quality of communication channels largely depends on node separation distances. In essence, a link between a pair of nodes is maintained if the distance between the pair does not exceed a threshold denoted by $2R$. The works of [34, 35, 36, 37, 38] propose decentralized control laws to prevent a network from getting disconnected. In those works, maintaining connectivity typically relies on strategies of preserving a spanning tree of the network graph.

In this dissertation, after establishing connectivity among a number of isolated clusters, we consider the problem of maintaining connectivity under node mobility. The mobile network also consists of two type of nodes, pre-deployed and intermediate nodes, and mobility of nodes

will start with the connected network established through our proposed algorithm. In our scenario of operation, similar to the connectivity establishment problem, each pre-deployed node is the advantaged gateway node for a cluster of standard nodes whose transmission ranges are limited. Referred to as members of a set V_0, these pre-deployed nodes and their associated cluster units may become mobile. Intermediate nodes, referred to as members of another set V_1, are additional advantaged nodes placed in the network so as to facilitate connectivity.

In [39] and [40], we proposed a class of node placement algorithms to robustly establish connectivity using a minimum number of intermediate nodes. Here, the phrase 'robust' indicates that connectivity is preserved in presence of small perturbations of node locations. In order to gain coverage overlap and thereby helping robustness, a hexagon rather than a disk (circle) is used to represent the communication range of individual nodes. Relying on a hexagonal coordinate system (HCS) within which an extended algebra is developed, a class of geometric distance optimization (GDO) node placement algorithms are introduced. Numerical results show the robustness advantages of GDO algorithms in sparse networks. However, connectivity is established among pre-deployed static nodes in set V_0 without the considerations of mobility.

In the second part of our research, we study how mobility of pre-deployed nodes impacts connectivity and develop strategies of relocating inetrmediate nodes in order to maintain a connected network. We will investigate this problem by splitting it into two subproblems. First, since establishing connectivity utilizing the hexagonal model of [39] offers a built-in robustness, the network may remain connected if the mobility of elements of V_0 are at small scales. The bounds of small scale mobility beyond which connectivity is not preserved are identified. Second, when the mobility of elements of V_0 exceed small scale mobility bounds, the underlying network would, with a high probability, become disconnected. In this case, a strategy of relocating the element of V_1 is pursued in order to restore connectivity. Developing

such strategy is of interest albeit the fact that one can always run EGDO algorithm to establish a connected network. The justification of our alternative approach is two folds. First, applying EGDO algorithm may require adding extra intermediate nodes which my not be practical in the middle of network operations. Second, the runtime of EGDO algorithm is anticipated to be relatively high even without requiring to add extra nodes. Hence, we focus on developing a lower complexity algorithm to restore connectivity if the circumstances permit. Otherwise, EGDO algorithm may still be used to restore connectivity.

In this dissertation, the network maintenance problem is investigated in Chapter 5 and Chapter 6. Our main contributions in this part are the followings. First, we analyze the small scale mobility bounds of the elements of V_0 within which the network remains connected. Second and given the same number of intermediate nodes used to establish the connectivity of the static network, an optimization problem is proposed to restore general connectivity when the mobility of elements of V_0 goes beyond the bound. The problem formulation manifests in the form of semidefinite programming applied to an algebraic graph and is proven to be NP-hard. Third, we consider a practical scenario in which node positions are reported through GPS signaling. We inspect the solution of the second problem in two steps. First, we answer a feasibility question as to whether the network can be reconnected using the same number of intermediate nodes. If the answer is yes, we develop an algorithm with polynomial time complexity to relocate the intermediate nodes.

1.3 Literature Review

1.3.1 Node Placement Strategies

In this section, related node placement strategies are reviewed. We go over a number of published approaches highlighting their desired performance goals and according solutions.

It should be highlighted that when it comes to "node placement" in literature, there are different types of node to be placed, such as sensor, relay, cluster-head, base-station, etc. In our work, we consider the so-called 'advantage node', which is prone to representing cluster-head in practice [41].

When establishing a connected network via advantage node placement, certain performance goals, such as minimizing the number of ANs used, maximizing area coverage, improving energy efficiency and gain the ability of fault tolerance, are usually desired. The authors in [42] studied the problem of covering an area of interest with the least number of SNs so as to maximize the probability of detecting targets. In [43], the authors investigated the problem of maximal coverage with the lowest sensor cost in underwater wireless sensor networks. These sensors are deployed on the bottom of the ocean and served by nearby gateway nodes which collect data from them. In [44], the author seek a maximal coverage strategy with the least node cost that maintains a robust connected network geometry even though one node fails. This requires the network to be 2-connected. These works all aim at maximal coverage area and the minimal nodes cost, while have secondary constraints on different aspects.

In optimization of energy efficiency, one aspect is to have relatively shorter path on average by controlling network topology because the energy consumption rate will be drastically increased if the communications between two nodes are through long distance links. Another aspect with the same objective is to optimize power control and load allocation. In massively dense sensor networks, it is unrealistic to model the network with node/edge notation. The authors in [45], studied the node deployment problem in terms of node density. They attempt to find a distribution function for sensor nodes so that all data flows over short paths to the data collector. In [4, 46], the authors optimize the traffic allocation by AN placement in order to gain high energy efficiency and consequently enhance average network life time. The authors in [6] reported a AN deployment strategy to balance non-uniformly distributed energy-consuming rate caused by deficient network geometry. In [47], the author investigated

the relation between node density of a network and its lifetime.

When the objective is to form a fault-tolerate network, researchers focus on achieving K-connectivity while establishing a connected network. A network is said to be K-connected if and only if there are K edge-independent paths between any two nodes. This problem is usually posed as an optimization problem with K-connected as constraint and the number of intermediate AN placed as objective function. Reference [48, 22, 20] are in this category.

In works aiming at minimizing the number of intermediate AN placed, both single and multi-tier networks are studied. In single-tier networks [12, 10], all nodes has the same communication capability. In multi-tier networks [49], R, referred to as the commutation radius of an AN, is strictly and typically one or two orders larger than that of a SN, denoted as r. In [7, 15] the authors mainly aim at minimizing network cost constrained by a certain length of network life, when establishing connectivity by AN placement. Their strategies are primarily applied to prefixed sensing spots within applications operating in non-harsh environments.

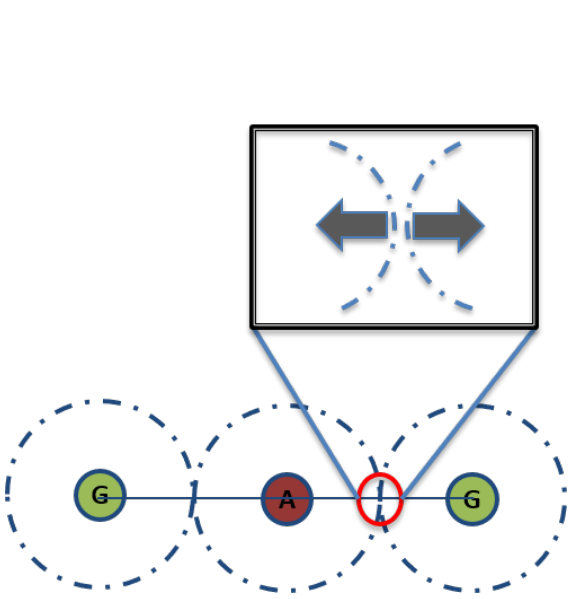
The classes of node placement problems mentioned above are typically proved to be NP-hard. There usually exist no analytical optimal solution, which left a wide space for researchers to fill with heuristic algorithms that efficiently solve the problem under practical assumptions.

In this dissertation, we mainly focus on the objective of establishing connectivity with the least number of intermediate ANs. Lin and Xue [9] prove this problem is NP-hard and provide a static algorithm to solve it. In their algorithm, a minimum spanning tree (MST) of sensor nodes is formed at the beginning. The formation of such MST may introduce some edges that are longer than the communication range of a standard sensor node hence requiring to rely on intermediate ANs, whose communication radius is R, to establish connectivity. Their method calls for dividing each edge equally by R in order to identify the positions of additional ANs. The authors claim their work has a performance ratio of 5, but Chen et al.

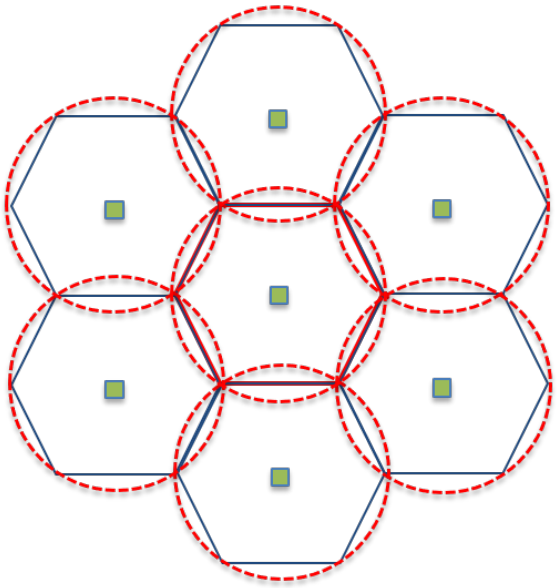
prove later in [11] that this algorithm is actually no more than 4 times the optimal solution (4-approximation), and provided a 3-approximation algorithm. Here, an algorithm is said to be α-approximation as long as it gives a solution no more than α times the optimal solution of the problem. An extension of Lin and Xue's work to homogeneous network where ANs and SNs share the same communication range and could both serve as a sensor or a relay is presented in [12] by X. Cheng. They considered a class of WSN, such as biomedical sensor networks where sensors are determined prior to deployment. To maximize network life time and reduce interference, it desired to maintain network connectivity with minimum number of ANs and minimum transmission power per node. To further extend the MST algorithm to the case of HetNets, Lloyd and Xue [10] consider a case in which R is larger than or equal to r. They first consider the problem in a way that either an SN or an AN could serve as a data forwarding and receiving node. This algorithm generates a path between two sensor nodes consisting of both SNs and ANs. Then, the authors propose a $(5+\epsilon)$ -approximation algorithm for establishing a connected path composed of only ANs between each pair of SNs. Slightly before Lloyd's work got published, Tang [14] proposes an exhaustive-search algorithm for HetNet graphs in which R is set to be 4 times larger than r. The proposed algorithm has an approximation ratio of 4.5.

Being a static algorithm, MST-based methods do not consider the potential of connectivity augmentation offered by ANs placed at later stages and those placed previously. This can lead to potentially over utilizing AN resources, since it is possible to establish connectivity with a smaller number of ANs. In addition, MST-based methods represent the communication range of a node by a disk, i.e., geometric disk model which can potentially prove problematic. As shown in Fig. 1.1a, when utilizing the geometric disk model, a pair of touching circles representing connected nodes may be easily disconnected as the result of small perturbations affecting the geometry of the system. In order to establish a robust network, we introduce, for the first time, hexagon model to represent the communication range. Two hexagons having one edge in touch are considered as connected. As shown in Fig. 1.1b, two connected

nodes share an overlapping area in the real world, which functions as the safety margin that is capable of compensating local perturbations. Further investigation of this model will be given in Chapter 2.



(a) Disk model incurs vulnerable connection.



(b) Hexagon model offers safety margin.

1.3.2 Connectivity Maintenance Strategies

In literature, there are two other often-seen concepts that are equivalent to connectivity maintenance, they are connectivity preservation and controlling connectivity. This topic is widely studied in the area of multi-agent systems, mobile networks and wireless sensor networks. There are three major strategies of connectivity maintenance. The first is to give a control protocol to each node (agent) so as to preserve its connectivity to its neighbors. This is usually done by impose a potential-field-like controller to each node [50, 51, 31]. The second is by establishing an initially k-connected network graph [52, 53, 54], which will remain connected even though $(k-1)$ edges in the graph were lost due to node mobility. The third, and usually the most difficult, is to design a network reconfiguration strategy to accommodate the node mobility [55, 56].

Spanos and Murry proposed a quantity called the *geometric connectivity robustness* [57] that can be employed as motion constraint for mobile nodes. Papas and Zavlanos [58] reported a centralized control law that preserves k-hops connectivity. That is to say, each node, while moving, keeps its k-hop neighbors unchanged. As a special case, when $k = 1$ it preserves the connectivity of each edge. Later on, they proposed the idea of maintaining connectivity by a controller based on potential field in [50], which laid the groundwork of several later research on decentralized control algorithms that maintain connectivity [59, 60, 35]. This potential field calls for exerting negative force to the motion of a node, when the node is moving towards a direction that reduces the global connectivity, characterized by the algebraic connectivity. In [61], the authors consider a practical application of the connectivity maintenance problem. The strategy of maintaining connectivity in their work is by defining an edge tension function, which is similar to the potential field. A big issue of applying potential field based algorithms in multi-agent systems is that the calculation of the algebraic connectivity requires the knowledge of Laplacian matrix of the graph. That is to say, we have to get the information of the entire topology at each moment so as to calculate the potential field, implying the algorithm we developed has to be centralized.

To relieve this issue, scholars come up with strategies that maintains local connectivity that run on each individual agent [62, 63, 64]. In [38], the authors proposed a decentralized estimate method for the algebraic connectivity and proved their designed decentralized algorithm is able to maintain connectivity. [31] provides a scheme of sensor deployment and local connectivity maintenance. Beyond preserving connectivity, this decentralized algorithm also considers classic sensor network deployment problem such as maximizing the coverage area, covering point of interest and barrier coverage.

When dealing with connectivity maintenance problem in the scenario of edge of node failure, establishing a k-connectivity at the very beginning may not always be a practical and economic method. So the problem of adaptively changing the topology by controlling node

movement is investigated. [65] presents a decentralized scheme for connectivity maintenance under the condition of possible node failure, while [66] provides a decentralized algorithm for the scenario of possible edge failure.

Due to the importance of mobile robotic networks or multi-agent networks, more and more scholars put attention on maintaining connectivity through reconfiguration of network topology. [67] studies whether a node in a network is able to move to a desired location while preserving the network connectivity. The problem is formulated to a convex optimization problem for which the author provided a centralized solution. In [68], the problem of maximizing the lifetime of a sensor network, by adding and relocating a set of relay nodes in the deployed network, is investigated. The author of [69] researches on a more practical application problem of the same nature as connectivity maintenance. The two problems addressed in this paper are maximizing the consensus rate and minimizing the communication cost by relocating the nodes in a wireless network. The fundamental assumption is a connected network. Both [68] and [69] formulated the problem in a Semi-definite Programming (SDP). [53] provide a broad and in-depth survey on connectivity maintenance problem utilizing the algebraic connectivity concept, which better explained why a lot of the problems in this area falls into the category of SDP or Mixed-integer SDP (MISDP).

The problem we will study in Chapter 5 and 6 is connectivity maintenance via topology reconfiguration, and with little doubt, is a MISDP.

Chapter 2

Connectivity Model and Hexagon Coordinate System

In this section, a connectivity model as well as its variant model are introduced. To facilitate the application of this model, a new coordinate system is define with corresponding algebras.

2.1 Connectivity Model

Based on the landmark Gilbert connectivity model [16], early connectivity models in network graphs mainly consider the distance between nodes. Later, a number of more realistic models [24, 70, 71] were established to capture connectivity using propagation, fading, shadowing, signal-to-interference-noise ratio (SINR), symbol error rate (SER), and capacity. A review of these recent works reveals that using a distance-based connectivity model is justified when high power long range communication dominates other factors such as interference, fading, and shadowing. Accordingly, this work assumes that ANs are characterized by longer communication ranges, higher powers, and higher lifetimes compared to SNs.

A pair of nodes $\{M,\ N\}$ are considered to be bi-directionally connected if both M and N located within each other's communication range . In the definition above, the distance between nodes is a realistic measure of connectivity because inter-cluster communication relies on LOS links established betweem high power ANs. For a pair of SN and AN nodes with ranges r and d in radii, connectivity is established only when the distance between two nodes is less than or equal to $\min\{r, d\} = r$.

In our model, a number of SNs form a connected cluster for which the center of geometry can be calculated. A number of these clusters in a given area compose a network topology scenario. The location of clusters could be random or follow some certain distribution rule depending on the SN deployment preference. The 3 red dots in Fig. 2.1 represent 3 SNs with communication ranges of r forming a sample cluster of SNs. Each cluster is assumed to be supported by an AN gateway node. This AN is typically located at the center of geometry of the cluster in order to maximize the number of SNs to which it is directly connected. Alternatively, AN gateways may have a small displacement from the center of geometry. Nonetheless, SNs within a cluster are all connected to the AN gateway node and able to communicate with nodes outside of the cluster through the AN gateway node. Thus, the problem of global connectivity is converted to connecting individual clusters utilizing additional intermediate AN nodes as necessary. Based on the connectivity condition given above, one AN ought to locate within the communication range of another AN so as to establish connectivity. A pair of ANs with communication ranges of $d = 2R$ are connected if the two circles with radii R and centered around them overlap.

In order to provide a margin of robustness in presence of location perturbation, we model the communication area of an AN by a hexagon with an edge length of R. Considering the extended range of AN compared to SN, we assume R is approximately two orders of magnitude larger than r. Without loss of generality, the communication area of an AN can be set as a hexagon with an edge length of $(12n + 7)r$ where n is a positive integer chosen

such that the expression accurately approximates the value of R. The length selection of $(12n + 7)r$ offers a couple of geometrical advantages. First, any vertex of a large hexagon overlaps with the vertex of a hexagonal cell at the same relative position. Second, center to edge distance of a hexagon is conveniently measurable by the distance measure defined in the next section. This distance relates to the minimum distance coverage by an AN and will be utilized in Section 3.3. Two ANs are then robustly connected if their associated hexagons have a common edge.

In our model, a number of SNs form a connected cluster for which the center of geometry can be calculated. A number of these clusters in a given area compose a network topology scenario. The location of clusters could be random or follow some certain distribution rule depending on the SN deployment preference. The 3 red dots in Fig. 2.1 represent 3 SNs with communication ranges of r forming a sample cluster of SNs. Each cluster is assumed to be supported by an AN gateway node. This AN is typically located at the center of geometry of the cluster in order to maximize the number of SNs to which it is directly connected. Alternatively, AN gateways may have a small displacement from the center of geometry. Nonetheless, SNs within a cluster are all connected to the AN gateway node and able to communicate with nodes outside of the cluster through the AN gateway node. Thus, the problem of global connectivity is converted to connecting individual clusters utilizing additional intermediate AN nodes as necessary. Based on the connectivity condition given above, one AN ought to locate within the communication range of another AN so as to establish connectivity. A pair of ANs with communication ranges of $d = 2R$ are connected if the two circles with radii R and centered around them overlap.

In order to provide a margin of robustness in presence of location perturbation, we model the communication area of an AN by a hexagon with an edge length of R. Considering the extended range of AN compared to SN, we assume R is approximately two orders of magnitude larger than r. Without loss of generality, the communication area of an AN can

be set as a hexagon with an edge length of $(12n + 7)r$ where n is a positive integer chosen such that the expression accurately approximates the value of R. The length selection of $(12n + 7)r$ offers a couple of geometrical advantages. First, any vertex of a large hexagon overlaps with the vertex of a hexagonal cell at the same relative position. Second, center to edge distance of a hexagon is conveniently measurable by the distance measure defined in the next section. This distance relates to the minimum distance coverage by an AN and will be utilized in Section 3.3. Two ANs are then robustly connected if their associated hexagons have a common edge.

2.2 Hexagonal Coordinate System

First, the node placement problem is projected into a so-called hexagonal coordinate system. To set up the HCS, we have to specify the origin, axes, and coordinates. The origin is defined as the center of geometry of all clusters. From this origin, we start tiling the plane with hexagonal cells. These cells have an edge length equal to the communication radius of an SN, r. The first cell share the same center of geometry as the origin point with coordinates $(0, 0)$. Then, we establish the rest of the tessellation with equal-sized hexagonal cells. Theoretically, an infinite tessellation can tile an infinite-extending plane without either overlapping or gaps. In practice, we stop when the area of interest is fully tiled. The x-axis goes through the origin and is perpendicular to a pair of parallel edges of the cell containing the origin. The x-axis cuts through all hexagonal cells along that direction through their center and edge. The y-axis is defined as the rotation of the x-axis by $\pi/3$ counter-clockwise, as shown in Fig. 2.1. The y-axis also crosses the origin and vertically cuts across the edges of all cells along the way including origin. In this coordinate system, coordinates are associated with those of a hexagonal cell unlike other coordinate systems [72, 73] such as that in reference [74] in which the $x-$axis goes through the center and a cell vertex. Points A and B in Fig. 2.1

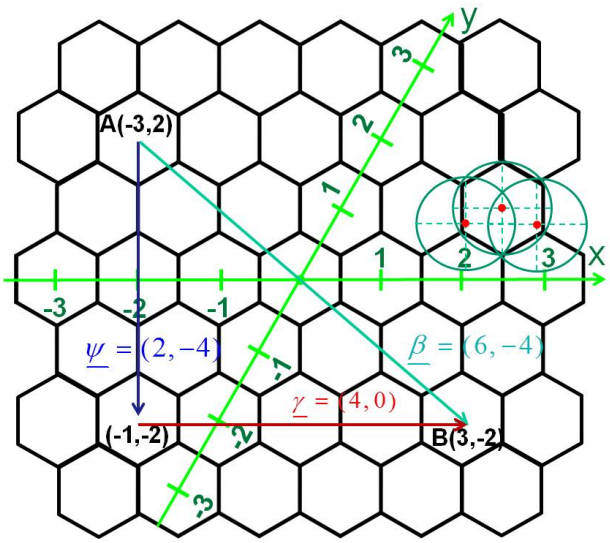


Figure 2.1: The hexagonal coordinate system (HCS).

illustrate a pair of coordinate examples.

2.2.1 Operation Definitions in HCS

Distance Measure

Since a point in an HCS actually represents the location of a hexagonal cell, a distance measure between two hexagonal cells aims at counting the number of cells moving from one cell to another. The distance between point A and B in Fig. 2.1 serves as a typical example. For a given pair of points $M(m_1, m_2)$ and $N(n_1, n_2)$, the distance measure for the vector $\underline{\vartheta} = (M, N)$ is defined as follows.

$$\begin{aligned} |\underline{\vartheta}| &= |(M,N)| = (m_1 - n_1, m_2 - n_2) \\ &= \mathbf{max}\{|m_1 - n_1|, |m_2 - n_2|, |m_1 - n_1 + m_2 - n_2|\} \end{aligned} \tag{2.1}$$

For example, $\underline{\beta} = (A, B)$ is a vector starting at the center of cell A and ending at the center of cell B in Fig. 2.1. The distance between A and B is 6 representing the shortest path from

A to B covers 6 cells.

$$\underline{\beta} = (A, B) = (6, -4), \quad |\underline{\beta}| = |(A, B)| = 6$$

Theorem 2.1. *The distance measure defined by (2.1) is a distance.*

Proof. Noticing that a distance in HCS calculated through (2.1) is non-negative, it is left to prove the triangular inequity:

$$\forall L, M, N, |(L, M)| + |(M, N)| \geq |(L, N)|$$

We have the following three cases to consider. *Case 1* RHS $= |l_1 - n_1|$. In this case, we have

$$\begin{aligned}
\text{LHS} &= \mathbf{max}\{|l_1 - m_1|, |l_2 - m_2|, |l_1 - m_1 + l_2 - m_2|\} \\
&\quad + \mathbf{max}\{|m_1 - n_1|, |m_2 - n_2|, |m_1 - n_1 + m_2 - n_2|\} \\
&\geq |l_1 - m_1| + |m_1 - n_1| \\
&\geq |l_1 - n_1| = \text{RHS}
\end{aligned}$$

Case 2 RHS $= |l_2 - n_2|$. In this case, we have

$$\begin{aligned}
\text{LHS} &= \mathbf{max}\{|l_1 - m_1|, |l_2 - m_2|, |l_1 - m_1 + l_2 - m_2|\} \\
&\quad + \mathbf{max}\{|m_1 - n_1|, |m_2 - n_2|, |m_1 - n_1 + m_2 - n_2|\} \\
&\geq |l_2 - m_2| + |m_2 - n_2| \\
&\geq |l_2 - n_2| = \text{RHS}
\end{aligned}$$

Case 3 RHS $= |l_1 - n_1 + l_2 - n_2|$. In this case, we have

$$\begin{aligned}
\text{LHS} &= \mathbf{max}\{|l_1 - m_1|, |l_2 - m_2|, |l_1 - m_1 + l_2 - m_2|\} \\
&\quad + \mathbf{max}\{|m_1 - n_1|, |m_2 - n_2|, |m_1 - n_1 + m_2 - n_2|\} \\
&\geq |l_1 - m_1 + l_2 - m_2| + |m_1 - n_1 + m_2 - n_2| \\
&\geq |(l_1 - m_1 + l_2 - m_2) + (m_1 - n_1 + m_2 - n_2)| \\
&= |l_1 - n_1 + l_2 - n_2| = \text{RHS}
\end{aligned}$$

□

The vector addition rule in HCS follows that of Cartesian coordinate system. For vector $\underline{\psi} = (\psi_1, \psi_2)$ and $\underline{\gamma} = (\gamma_1, \gamma_2)$, we have $\underline{\psi} + \underline{\gamma} = (\psi_1 + \gamma_1, \psi_2 + \gamma_2)$.

Inner Product

The definition of inner production in an HCS is not the same as that of Cartesian coordinate system since the two basis vectors are not perpendicular to each other. Let's call $\underline{e}_1$ and $\underline{e}_2$ the two basis vectors in the HCS along x- and y-axis, respectively. We define the inner product as follows.

$$\underline{x} \cdot \underline{y} = \underline{x}^T \Delta \underline{y} \tag{2.2}$$

where $\underline{x} = x_1\underline{e}_1 + x_2\underline{e}_2$, $\underline{y} = y_1\underline{e}_1 + y_2\underline{e}_2$, T represents the transpose operator, and Δ is a symmetric matrix defined below.

$$\Delta = \begin{bmatrix} 1 & \cos\frac{\pi}{3} \\ \cos\frac{\pi}{3} & 1 \end{bmatrix} \tag{2.3}$$

For example, in Fig. 2.1, the inner product between $\underline{\gamma}$ and $\underline{\beta}$, $\underline{\gamma}$ and $\underline{\psi}$ are calculated as below.

$$\underline{\gamma} \cdot \underline{\beta} = \begin{bmatrix} 4 & 0 \end{bmatrix} \begin{bmatrix} 1 & \cos\frac{\pi}{3} \\ \cos\frac{\pi}{3} & 1 \end{bmatrix} \begin{bmatrix} 6 \\ 4 \end{bmatrix} = 32$$

$$\underline{\gamma} \cdot \underline{\psi} = \begin{bmatrix} 4 & 0 \end{bmatrix} \begin{bmatrix} 1 & \cos\frac{\pi}{3} \\ \cos\frac{\pi}{3} & 1 \end{bmatrix} \begin{bmatrix} 2 \\ -4 \end{bmatrix} = 0$$

It is observed that the inner product of a pair of vectors is zero if they are perpendicular to each other, as shown by vector $\underline{\gamma}$ and $\underline{\psi}$ in Fig. 2.1. Further, the inner product operation is commutative as Δ is a symmetric matrix.

2.2.2 Orientation of Distance Vector

When dealing with the least number of ANs required to link two clusters, one has to realize that the maximum covered distance of an AN has its own orientation. When the distance vector between two clusters is closely aligned with x- or y-axis, one has to possibly use more ANs than a case in which the distance vector is oriented at the direction $\pi/6$ away from each axis. In the latter case, one uses the length of diagonal to divide the distance and decide how many ANs are needed. Since we are mainly concerned about whether the distance vector is more aligned with x-, y-axis, or with the diagonals of the head and tail cluster, we take the basis axis as the reference point of the orientation. In the following subsections, we discuss a number of cases in which $\underline{\vartheta}$ is in different quadrants. We specify the quadrant in which $\underline{\vartheta}$ is located by inspecting a vector parallel to and of the same length as $\underline{\vartheta}$ with a starting point at the origin.

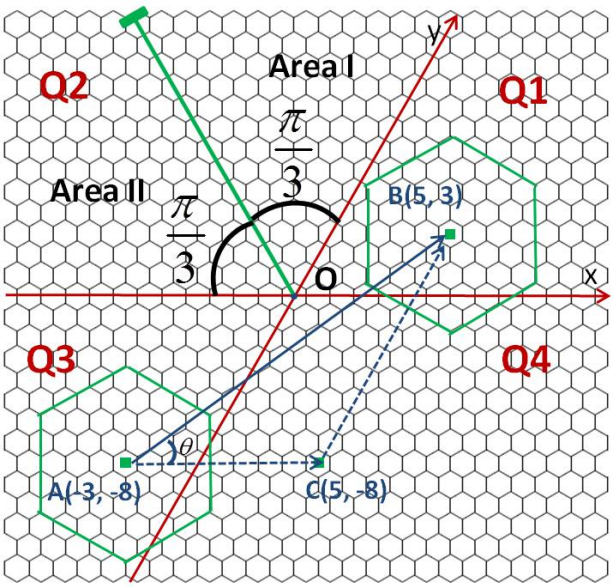


Figure 2.2: Calculating angle θ between ϑ and $x^{(+)}$-axis when ϑ is in the 1st or 3rd quadrant. In the other two quadrants, the quadrant is divided into two areas and θ is calculated with respect to $y^{(+)}$-axis in Area I or $x^{(-)}$-axis in Area II.

$\underline{\vartheta}$ is in the 1st or 3rd Quadrant

As shown in Fig. 2.2, the distance vector between cluster A and B is (A, B). The orientation of (A, B) is represented by θ which is between (A, B) and x-axis. In the triangle formed by A, B, and C, we can identify the value of θ from the Law of Sines, with $x = |(A, C)|$ and $y = |(C, B))|$, as $\frac{1}{y}\sin\theta = \frac{1}{x}\sin(\frac{\pi}{3} - \theta)$.

Hence,

$$0 < \theta = \tan^{-1}\left(\frac{\sqrt{3}\,y}{2x + y}\right) < \frac{\pi}{3} \tag{2.4}$$

$\underline{\vartheta}$ is in the 2nd or 4th Quadrant

In HCS, quadrants 2 and 4 are larger in area than quadrants 1 and 3. The angle between $y^{(+)}$-axis and $x^{(-)}$-axis is $2\pi/3$ which exceeds $\pi/2$. Whether we take $y^{(+)}$- or $x^{(-)}$-axis as the reference, the method of the former subsection leads to a point of discontinuity in Eq. (2.4).

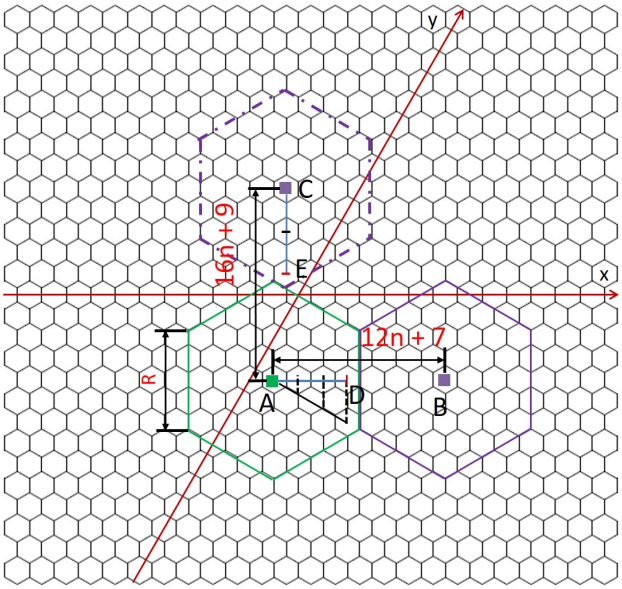


Figure 2.3: The longest possible distance covered by an AN.

Therefore, we partition each quadrant into two areas, as shown in quadrant 2 of Fig. 2.2. In Area I, the orientation of distance vector is referred to as $y^{(+)}$-axis, while in Area II, it is referred to as $x^{(-)}$-axis. Then, we can extract the associated equations from the Law of Sines separately. In Area I with $|x| \leq |y|$, we have $\frac{1}{-x}\sin\theta = \frac{1}{y}\sin(\frac{2\pi}{3} - \theta)$. Therefore,

$$0 < \theta = \tan^{-1}\left(\frac{-\sqrt{3}\,y}{2y + x}\right) < \frac{\pi}{3} \tag{2.5}$$

In Area II, $|x| \geq |y|$, we have $\frac{1}{-y}\sin\theta = \frac{1}{x}\sin(\frac{2\pi}{3} - \theta)$. Then,

$$0 < \theta = \tan^{-1}\left(\frac{-\sqrt{3}\,y}{2x + y}\right) < \frac{\pi}{3} \tag{2.6}$$

With distance orientation information, we are able to calculate the least number of intermediate ANs required to connect two clusters. That is to get the longest covering range of one AN along the direction of the distance vector and then divide the distance by the range. The following lemma gives the possible longest distance covered by an AN with a communication range $R = (12n + 7)r$.

Lemma 2.1. *The longest possible distance covered by an AN is an odd integer between* $(12n+7)$ *and* $(16n+9)$ *corresponding to direction relative to its neighboring AN.*

Proof. Fig. 2.3 shows two extreme cases. Segment $|(A,B)|$ is the shortest possible distance covered by adding one AN, B, that is connected to A. The direction of vector (A,B) is perpendicular to the common edge. On the other hand, segment $|(A,C)|$ is the longest possible distance covered by adding one AN. Here, we calculate them separately. Recalling that the edge length of a large hexagon in Fig. 2.3 is R, that of a small hexagon cell is r, and $R=(12n+7)r$, we have

$$|(A,B)| = 2\times|(A,D)|+1 = 2\times\frac{R-1}{2r}+1 = 12n+7$$

$$|(A,C)| = 2\times|(C,E)|+1 = 2\times(2\times\frac{R-1}{3r})+1 = 16n+9$$

As long as two ANs are connected and have one common edge, the distance between them in HCS is larger than the minimum case $|(A,B)|$ and less than the maximum case $|(A,C)|$. Last but not least, if two ANs have one edge in common, the distance between them is always an odd number. □

Chapter 3

Advantaged Node Placement Algorithm

In this chapter, we first prove the problem of connectivity establishment via advantaged node placement is a NP-hard problem. The optimal solution of this type of problem could be obtained through exhaustive search algorithm. We will provide such an algorithm. However, owing to its tremendous time complexity, it is hard to implement in physical network system that carries limited battery source. Therefore, we develop corresponding heuristic algorithm, known as Geometric Distance Optimization(GDO) algorithms, to solve it. Further more, in order to increase the efficiency and reduce the time complexity, we improve the GDO algorithm and give the Enhanced GDO (EGDO) algorithm. The corresponding simulation results reflecting the comparison of time complexity will be given in the next chapter.

3.1 NP-hard Problem Statement and Exhaustive Search Algorithm

In this section, we prove that our node placement problem in HCS is NP-hard by showing that it is a reduction from Knapsack problem which is known to be NP-complete. Then, we provide an exhaustive search algorithm to solve the problem as a comparison benchmark.

3.1.1 NP-Hard Problem Statement

Problem 1 (Knapsack Problem [75]) Given a set of items $E = \{e_1, \cdots, e_t\}$ each with a weight w_i and a profit v_i where $i \in \{1, \cdots, t\}$, is there a way of choosing x_i units of each item e_i to fill the knapsack such that the profit of the items chosen $\sum_{i=1}^{t} x_i v_i$ is at least V while the total weight of the items chosen $\sum_{i=1}^{t} x_i w_i$ is not exceeding W?

Problem 2 (Node placement problem in HCS) Given G pre-deployed gateway nodes with integer coordinates in an HCS and a minimum spanning tree of length L formed by these nodes, can one cover the total distance of L by N additional intermediate nodes?

In Problem 2, covering length L with $(N + G)$ ANs is equivalent to being able to find a connected path between any arbitrary pair of nodes where every pair of neighboring nodes have distances in the range $[(12n + 7)r, (16n + 9)r]$.

Theorem 3.1. *There is a polynomial time reduction from Problem 1 to Problem 2.*

Proof. Suppose set D has cardinality μ and contains all odd integers between $(12n + 7)$ and $(16n + 9)$, i.e., $D = \{12n + 7, 12n + 9, \ldots, 16n + 9\}$. We start with an instance I of Problem 1 with which set E with cardinality μ is associated. Then, we construct an instance I' of

Problem 2 with which set D also with cardinality μ is associated.

According to Lemma 2.1, each intermediate AN, based on the orientation of distance vector to its neighbor, covers a distance type $\underline{d_i}$ where $|\underline{d_i}| \in D$. These $\underline{d_i}$'s are items to be packed in instance I'. Let y_i denote the number of ANs of type $\underline{d_i}$. Then, the total distance l covered by all intermediate ANs is expressed as

$$l = \sum_i y_i |\underline{d_i}| \tag{3.1}$$

Let profit V in instance I be equal to L. Assuming the weight of each AN is 1, i.e., $w_i = 1$, the total weight of all intermediate ANs amounts to the number of intermediate ANs, i.e.,

$$\sum_i y_i = N \tag{3.2}$$

Considering the statement above and assuming $W = N$, the process of constructing I' from I occurs in polynomial time.

In Problem 2, we are seeking a yes/no answer to the question "Can we, by using N intermediate ANs, cover a total distance of $l \geq L$?" If the answer to Problem 1 is yes, we can fill the knapsack such that a minimum profit of V is reached without exceeding a maximum weight of W.

Through the reduction above, it is feasible to cover a length of at least L by placing at most N additional nodes. In addition, if the answer to Problem 2 is no, which is a special instance I of Problem 1 with $V = L$, $w_i = 1$, and $W = N$, Problem 1 will also have no answer. This implies a polynomial reduction from Problem 1 to Problem 2. □

Therefore, we conclude that Problem 2 is NP-hard. In the next subsection, we provide an exhaustive search algorithm to solve Problem 2.

3.1.2 Exhaustive Search Algorithm

Our exhaustive search algorithm uses a number of intermediate ANs and tries to rearrange their locations so as to establish global connectivity, until the smallest number of ANs that connect the entire graph is identified. We use κ to denote the number of intermediate ANs in exhaustive search. There is a finite set of feasible locations representing the candidate coordinates of intermediate AN locations in HCS. Generally speaking, all coordinate points of HCS except those occupied by pre-deployed clusters are feasible. The number of feasible locations M is then derived as

$$M = \lceil \frac{\Omega}{1.5\sqrt{3}\, r^2} \rceil - G \tag{3.3}$$

where $1.5\sqrt{3}\, r^2$ represents the area of a hexagonal cell of edge length r, Ω is the field area, and G is the number of pre-deployed clusters. Those M feasible locations are then stored in an $M \times 2$ matrix $\mathbf{F}$.

In our exhaustive search algorithm, we test all $\binom{M}{\kappa}$ possible combinations of M coordinates in $\mathbf{F}$ and check if there is one configuration that accomplishes connectivity of all clusters. If not, we increase κ by 1 and repeat the same process until the least number of intermediate ANs rendering global connectivity is reached. The algorithmic pseudo code is given in Algorithm 1.

One may notice the considerable computational complexity of the nested 'for' loop. Given M feasible AN locations, when the optimal solution is reached, say connecting the entire graph with κ ANs, then the runtime of exhaustive algorithm is at least in the order of $\mathcal{O}(\sum_{i=1}^{\kappa-1} \binom{M}{i})$. It can be shown, by Stirling's formula and binomial theorem, that the runtime is bounded

as shown below.

$$\frac{(M+1)^{\kappa-1}}{(\kappa-1)!} < \mathcal{O}(\sum_{i=1}^{\kappa-1} \binom{M}{i}) < 2^M \tag{3.4}$$

In practice, we are able to strategically preclude some locations that have very low possibility of accommodating an AN. For instance, an AN may not be placed too close to a pre-deployed cluster, and all ANs typically are, but not always, located inside the convex hull containing all pre-deployed clusters. With this strategy, we can reduce the size of $\mathbf{F}$ to some extent. However, to the best of our knowledge, there is no systematic strategy of reducing the number of feasible locations.

Algorithm 1 Exhaustive Search Algorithm

```
Input: Location of pre-deployed clusters
Output: Coordinates of intermediate ANs
Establish finite HCS and
Put coordinates of feasible AN positions in F_{M×2}
Set κ = 0
while (Graph G not fully connected) do
  κ += 1
  for all (M choose κ) possible combos of feasible positions do
    Place κ ANs at these positions
    if G is connected then
      Set coordinates of κ ANs from F_{M×2}
      break
    end if
  end for
end while
```

3.2 Heuristic Algorithm 1: GDO algorithm

As the node placement problem described in the previous section is NP-hard, it is realistic to find a heuristic near-optimal solution offering a reasonable time complexity. Hence, this

section provides a description of our heuristic connectivity algorithms and their complexity analysis.

As illustrated in chapter 2, we model the communication range of an AN gateway by a hexagonal cell. The communication range R of an AN is approximated by $(12n + 7)r$ where n is a positive natural number.

A robust connectivity criterion is defined as when two large hexagons have a common edge. Consequently, we are dealing with the task of connecting a number of hexagons within the network graph by optimally placing a number of hexagons of the same size between each pair as needed. Consequently, we are dealing with the task of connecting a batch of large hexagons within the network graph by optimally placing a number of hexagons of the same size between each pair. As this task can be abstracted as a distance optimization problem within the HCS, we introduce a class of geometric distance optimization algorithm.

Given a number of clusters distributed in the plane, the first step is to set up the HCS origin and axes. We set the origin of the HCS at the center of geometry of these clusters. Then, we set up x- and y-axis for HCS at the origin. After setting up the HCS, the coordinates of all clusters in HCS are specified. Then, the following 5-step iterative algorithm leads us to achieving connectivity of the network graph using a number of additional ANs. The gist of these steps is generalized in Algorithm 2.

Algorithm 2 GDO Algorithm

Input: Location of pre-deployed clusters
Output: Coordinates of intermediate ANs
Step 1: Establish HCS, calculate MST
while (Graph $\mathcal{G}$ not fully connected) **do**
Step 2: Identify clusters P and Q to be connected
Step 3: Connect P and Q by sloving the optimization problem
Step 4: Recalculate MST and check stoppage rule
Step 5: Break if graph is fully connected, otherwise go to Step 2
end while

Step 1: Calculate Minimum Spanning Tree

In this step, the distance-weighted minimum spanning tree of the current deployed clusters is identified using the distance metric defined in the previous section. First, the distances between each pair of clusters are calculated and stored in a weight function. Then by employing Kruskal's algorithm, the minimum spanning tree is calculated. We store all of the edge lengths in an array $M^{(k)}$ of size m where k represents the iteration number and m represents the number of edges in the minimum spanning tree. Corresponding to this array, we keep an $m \times 2$ matrix each row of which containing the end cluster nodes connected by the associated edge in $M^{(k)}$.

Step 2: Identify the Pair of Connecting Clusters

Two strategies of identifying the pair of connecting clusters are introduced. Those are namely the shortest edge and the longest edge cluster pairs referred to as ShortestHCS and LongestHCS, respectively. This step is performed by searching for the smallest- or largest-valued element in $M^{(k)}$. Assuming this identified element implies clusters P and Q to be connected in the next step.

Step 3: Connect the Pair of Selected Clusters

This step attempts to achieve two goals. The primary goal is to connect the selected pair of clusters P and Q (representing either gateway ANs or intermediate ANs) with the least number of intermediate ANs. The secondary goal is to deploy those intermediate ANs, which we denote as Υ, in a way to bring the remaining isolated clusters closer thereby helping their future connectivity. We refer to the set of intermediate ANs in iteration k as $A^{(k)}$. In order to achieve these goals, we utilize the following iterative process.

Case 1: When using a single AN suffices to connect P and Q, $A^{(k)} = \{\Upsilon^{(k)}\}$ and $\Upsilon^{(k)}$ is placed between P and Q. The exact position of $\Upsilon^{(k)}$ is calculated by solving the optimization problem given in this section.

In order to identify the coordinates (x, y) of the new node $\Upsilon^{(k)}$ placed in the k-th iteration of Case 1, we introduce a pair of conditions.

1. Maintain the connectivity of all three ANs, namely, P, Q, and the new node $\Upsilon^{(k)}$.
2. Identify the position of node $\Upsilon^{(k)}$ by maximizing the probability of connecting the newly aggregated cluster to other pre-deployed clusters and minimizing the overlap area between $\Upsilon^{(k)}$ and the other two nodes.

In short, we want to connect the pair of selected clusters by placing $\Upsilon^{(k)}$, as well as expect to facilitate the connectivity of remaining clusters by intelligently placing $\Upsilon^{(k)}$ when possible. Accordingly, we formulate the following distance maximization problem graphically depicted in Fig. 3.1.

$$\max_{x,y} \quad |(P, \Upsilon^{(k)})| + |(Q, \Upsilon^{(k)})| \tag{3.5}$$

$$\text{S.T.} \quad (P, R) \cdot (R, \Upsilon^{(k)}) = 0 \tag{3.6}$$

$$(Q, S) \cdot (S, \Upsilon^{(k)}) = 0 \tag{3.7}$$

$$|(P, R)| \leq \lambda \tag{3.8}$$

$$|(Q, S)| \leq \lambda \tag{3.9}$$

$$|(R, \Upsilon^{(k)})| \leq 8n + 4 \tag{3.10}$$

$$|(S, \Upsilon^{(k)})| \leq 8n + 4 \tag{3.11}$$

In this problem, $\lambda = 12n + 7$ and $(P, R), (Q, S) \in \Psi$.

$$\Psi = \{(\lambda, 0), (-\lambda, 0), (0, \lambda), (0, -\lambda), (\lambda, -\lambda), (-\lambda, \lambda)\} \tag{3.12}$$

By definition of Ψ, $\Upsilon^{(k)}$ is connected to P (or Q) and has the least overlap area with P (or Q) as long as $\Upsilon^{(k)}$ assumes its value from the set of feasible positions determined by the inner product constraint (3.6) (or (3.7)). In other words, the distance between $\Upsilon^{(k)}$ and P (or Q) is maximized along the direction of vector (P, R) (or (Q, S)).

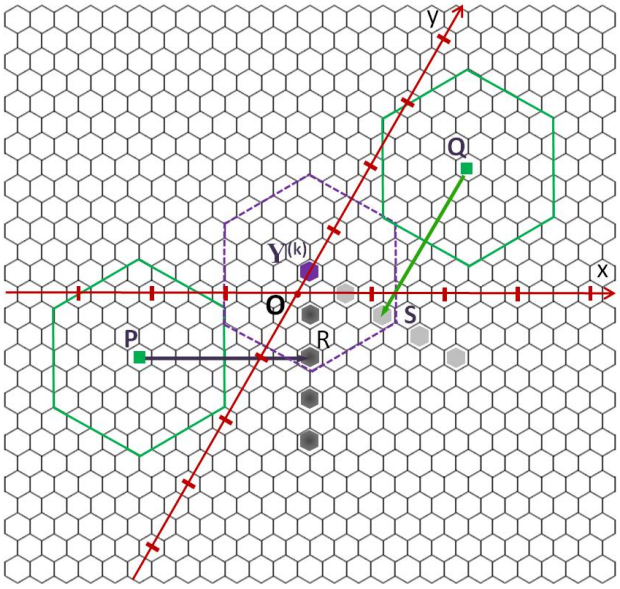


Figure 3.1: A graphical representation of the feasible points of $\Upsilon^{(k)}$ in Case 1.

In what follows, we explain the meaning of these constraints. As shown in Fig. 3.1, the cells in solid dark and light grey color represent all feasible positions of $\Upsilon^{(k)}$ assuring connectivity to nodes P and Q. These two inner product constraints maintain $\Upsilon^{(k)}$ slides along the line perpendicular to (P, R) at point R and (Q, S) at point S as shown by gray cells in Fig. 3.1, while $|(P, R)|$ identifies the furthest position $\Upsilon^{(k)}$ can reach along the direction(s) of (P, R) while staying connected to P. In short, the inner product constraint identifies the track of movement for $\Upsilon^{(k)}$ and $|(R, \Upsilon^{(k)})|$ controls the range on the track. Although (P, R) has six possible directions, we do not need to inspect them all. Based on the relative

position of Q with respect to P, only one facade of each node needs to be considered. The same explanation also applies to Q. To solve the optimization problem, we first ignore the inequality constraints and calculate the position of $\Upsilon^{(k)}$ by the two equality constraints, we verify the connectivity of the two gateway nodes. With different pairs of (P, R) and (Q, S) selected in set Ψ, there will be two solutions associated with the coordinates of $\Upsilon^{(k)}$ satisfying inequality constraints. The one closer to the origin is chosen in order to improve the probability of connecting to other clusters. Now, let us assume we solve Case 1 of Step 3 leading to specifying the coordinates of intermediate node $\Upsilon^{(k)}$ as (x, y), P as (p_1, p_2), and Q as (q_1, q_2). Now select (P, R) to be $(\lambda, 0)$ and (Q, S) to be $(0, -\lambda)$. Then, solving the equality constraint (3.6) yields

$$x + \frac{1}{2}y \;=\; p_1 + \lambda + \frac{1}{2}p_2 \tag{3.13}$$

Similarly, solving the equality constraint (3.7) yields

$$\frac{1}{2}x + y \;=\; \frac{1}{2}q_1 + q_2 - \lambda \tag{3.14}$$

By solving Eq. (3.13) and Eq. (3.14), the coordinates of $\Upsilon^{(k)}$ are identified which are required to be integers within the HCS. More importantly, the position of $\Upsilon^{(k)}$ must satisfy the inequality constraints (3.8), (3.9), (3.10), and (3.11). In this case, all possible combinations of (P, R) and (Q, S) in set Ψ are to be inspected. Note that (P, R) and (Q, S) cannot be parallel to each other, otherwise Eq. (3.13) and Eq. (3.14) have no joint solution. This rule only applies to Case 1.

Case 2: When using only one AN cannot establish connectivity between P and Q, the following iterative process is initiated. In this case, we assume $A^{(k)} = \{\Upsilon_1^{(k)}, \cdots, \Upsilon_{\nu^{(k)}}^{(k)}\}$, i.e., $\nu^{(k)}$ ANs are required to connect P and Q where $\nu^{(k)} \geq 2$.

1. Place an AN next to each of the two clusters P and Q. Referred to as $\Upsilon_i^{(k)}$ and $\Upsilon_{i+1}^{(k)}$, $1 \leq i < \nu^{(k)}$, these two ANs are connected to their associated clusters, P and Q, respectively.

2. Find the exact positions of $\Upsilon_i^{(k)}$ and $\Upsilon_{i+1}^{(k)}$ by minimizing the sum of three distances between $\Upsilon_i^{(k)}$ and $\Upsilon_{i+1}^{(k)}$, $\Upsilon_i^{(k)}$ and the origin, $\Upsilon_{i+1}^{(k)}$ and the origin.

3. Inspect connectivity between $\Upsilon_i^{(k)}$ and $\Upsilon_{i+1}^{(k)}$. If connected, terminate the process. If more ANs are needed, replace the two end nodes with $\Upsilon_i^{(k)}$ and $\Upsilon_{i+1}^{(k)}$ and then solve the problem of connecting them by going through the same process described before.

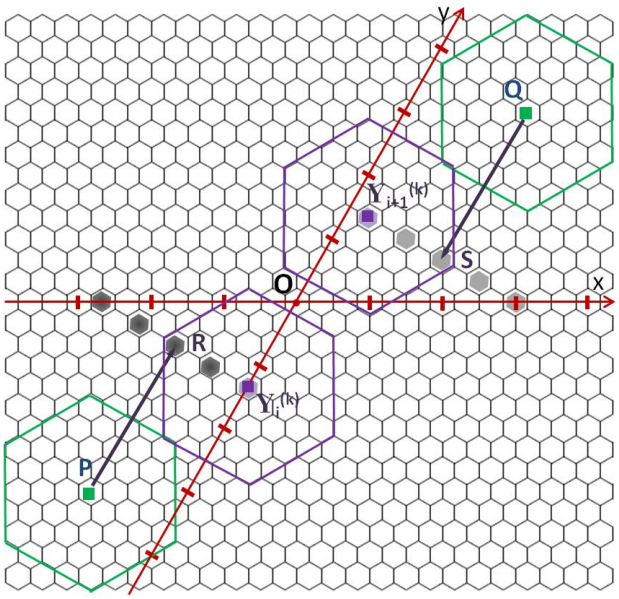


Figure 3.2: A graphical representation of the feasible points of $\Upsilon_i^{(k)}$ and $\Upsilon_{i+1}^{(k)}$ in Case 2.

We assume the AN pair P and Q are being connected using ANs $\Upsilon_i^{(k)}$ and $\Upsilon_{i+1}^{(k)}$ with coordinates (x_i, y_i) and (x_{i+1}, y_{i+1}), respectively. We use two sets of constraints to find feasible positions of $\Upsilon_i^{(k)}$ and $\Upsilon_{i+1}^{(k)}$ corresponding to P and Q, respectively. Then, different combinations of these feasible positions of $\Upsilon_i^{(k)}$ and $\Upsilon_{i+1}^{(k)}$ are inspected in order to find the one minimizing the distance between $\Upsilon_i^{(k)}$ and $\Upsilon_{i+1}^{(k)}$, $\Upsilon_i^{(k)}$ and the origin, as well as $\Upsilon_{i+1}^{(k)}$ and the origin. As shown in Fig. 3.2, dark gray cells on the line segment perpendicular to (P, R)

represent all feasible positions of $\Upsilon_i^{(k)}$. Similarly, light gray cells on the line segment perpendicular to (Q, S) represent all feasible positions of $\Upsilon_{i+1}^{(k)}$. The distance optimization problem is accordingly described as below.

$$\min_{x_i, y_i, x_{i+1}, y_{i+1}} \quad |(\Upsilon_i^{(k)}, \Upsilon_{i+1}^{(k)})| + |(O, \Upsilon_i^{(k)}| + |(O, \Upsilon_{i+1}^{(k)})| \tag{3.15}$$

$$\text{S.T.} \qquad (P, R) \cdot (R, \Upsilon_i^{(k)}) = 0 \tag{3.16}$$

$$(Q, S) \cdot (S, \Upsilon_{i+1}^{(k)}) = 0 \tag{3.17}$$

$$|(P, R)| \leq \lambda \tag{3.18}$$

$$|(Q, S)| \leq \lambda \tag{3.19}$$

$$|(R, \Upsilon_i^{(k)})| \leq 8n + 4 \tag{3.20}$$

$$|(S, \Upsilon_{i+1}^{(k)})| \leq 8n + 4 \tag{3.21}$$

The set of constraints used to find feasible positions of $\Upsilon_i^{(k)}$ are expressed by Eq. (3.16) and Eq. (3.20), while the one used to find positions of $\Upsilon_{i+1}^{(k)}$ are expressed by Eq. (3.17) and Eq. (3.21). Here, (P, R) and (Q, S) are in Ψ.

However, (P, R) only has one choice in Ψ because the vector (P, R) can only point to the facade that faces Q. Similarly, (Q, S) only has one choice in Ψ because the vector (Q, S) can only point to the facade that faces P. After finding all feasible positions for $\Upsilon_i^{(k)}$ and $\Upsilon_{i+1}^{(k)}$, we have to conduct an exhaustive search among different combinations of the feasible positions to find the unique combination that minimizes the objective function.

Step 4: Recalculate Distance Matrix

After forming a connected cluster containing nodes A_k(s), P, and Q, the distance matrix between clusters need to be recalculated. In the calculation, intermediate ANs are treated as normal clusters. This distance matrix will be used to calculate the new distance-weighted minimum spanning tree for the next iteration. Before we pass it down, we have to prevent those connected clusters from being selected again, so we set the distance between connected clusters in order to prevent them from being selected.

In the case of LongestHCS, the distance of the selected element(s) is replaced by a small number. The distance between A_k(s) and P, A_k(s)and Q are set to be small values. This reflects, in the distance weight matrix, that P, Q and A_k(s) between them form a clusters, within which, ANs are close to each other. Then, the new minimum spanning tree is recalculated based on the new distance weights. In the case of ShortestHCS, the distance of the selected element(s) is replaced by a large number. Finally, we have a new array $M^{(k+1)}$ at iteration $(k+1)$.

Step 5: Stoppage Rule

If the stoppage rule below is not met, go back to Step 2. The description of the stoppage rule is as follows. As the number of iterations grows, the number of connected clusters increases. The LongGDO algorithm stops when the largest-valued element in M is smaller than or equal to a given threshold implying to have a fully connected graph; while the ShortGDO algorithm stops when the picked up pair of clusters are already connected. Consequently, the algorithm stops with a fully connected graph.

3.3 Heuristic Algorithm 2: EGDO algorithm

The main algorithm of interest in this section is referred to as enhanced geometric distance optimization algorithm or EGDO. The name stems from the fact that the algorithm is an enhanced version of a pair of GDO algorithms proposed in Section 3.2 and [40]. Our work of [40] shows that the LongestHCS algorithm outperforms ShortestHCS algorithm in most scenarios. The main improvement of EGDO algorithm over GDO (LongestHCS) algorithm is its significantly improved time complexity. As described in the following context, the latter is achieved by locally modifying the existing MST in iterative steps as opposed to forming a new MST after each iteration as it was the case of GDO algorithms. In the rest of this section, we refer to the LongestHCS algorithm illustrated in Section 3.2 and [40] as the GDO algorithm.

Similar to GDO algorithm (Algorithm 2), the first step of EGDO is also the set-up of HCS origin and axes. The details are omitted as one can refer to Section 3.2. After the HCS is set up, the minimum spanning tree among the pre-deployed nodes is calculated and potentially adjustable nodes are identified. Then we select a pair of nodes to be connected and launch intermediate ANs placement iteration. After connectivity between this selected pair of nodes are established, the MST is modified according to the distance between the potentially adjustable nodes and the newly placed intermediate nodes. These procedures will be repeated before global connectivity is established. The individual steps of EGDO algorithm are described in details below. By convention, we generalize the gist of the algorithm first.

Step 1: Calculate MST and Find Terminals

In this Step, we first initialize the iteration counter k to 1. Then, we calculate the initial distance weighted MST formed by a given set of clusters. The distance between a pair of

Algorithm 3 EGDO Algorithm

Input: Location of pre-deployed clusters
Output: Coordinates of intermediate ANs
Step 1: Establish HCS, calculate MST, find terminals
while (Graph $\mathcal{G}$ not fully connected) **do**
Step 2: Identify clusters P and Q to be connected
Step 3: Connect P and Q using minimum no of ANs
Step 4: Modify MST using ANs placed in Step 3
Step 5: Break if graph is fully connected
end while

clusters is calculated based on the distance measure definition in Section 2.2. The MST is calculated using Kruskal algorithm [76]. The calculated MST is presented by an $N \times 2$ matrix $\mathbf{M}^{(k)}$ in which k represents the iteration number, each row identifies the two vertices of an edge, and N is the number of edges. Matrix $\mathbf{M}^{(k)}$ includes the edges in an increasing order of edge length. Given $\mathbf{M}^{(k)}$, we find all terminal nodes. A terminal node is an AN in the network connected to only one AN. From these terminals, we establish a set of nodes of interest for use in the next step. Once we place a new intermediate AN, we compare the distances between this new AN and the nodes in which we are interested. We refer to the nodes in which we are interested as potentially adjustable nodes or PANs. If the distance between the new AN and a PAN is smaller than the edge length connected to that PAN, we change the route by deleting the edge connected to the PAN and connecting the PAN with the new AN.

Step 2: Identify the Pair of Connecting Clusters

Since the rows of distance matrix are in increasing order, the last row of $\mathbf{M}^{(k)}$ identifies the edge to be connected in the next step. Let us assume the elements of the last row are clusters P and Q.

One might be curious as to why we select the pair of nodes that have the longest distance in

between. The answer lies in the fact that to always choose the longest distance pair yields our best experimental results from among the variants tested. Other variants include always selecting the shortest distance pair, alternating between shortest and longest distance pairs, and several types of clustering strategies discussed in [77].

Step 3: Connect the Pair of Selected Clusters

This step is the same as Step 3 in Algorithm 2 which can be found in Section 3.2, we omit the details here.

Step 4: Modify MST

In this step, first the current PAN set is identified following the placement of one or more intermediate ANs in the previous step. In Case 1 of Step 3, this step takes place after the single AN is placed. In Case 2 of Step 3, this step is initiated right after $\Upsilon_i^{(k)}$ and $\Upsilon_{i+1}^{(k)}$ are placed. It is observed that placing one or two new ANs can only introduce local changes to the topology of the network, i.e., topology changes are limited to the neighboring nodes of newly placed ANs.

In a given MST, a 2-connected node or terminal may be connected to a newly placed ANs guaranteed not to create a loop. However, connecting a node with 3 or more edges to a PAN may result in creating a loop. In order to avoid the possibility of creating a loop, only 2-connected nodes and terminals in an MST are considered as PANs.

Accordingly, we propose a *line graph* method in order to identify these PANs. A line graph starts from a terminal node. This terminal is the first node. Then following the line, the second node is reached and so on. The line ends when a 3-connected node, a terminal node, or a currently selected node is reached. If a node on the line does not belong to any of the

categories above, then it is a 2-connected node and is subsequently added to the current PAN set. Note that we are only interested in terminals or 2-connected nodes because 3-connected nodes cannot be modified or else the entire spanning tree will become disconnected. The nodes on the line represent the current PAN set which is one subset of all PANs.

Having identified the current PAN set, the MST can be modified accordingly. In order to always keep the last row of $\mathbf{M}^{(k)}$ as the edge to be connected, we modify the MST according to the cases of *Step 3* above.

If we are to follow Case 1 and place a single AN, we remove the last row $[P\ Q]$ of $\mathbf{M}^{(k)}$ and insert two rows $[P\ \Upsilon^{(k)}]$ and $[Q\ \Upsilon^{(k)}]$ as the top rows of $\mathbf{M}^{(k)}$. This changes the representation of MST from $\mathbf{M}^{(k)}$ in this iteration to $\mathbf{M}^{(k+1)}$ in the next iteration.

If we are to follow Case 2 when adding a pair of ANs $\Upsilon_i^{(k)}$ and $\Upsilon_{i+1}^{(k)}$, we replace the last row of $\mathbf{M}^{(k)}$ with $[\Upsilon_i^{(k)}\ \Upsilon_{i+1}^{(k)}]$ and then insert $[\Upsilon_i^{(k)}\ P]$ and $[\Upsilon_{i+1}^{(k)}\ Q]$ as the top rows of $\mathbf{M}^{(k)}$ for iteration $(k+1)$.

Second, for each newly added AN in set $A^{(k)}$, say $\Upsilon_i^{(k)}$, we compare the distance between $\Upsilon_i^{(k)}$ and the j-th node on a line graph, $\delta_{ij}^{(k)}$, with the distance between j-th and $(j+1)$-th node on the line. If $\delta_{ij}^{(k)}$ is smaller, we modify the tree by deleting the edge between j-th and $(j+1)$-th node on this line graph, and then adding the edge between $\Upsilon_i^{(k)}$ and j-th node. After this modification, $\Upsilon_i^{(k)}$ becomes a 3-connected node, as $\Upsilon_i^{(k)}$ will be connected with P and Q or other intermediate ANs between P and Q, as well as j-th node on this line graph. Therefore, the edges ending at $\Upsilon_i^{(k)}$ can never be modified. After making each modification, we stop searching for other PANs on the current or other line graphs.

In this process, we always compare $\delta_{ij}^{(k)}$ with the edge lengths of MST entries and insert the associated edges at the right place in order to preserve the ascending order of edge lengths in MST matrix.

Step 5: Check Stoppage Rule

When the selected pair of clusters is found to have already been connected, the algorithm stops. Otherwise, we increment the iteration counter k by 1 and go back to *Step 2*.

It is worth emphasizing that the main difference between EGDO and GDO algorithm is the fact that EGDO algorithm adjusts the MST due to local topology changes associated with adding ANs as opposed to recalculating a new MST done by GDO. This leads to a significant reduction of average time complexity in EGDO algorithm as analyzed in Section 3.4.2 and proved through numerical experiment in Chapter 4.

3.4 Analysis of Complexity

In this subsection, we analyze the computational complexity of EGDO in comparison with GDO algorithm. We first determine the time complexity of solving the optimization problem of Step 3 as it is the common step shared by both GDO and EGDO algorithms, and then analyze the complexity of the recursive algorithm.

3.4.1 Complexity of Solving the Optimization Problem

The total number of cases that need to be inspected in order to solve the optimization problem of Step 3 of Section 3.2 is equal to $\binom{6}{2} - 3 = 12$. As mentioned before, each (P, R) and (Q, S) has 6 possible directions but cannot be in parallel or else there is no solution to Eq.(3.13) and Eq.(3.14). However, this number can be reduced to 4 based on the relative position of P with respect to Q. We argue that solving the optimization problem in Case 1 of Step 3 takes constant time as solving Eq.(3.13) costs constant time.

If we are to follow Case 2 in *Step 3*, we conduct an exhaustive search for combinations of feasible positions for $\Upsilon_i^{(k)}$ and $\Upsilon_{i+1}^{(k)}$. The total number of all individual feasible positions for $\Upsilon_i^{(k)}$ is

$$\frac{2(R-r)}{3r}+1=\frac{2(\lambda-1)}{3}+1=8n+5 \tag{3.22}$$

The same number also represents the total number of all individual feasible positions for $\Upsilon_{i+1}^{(k)}$. Thus, the total number of combinations that either GDO or EGDO algorithm need to inspect in Case 2 of *Step 3* is $(8n+5)^2$. This is an exhaustive search within a finite number of candidates since n is determined by the communication range of an AN. Therefore, finding the particular combination of the pair $(\Upsilon_i^{(k)}, \Upsilon_{i+1}^{(k)})$ that minimizes their distance takes constant time.

Next, we note that Case 2 follows the same approach iteratively until P and Q are connected. This is because $\nu^{(k)}$, the total number of intermediate ANs required to connect P and Q in iteration k, is a finite number known at the beginning of each iteration and is decreasing in consequent iterations as the selected edge length within MST never increases. Hence, we conclude that solving the optimization problem of Case 2 also takes constant time. This constant is a function of R/r as well as the number of ANs needed to connect the two selected nodes.

3.4.2 Complexity of EGDO Algorithm

In this subsection, we analyze the complexity of the other steps of the EGDO algorithm. In Step 1, the time complexity of calculating the distance weight matrix between all nodes is in the order of $\mathcal{O}(N_0^2)$ provided that there are N_0 pre-deployed clusters. To calculate the minimum spanning tree takes $\mathcal{O}(E \log N_0)$ where E is the number of edges in the initial network graph. Since we need to inspect all edges in the weight matrix, E is close to N_0^2.

To find terminals, we need to inspect the degree of each node. Completing this process takes a time complexity of $\mathcal{O}(N_0)$. Hence, the total complexity of Step 1 is in the order of $\mathcal{O}(N_0^2) + \mathcal{O}(N_0^2 \log N_0) = \mathcal{O}(N_0^2 \log N_0)$. In Step 4 and in order to find the sets of PANs, we start from each terminal node and stop after reaching a certain type of node. This is a search process that usually stops way before going through all nodes at present. Assuming that the algorithm starts with N_0 pre-deployed clusters, stops after n_t iterations, $\nu^{(k)}$ is the number of intermediate ANs added in iteration k with $\nu^{(0)} = 0$, and $\Gamma(k) = N_0 + \sum_{i=0}^{k} \nu^{(i)}$ represents the total number of ANs after iteration k. Thus, the worst case time complexity of Step 4 is $\mathcal{O}(\Gamma(k-1))$ at k-th iteration. However, the average time complexity is much shorter as the search stops way before $\Gamma(k-1)$. Step 2 and Step 5 takes constant time which can be ignored.

In summary, the worst case time complexity of EGDO algorithm is in the order of

$$\mathcal{O}(N_0^2 \log N_0) + \sum_{k=1}^{n_t} \mathcal{O}(\Gamma(k-1)) < \mathcal{O}\left[N_0^2 \log N_0 + n_t \Gamma(n_t)\right]$$

However, the average time complexity is much shorter considering the fact that the search process of Step 4 stops before $\Gamma(k-1)$ as showed by our experiments in Section **??**.

3.4.3 Complexity of GDO Algorithm

In comparison, we analyze the complexity of the other steps of GDO algorithm. In Step 1 the time complexity of calculating the distance weight matrix between all nodes is $\mathcal{O}(N_0^2)$ provided that there are N_0 pre-deployed clusters. Similarly, the time complexity of Step 1 is in the order of $\mathcal{O}(N_0^2 \log N_0)$. Since the GDO algorithm recalculates the MST after each iteration and the number of nodes in MST is increasing, the runtime accumulates. With the

same definition of n_t, $\nu^{(k)}$, and $\Gamma(k)$, the time complexity of GDO algorithm is

$$\sum_{k=0}^{n_t} \mathcal{O}\left[\Gamma(k)^2 \log \Gamma(k)\right] < \mathcal{O}\left[n_t \Gamma(n_t)^2 \log \Gamma(n_t)\right]$$

The worst case time complexity of GDO algorithm is hence in the order of $\mathcal{O}\left[n_t \Gamma(n_t)^2 \log \Gamma(n_t)\right]$. Even though the bound is not tight, GDO algorithm has a much higher time complexity than EGDO as presented in Chapter 4.

Chapter 4

Experimental Results

In this chapter, we provide numerical evaluation of GDO and EGDO algorithms. We start with comparing the results of our algorithm with exhaustive search algorithm. The latter serves as the benchmarking baseline finding the global optimal solution to the problem of node placement albeit with a very high time complexity. We show that our algorithms provide results close to the optimal solution given by exhaustive search within a limited area where the time complexity of exhaustive search is affordable. Then, we compare the performance of EGDO algorithm with the class of GDO and variants of the SMT [9] algorithms which solves the problem efficient yet with disk model rather than hexagons. The results shows SMT algorithms, without consideration of connectivity robustness, has advantage over GDO and EGDO on the number of AN used. But if we endow SMT with the same level of embedded robustness by replacing disk model with hexagon model, SMT algorithm lost its advantage and tends to use more intermediate nodes than GDO algorithms. As the title of this dissertation is regarding to sparse network, we also investigate numerically the range of network sparsity within which our proposed algorithms retain advantage. Our experimental results cover AN cost, i.e., the number of intermediate ANs, runtime, robustness, and the effects of HCS.

4.1 Comparison with Exhaustive Search Algorithm

In this subsection, we compare the AN cost of EGDO algorithm with exhaustive search algorithm, without considerations of runtime, in order to show our EGDO algorithm is in fact producing results close to the global optimal solution.

In order to examine the deviation of EGDO solution from the globally optimal solution, we run experiments in a field of $4500 \times 4500m^2$, with $r = 50m$ and $R = 350m$. The selection of parameters allows for completing exhaustive search experiments in realistic time. Fig. 4.1 gives the results. The horizontal axis is the number of pre-deployed clusters varying in the range from 2 to 25 and the vertical axis is the AN cost. For each point on the x-axis, we run 50 different configurations and record the number of intermediate ANs used. Then, we fit the data to a polynomial curve. The blue and black curves show the AN cost of EGDO and exhaustive search algorithms, respectively. While the AN cost of EGDO is always higher than that of exhaustive search, the largest gap observed between two curves along the vertical axis is less than 10%. Without being able to offer a mathematical proof, the gap falls in the range of 1.1-approximation ratio. It can also be observed that both curves start dropping beyond a certain point. This is because when the number of pre-deployed clusters grows, the network becomes denser and requires less ANs to establish connectivity. This aspect will be further investigated in the following subsections.

On the aspect of runtime, the completion time of EGDO algorithm is in the range of 1 to 2 seconds in our simulation setting. However, the exhaustive search algorithm takes from several minutes to over ten hours to complete within the same simulation settings.

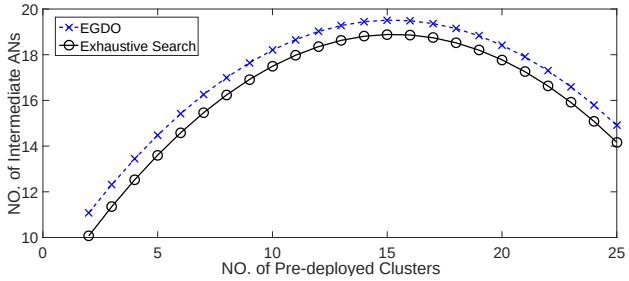


Figure 4.1: An AN cost comparison of EGDO and exhaustive search algorithms.

4.2 Performance Comparison of SMT and EGDO Algorithm

In this subsection, we compare the performance of SMT, GDO, and EGDO algorithms measured by the minimum AN cost and runtime. When comparing the two classes of algorithms, we also consider the fact that GDO and EGDO algorithms dynamically update minimum spanning trees while the original SMT algorithm forms the minimum spanning tree once statically. Therefore, we modify the SMT algorithm to become a dynamic algorithm in which the minimum spanning tree is recalculated after connecting every edge. We refer to the original static SMT algorithm as StaSMT and the revised dynamic SMT algorithm as DynSMT. Because SMT algorithms model the communication range of a node as a disk while GDO and EGDO do so as a hexagon embedded within the disk, SMT algorithms cover distance with a smaller number of ANs in average. Yet one should notice that this is not a fair comparison as a circle always covers a longer distance than the hexagon embedded in it. Additionally, it is important to note that DynSMT and GDO algorithms recalculate the entire spanning tree each time after a pair of clusters are connected. Hence, the time complexity of these algorithms is higher than those of StaSMT and EGDO algorithms.

The experiments are conducted in a $200km \times 200km$ field with $r = 50m$ and $R = 4550m$. Fig. 4.2 provides an AN cost comparison among the StaSMT, DynSMT, GDO, and EGDO algorithms for a fixed field size. The results show that GDO algorithm uses an average of

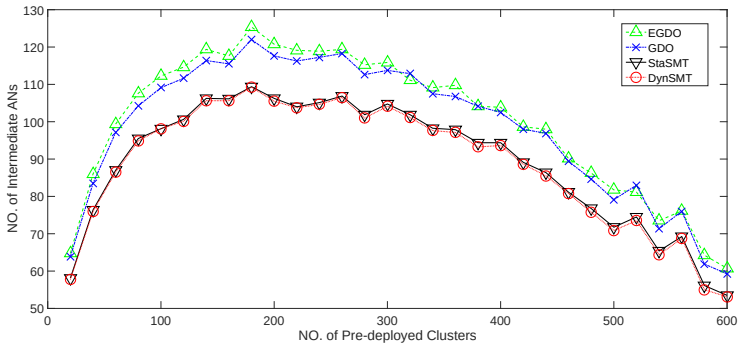


Figure 4.2: An AN cost comparison among StaSMT, DynSMT, GDO, and EGDO algorithms.

10% more AN resources than StaSMT. Further, the EGDO algorithm sometimes consumes a slightly larger number of AN resources than GDO algorithm, because in doing local modification it might miss some larger scale variations in network graph topology caused by a newly added AN. However, comprehensive experimental results have shown that these differences are negligible. Interestingly, it is also observed that there is no significant difference between the performance of the two variants of the SMT algorithm. This is alluded to the fact that unlike the GDO algorithm, the recalculated minimum spanning tree in DynSMT is not much different from the previously calculated minimum spanning tree obtained by StaSMT algorithm. The results of all four algorithms show an initial rise followed by a drop alongside some variations. The rise is related to the fact that an increase in the number of pre-deployed clusters N in a sparse network requires utilizing more intermediate ANs. As the N grows even larger within a fixed field size, the sparse network evolves to a dense network covering most of the field with AN gateways thereby reducing the number of intermediate ANs. All four algorithm tend to use the same number of intermediate ANs as the value of N grows to 600 in this setting.

Fig. 4.3 includes a comparison of runtimes among the StaSMT, DynSMT, GDO, and EGDO algorithms for the same fixed field size. It is observed that StaSMT has the lowest runtime because it only forms the minimum spanning tree once. DynSMT has a much longer runtime than the other three algorithms in general. Among three dynamic algorithms GDO, EGDO,

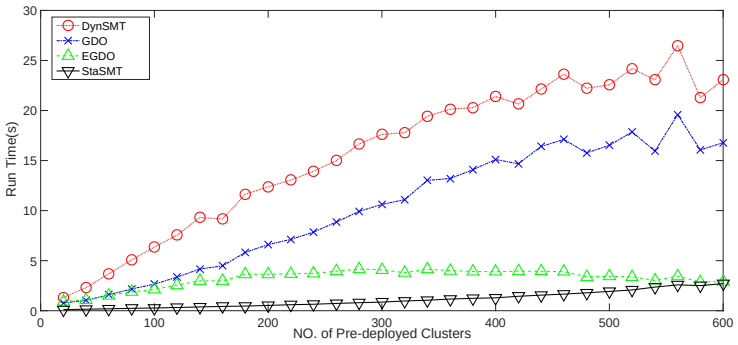


Figure 4.3: A runtime comparison of StaSMT, DynSMT, GDO, and EGDO algorithms.

and DynSMT, EGDO has the shortest runtime by far. While the runtime is generally higher than that of StaSMT, it gets closer to that of StaSMT for values of N greater than 500. This behavior is related to the fact that the cost of calculating the minimum spanning tree increases as N grows and also a smaller number of intermediate nodes are needed.

Considering the fact that the AN cost performance of DynSMT is slightly better than that of StaSMT but its runtime is significantly longer, we conclude that the advantage of DynSMT does not justify its increased time complexity. Therefore, we mainly compare the performance of StaSMT and EGDO algorithms in the rest of our experiments, considering comparable performance of GDO and EGDO but much better time complexity of EGDO. We note that EGDO algorithm uses an additional 10% AN resources in average due to the use of a hexagon instead of a circle to represent the communication range of a node and also has a slightly longer runtime compared to StaSMT algorithm. However, it offers much better robustness characteristics as reported in the next subsection.

4.3 Experiment on Robustness - Partial and Global Robustness Tests

In this subsection, we evaluate the robustness of network connectivity algorithms by applying perturbations to the position of nodes. In each experiment, we first establish global network connectivity applying EGDO and StaSMT algorithms. Once connectivity is established, we introduce random perturbations to the position of pre-deployed clusters. This scenario is referred to as partial perturbation as it does not perturb the position of intermediate ANs added for establishing connectivity. We also conduct additional robustness experiments in which all existing ANs after node placement are perturbed. We refer to such experiments as global perturbation experiments. A perturbation constitutes a random directional displacement of the AN from its original position by a fixed distance $4r$. The fixed value of perturbation displacement $4r$, albeit in random direction, represents the experimental finding within the topology of our experiments introducing the most pronounced impact on network connectivity without completely partitioning the network. In each experiment and after applying perturbation, we test global connectivity.

We conduct our experiments in different field sizes but report sample results for a $200km \times 200km$ field. The set of pre-deployed clusters are distributed randomly following a uniform Poisson point process in the field of experiment. We set parameters r and R at $50m$ and $4550m$, respectively. The number of clusters varies from 20 to 160 by a step size of 20. Before reporting our results, we define a measure to quantify robustness. Equation (4.1) gives the definition of the measure referred to as robustness factor (RF). The RF measure not only takes into consideration the probability of staying connected after perturbation, but also the number of intermediate ANs used to establish connectivity.

$$RF = (Pr_{EGDO} - Pr_{SMT}) \times \frac{\eta_{SMT}}{\eta_{EGDO}} \tag{4.1}$$

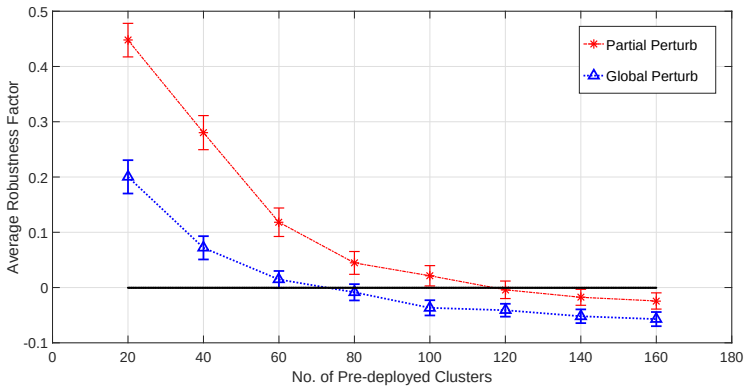


Figure 4.4: A drawing of average robustness factor as a function of the number of pre-deployed clusters in perturbation tests.

In Equation (4.1), Pr_{EGDO} and Pr_{SMT} represent the probabilities of remaining connected after perturbation is applied to the cases of EGDO and SMT algorithms, respectively. Accordingly, the calculation of Pr_{EGDO} in perturbation tests is described below. In each experiment, the global connectivity count is increased by one if the network remains connected after applying perturbation. The value of Pr_{EGDO} is identified by dividing the global connectivity count to the total number of experiments, which is 500 here. Similarly, Pr_{SMT} is identified. The numbers η_{EGDO} and η_{SMT} represent the number of intermediate ANs used to establish global connectivity in EGDO and SMT algorithms.

Because EGDO algorithm uses hexagons instead of circles, it generally covers a given distance along a line with a larger number of ANs than SMT. However, placing nodes towards the center of geometry within HCS offsets some of the impact. Generally speaking, the EGDO algorithm is observed to use a larger number of intermediate ANs than StaSMT. In return, it offers a higher level of robustness.

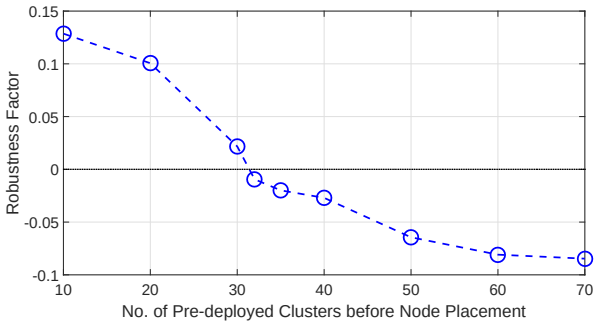
Experimental results of partial perturbation within 95% confidence intervals are shown by red line in Fig. 4.4. The horizontal axis shows the number of pre-deployed disconnected clusters before we apply any node placement algorithm. The vertical axis is the value of RF averaged over 500 different scenarios at each given number of pre-deployed clusters. We notice that the value of RF is in the range $[-1, 1]$ as two probability measures are within

$[0, 1]$ and the EGDO algorithms is expected to use a larger number of ANs than the SMT algorithm. A positive value of RF closer to 1 means that EGDO algorithm achieved much better robustness characteristics while using a relatively small number of ANs. An inspection of the reported results of Fig. 4.4 reveals that the EGDO algorithm shows a significant performance advantage in sparse networks. However and as the number of pre-deployed clusters increases, there is a threshold of cluster density beyond which EGDO algorithm will lose its advantage over SMT algorithm. More information about the threshold will be given in the next subsection.

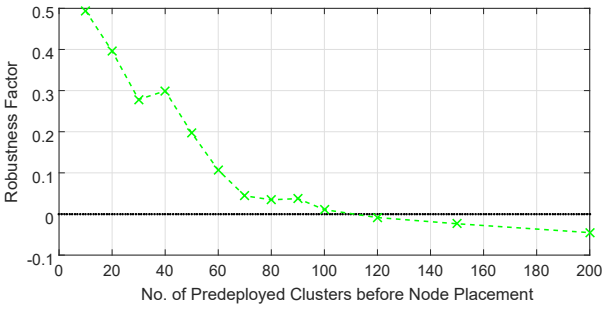
Besides partial perturbation tests, we also conduct global perturbation experiments. In these tests, we perturb the positions of pre-deployed AN gateway nodes as well as intermediate AN nodes. All ANs within the connected network graph are displaced along a random direction by an amplitude of $4r$. The value of RF is calculated in the same way as explained before. The test results within 95% confidence intervals are shown in Fig. 4.4 by the blue curve. Compared to partial perturbation test results, the RF values in global perturbation tests show a lower starting point and a faster drop rate as the density of clusters grows higher. The results show that the difference in perturbation robustness is very significant in some scenarios. Specifically, it is observed that the value of Pr_{EGDO} is one to two orders of magnitude larger than the value of Pr_{SMT} in some instances.

4.4 Inspection of Cluster Density Threshold Value

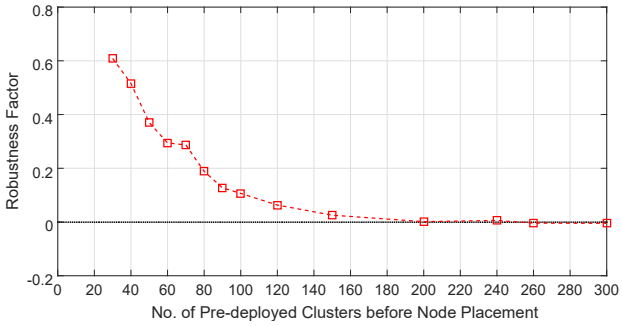
As described in the previous subsection, we observe a threshold of AN density beyond which the network can no longer be regarded as sparse. The threshold to which we refer as τ denotes a cluster density value passed which the EGDO algorithm offers no advantage compared to the SMT algorithm. In this subsection, we raise a hypothesis that the value of threshold τ is related to the density of ANs, namely, the field area divided by the total area of AN



(a) $100km \times 100km$



(b) $200km \times 200km$



(c) $300km \times 300km$

Figure 4.5: The identification of threshold τ for different field sizes.

coverage. We note that both SMT and EGDO algorithms seek to minimize the AN cost. Yet, the SMT algorithm attempts at reducing the total distance covered by ANs while the EGDO algorithm tries to reduce the AN overlap areas. In essence, minimizing the area of overlap is no longer meaningful when the AN density goes beyond a certain value. As cluster density grows, the average overlap area increases. Thus, the robustness of SMT algorithm will inherently improve and EGDO algorithm no longer offers any robustness advantage. To numerically validate this hypothesis, we conduct experiments on three different field sizes, of $100km \times 100km$, $200km \times 200km$, and $300km \times 300km$. We apply the partial perturbation test to each field and vary the number of pre-deployed clusters. The threshold value for each field size is identified as where the plots of RF versus AN cross the horizontal axis. Perturbation experiments are repeated 100 times in each scenario and for every number of clusters. Further, we test 100 different scenarios and report the average results. In Fig. 4.5a, Fig. 4.5b, and Fig. 4.5c, the RF curves approximately cross the x-axis at values of 32, 120,

and 260.

Table 4.1 records average intermediate AN cost for each given number of pre-deployed clusters in the test of the $100km \times 100km$ field. Table 4.2 and Table 4.3 show the AN cost in the tests of $200km \times 200km$ and $300km \times 300km$ field sizes, respectively. As described above, the threshold is defined as

$$\tau \propto \frac{\text{Field Area}}{\text{AN area} \times \text{No. of ANs}} \tag{4.2}$$

The threshold values τ_1, τ_2, and τ_3 are calculated below for $100km \times 100km$, $200km \times 200km$, and $300km \times 300km$ field size scenarios where c absorbs all constants.

$$\tau_1 \approx c \times \frac{10^{10}}{4550^2 \times (32 + 27.1)} = 0.017 \times \frac{10^{10}}{4550^2} c$$

$$\tau_2 \approx c \times \frac{4 \times 10^{10}}{4550^2 \times (120 + 102.7)} = 0.018 \times \frac{10^{10}}{4550^2} c$$

$$\tau_3 \approx c \times \frac{9 \times 10^{10}}{4550^2 \times (260 + 227.2)} = 0.018 \times \frac{10^{10}}{4550^2} c$$

From the calculations, the values of τ_1, τ_2, and τ_3 are all around to $0.018c$. While not reported here, we have observed similar patterns with different values of r, R, and field sizes. The results numerically support our hypothesis that the value of threshold τ_i is related to the ratio of the field area and the total area covered by ANs.

4.5 An Comparison of SMT and EGDO in HCS

Since EGDO algorithm utilizes hexagonal tiles instead of radial disks to model the range of advantaged nodes, one can raise the question as to what happens when applying SMT algorithm to a network using hexagonal tiling. In order to answer this question, we run an

No. of Clusters	EGDO AN cost	SMT AN Cost
10	20.2	18.1
20	26.6	24.0
30	29.2	26.1
32	**29.7**	**27.1**
35	30.3	27.3
40	30.9	28.6
50	30.9	28.2
60	30.5	28.0
70	30.2	28.4
80	29.5	27.2

Table 4.1: Average AN cost in $100km \times 100km$ field test.

No. of Clusters	EGDO AN cost	SMT AN Cost
10	46.5	41.6
20	63.7	56.9
30	74.8	67.4
40	84.1	75.5
50	91.4	81.9
60	96.0	86.2
70	100.4	90.8
80	103.9	93.8
90	105.2	95.2
100	109.0	99.8
120	**112.9**	**102.7**
150	116.3	106.4
200	117.9	109.1

Table 4.2: Average AN cost in $200km \times 200km$ field test.

additional experiment.

Our experimental setting is described as follows. Within an area of $120km \times 120km$, we randomly deploy a number of clusters ranging from 50 to 500 at an increasing step size of 50. In this experiment, we set r and R at $30m$ and $2730m$, respectively. For each fixed number of pre-deployed clusters, we run 100 different randomly distributed scenarios. Then, we average the number of ANs to report our results. Fig. 4.6 compares the AN cost of establishing connected graphs, through SMT and EGDO algorithms with the same level of built-in robustness, as a function of the number of pre-deployed clusters. In this setting, the

No. of Clusters	EGDO AN cost	SMT AN Cost
30	127.8	120.7
40	136.5	121.6
50	148.7	133.3
60	161.8	145.3
70	170.6	153.0
80	179.0	159.8
90	187.0	169.3
100	193.2	173.5
120	205.4	184.8
150	218.3	197.0
200	233.8	211.3
240	244.3	222.3
260	**247.3**	**227.2**
300	252.1	231.6

Table 4.3: Average AN cost in a $300km \times 300km$ field test.

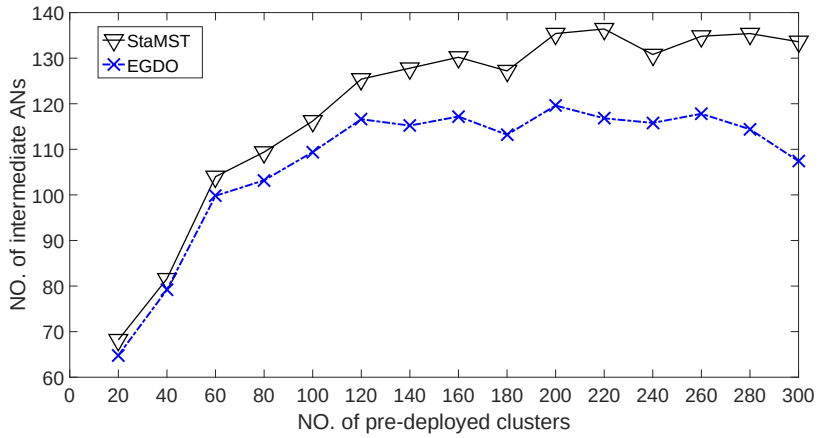


Figure 4.6: An AN cost comparison of SMT and EGDO algorithms in HCS.

network is no longer considered sparse once the number of pre-deployed clusters reaches 250.

It can be observed from the results that EGDO algorithm performs slightly better when the number of clusters is small. As the number of pre-deployed clusters grows, the EGDO algorithm intends to use even a smaller number of ANs than the SMT algorithm to establish full connectivity. The number of ANs used by the EGDO algorithm is typically 10% to 20% less than those used by the SMT algorithm for as long as the network is sparse, i.e., the number of pre-deployed clusters is less than 250. Interestingly, the AN cost advantage of

the EGDO algorithm becomes even more apparent for a dense network with more than 250 pre-deployed clusters. However, the advantages of EGDO over SMT in dense networks are not of high significance because a dense network naturally offers robustness.

While not shown here, it is also important to note that representing the communication range of an AN with a reduced radius circle or a reduced edge square in CCS leads to utilizing an increased number of ANs in establishing connectivity.

Chapter 5

Maintaining Connectivity in Mobile Network

5.1 Analysis of Mobility Bound under Small Scale Node Mobility

In this section, we quantitatively analyze the built-in robustness of the HCS approach of [39] by allowing random omni-directional mobility of nodes in set V_0 at small scales. We geometrically analyze the maximum mobility distance of an edge resulting in disconnecting the network.

In [39], we assume the communication radius of a gateway node is $2R$ and that of a standard node is r where R is typically two orders of magnitude larger than r. As shown by Fig. 5.1, the robust communication range of a gateway node is represented by a large hexagon with an edge length of $R = (12n + 7)r$ where n is a positive integer. The integer n is chosen to facilitate HCS set-up without loss of generality. Standard nodes in a small hexagon cell are

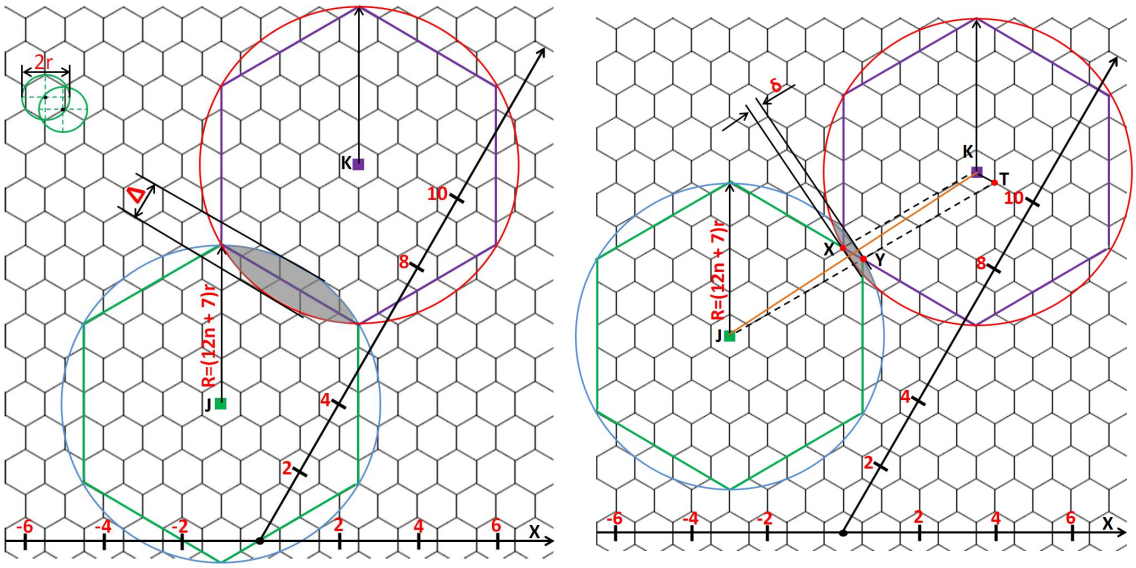


Figure 5.1: The cases of coverage overlap in which the overlapping area between two connected nodes is (a) the largest, and (b) the smallest.

connected to their gateway node located at the center of that cell. In this work, the word 'node' denotes the gateway node whose communication radius is $2R$. A pair of nodes are connected if their distance is less than $2R$.

A pair of connected nodes in HCS typically have a common edge. In reality, the communication ranges follow the disk format and would hence overlap as illustrated by Fig. 5.1. For a pair of connected nodes in HCS, the overlapping area between the two disks functions as a safety margin. Fig. 5.1a and Fig. 5.1b show cases in which the largest and the smallest safety margin takes place, respectively. When a node becomes mobile and the direction of movement is unknown, the safety margin between this node and its neighbors may decrease or increase. We are not interested in scenarios in which mobility enhances safety margin. Rather, we take interest in those cases for which safety margin reduces. The worst effect of the latter cases is materialized when the center-to-center distance between two nodes increases.

When two nodes are connected, the safety margin can accommodate mobility at a distance

of Δ in the case of the largest safety margin and δ in the case of the smallest safety margin. The value of Δ is given below. The calculation is geometrically straightforward and the details are omitted here.

$$\Delta = 2(1 - \frac{\sqrt{3}}{2})R$$

The following theorem places an upper bound on the value δ.

Theorem 5.1. *In the case of the smallest safety margin, an edge can tolerate a maximum mobility distance of $\delta = r/2$ for any given value of n satisfying $R = (12n + 7)r$.*

The proof of Theorem 5.1 is given in Appendix B.2.

For a connected pair of nodes, the margin of connectivity typically falls in the range of $[\delta, \Delta]$. Assuming all safety margins in a connected network are approximately uniformly distributed, the perturbation bound Γ has to be less than or equal to the expected value of the margin. This implies

$$\Gamma \leq \frac{\delta + \Delta}{2} \tag{5.1}$$

Plugging in the values, we have

$$\begin{aligned}
&\Gamma \leq 0.134(12n + 7)r + \frac{r}{4} \\
&\Gamma \leq 7.6r \qquad \text{when} \quad n = 4 \\
&\Gamma \leq 9.2r \qquad \text{when} \quad n = 5 \\
&\Gamma \leq 10.8r \qquad \text{when} \quad n = 6 \\
&\Gamma \leq 12r \qquad \text{when} \quad n = 7 \\
&\quad \ldots
\end{aligned} \tag{5.2}$$

The above analysis establishes a theoretical estimate of perturbation bound due to node mobility in a connected network.

5.2 Connectivity Maintenance under Large Scale Node Mobility

In the previous section, we assumed small scale mobility of pre-deployed clusters and showed how the embedded safety margin of HCS model can preserve connectivity. In this section, we study a case in which the mobility of nodes in set V_0 is at a larger scale exceeding the given thresholds of (5.2). Let $p_i \in \mathbb{R}^2$ denote the position of $v_i \in V_0$ and $q_j \in \mathbb{R}^2$ denote the position of $v_j \in V_1$. When the mobility bounds of (5.2) are exceeded, we consider relocating the existing nodes within set V_1 in order to restore connectivity.

5.2.1 Graph Theoretic Analysis

We first introduce the general notations used throughout this chapter of the dissertation. Given a graph $G(V, E)$ with n nodes where V is the set of nodes and E is the set of edges, we denote $v_i \in V$ as a node in G and $(v_i, v_j) \in E$ as the edge between v_i and v_j. In this work, we only consider non-directional graphs in which $(v_i, v_j) = (v_j, v_i)$. Utilizing the distance connectivity model with $p_i \in \mathbb{R}^2$ representing the position of $v_i \in V$, $(v_i, v_j) \in E$, nodes i and j are connected if $\|p_i - p_j\| \leq 2R$ where $2R$ is the communication range of the nodes. Here, we pick $2R$ because we can geometrically use two touching disks of the radius R to represent a pair of connected nodes. Let the symmetric matrix $A \in \mathbb{R}^{n \times n}$ denote the adjacency matrix

of G, and a_{ij} denote the element $\{i, j\}$ in A. Then,

$$a_{ij} = \begin{cases} 1, & \text{if } (v_i, v_j) \in E \\ 0, & \text{otherwise} \end{cases} \tag{5.3}$$

Matrix $D = \mathbf{diag}\{d_1, d_2, \ldots, d_n\}$ is the degree matrix of graph G where $d_i = \sum_{j=1}^{n} a_{ij}$ denotes the degree of v_i. The Laplacian matrix is then defined as $L = D - A$. Further, S_+^n represents the set of symmetric positive semidefinite (PSD) matrices of dimension n. Being a real symmetric matrix, L carries the following properties.

1. Due to its real symmetry, all eigenvalues of L denoted by $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$, are real numbers.

2. According to Gershgorin's theorem, all eigenvalues of L are non-negative. Consequently, $L \in S_+^n$ has a complete orthonormal eigenspace referred to as $\{u_1, u_2, \ldots, u_n\}$.

3. Since $L\mathbf{1} \equiv \mathbf{0}$, $\lambda_1 = 0$ and vector $\mathbf{1}$ are an eigenvalue-eigenvector pair of any Laplacian matrix. Here $\mathbf{1}$ is the vector of all ones and $\mathbf{0}$ is a vector of all zeros.

4. $\forall u_i$ (given $u_1 = \mathbf{1}$), we have $u_i^T u_j = 0$ for $i \neq j$) and $u_i^T u_i = 1$. In particular, for $i \geq 2$, $u_i \in \mathbf{1}^\perp$.

5. The second smallest eigenvalue, λ_2 is known as 'the algebraic connectivity' of a graph. The graph is connected if and only f $\lambda_2 > 0$. Furthermore, the number of zero eigenvalues of L equals to the number of connected components in graph G.

6. Finally, λ_2 can be calculated through Raleigh quotient

$$\lambda_2 = u_2^T L u_2 = \min_{u \in \mathbf{1}^\perp, u \neq \mathbf{0}} \frac{u^T L u}{u^T u}$$

The proofs of these properties can be found in [78]. From property 5 and 6, we see that $\lambda_2 > 0$ is the condition of graph connectivity. However, verifying this condition has a high computational complexity associated with minimizing Raleigh quotient. The following lemma provides a lower complexity alternative.

Lemma 1. *The eigenvalue $\lambda_2 > 0$ if and only if $\exists W$ such that $W^T L W$ is positive definite where $W = [w_1, w_2, \ldots, w_{n-1}] \in \mathbb{R}^{n\times(n-1)}$ with $w_i^T \mathbf{1} = 0$ $\forall i \in \{1, 2, \ldots, n-1\}$ and $w_i^T w_j = 0$ $\forall i \neq j$.*

Appendix B.1 includes the proof of Lemma 1.

In order to describe the change of topology when nodes become mobile, we introduce the augmented Laplacian matrix. We consider two types of nodes. Based on the assumptions made in previous sections, we e can place intermediate nodes in set V_1 but not pre-deployed nodes in set V_0.

Suppose we have a graph $G(V_0 \cup V_1, E)$ at time $(k-1)$ with M pre-deployed nodes and N intermediate nodes, where $M = |V_0|$ and $N = |V_1|$. Without loss of generality, assume nodes in V_0 are numbered from 1 to M and nodes in V_1 from $M+1$ to $M+N$. This assumption is justified because as long as two graphs are isomorphic, their matrix representation can be switched back and forth through permutation transformation [79].

Let matrix $P \in \mathbb{R}^{M\times 2}$ represent the positions of nodes in set V_0. The i-th row of P contains the 2-D Euclidean coordinates of node $v_i \in V_0$. Similarly, the positions of intermediate nodes are contained in $Q \in \mathbb{R}^{M\times 2}$. Let $L_0 \in \mathbb{R}^{M\times M}$ denote the Laplacian graph among nodes in V_0. In general, L_0 represents a disconnected graph, yet it doesn't imply all nodes in V_0 are isolated. From the algebraic graph theory point of view, establishing a connected network among the nodes of $V_0 \cup V_1$ can be done by augmenting L_0 to a higher dimension Laplacian matrix $L \in \mathbb{R}^{(M+N)\times(M+N)}$ whose nullity is 1.

To augment L_0 and include nodes of V_1 in the graph, we define matrix $C \in \mathbb{R}^{M\times N}$ that contains binary variables c_{ij} with $i \in \{1, 2, \ldots, M\}$ and $j \in \{1, 2, \ldots, N\}$. Each c_{ij} is associated with the edge (v_i, v_j) where $v_i \in V_0$ and $v_j \in V_1$. We also define symmetric matrix $F \in \mathbb{R}^{N\times N}$ whose off-diagonal elements are $\{0, -1\}$. For convenience, we denote the off-diagonal elements as $-f_{ij}$ $(i > j)$ where $i, j \in \{M+1, M+2, \ldots, M+N\}$. Consequently, each f_{ij} has a binary value indicating the edge (v_i, v_j) exists where $v_i, v_j \in V_1$.

In a dynamic network, an edge between two nodes could possibly appear and disappear from time to time. Edge additions and deletions are represented by binary variables $\{c_{ij}, f_{ij}\}$ and incidence matrix. Let us assume the Laplacian matrix of graph $G(V, E)$ at time $(k-1)$ is $L(k-1)$. If an edge appears between v_i and v_j at t, the Laplacian matrix will be updated as $L(k) = L(k-1) + h_{ij}^T h_{ij}$ where $h_{ij} = [0, \ldots, 1, 0, \ldots, -1, 0, \ldots](i \neq j)$ with 1 appearing at the i-th position and -1 at the j-th position. Conversely, $L(k) = L(k-1) - h_{ij}^T h_{ij}$, if the edge (v_i, v_j) is deleted from the graph. With these assumptions and notations, the augmented Laplacian matrix of graph $G(V_0 \cup V_1, E)$ is expressed as

$$\begin{aligned} L &= \left[\begin{array}{c|c} L_0 & 0 \\ \hline 0 & 0 \end{array}\right] + \sum c_{ij} h_{ij}^T h_{ij} + \sum f_{ij} h_{ij}^T h_{ij} \\ &= \begin{bmatrix} L_0 + D_c & -C \\ -C^T & F \end{bmatrix} \end{aligned} \tag{5.4}$$

We assume the graph is connected at $k = 0$, i.e., Laplacian $L(k)$ has $\lambda_2 > 0$ but the nodes in set V_0 become mobile for $k > 0$. Our goal is then to maintain $\lambda_2(L(k)) > 0$ for $\forall k > 0$ by controlling decision variables, i.e., the elements of C, F, and Q.

5.2.2 Problem Statement

Eq. (5.4) provides a description of the topology change as well as a measure of network connectivity $\lambda_2(L(k))$ when the nodes in set V_0 become mobile. Hence, we have to relate the topology change to the new positions of the nodes in set V_1. A binary variable c_{ij} indicates the existence of an edge (v_i, v_j) according to our connectivity model, i.e., $(v_i, v_j) \in E$ implies that the distance between v_i and v_j is less than $2R$ or $\|p_i - q_j\| \leq 2R$. Similarly, $f_{ij} = 1$ implies $\|q_i - q_j\| \leq 2R$. With these conditions, we pose the following feasibility problem solving which results in finding the new positions of nodes of V_1 according to the changes made in positions of nodes of V_0.

Problem 5.1.

$$\min_{C,F,Q} \quad 0 \tag{5.5}$$

$$S.T. \quad W^T L W \succ 0 \tag{5.6}$$

$$\forall c_{ij}, p_i, q_j, \quad c_{ij} \cdot \|p_i - q_j\| \leq 2R \tag{5.7}$$

$$\forall f_{ij}, q_i, q_j, \quad f_{ij} \cdot \|q_i - q_j\| \leq 2R \tag{5.8}$$

$$2 \leq f_{jj} \leq 5 \tag{5.9}$$

$$c_{ij} \in \{0, 1\} \tag{5.10}$$

$$f_{ij} \in \{0, 1\} \tag{5.11}$$

$$\forall j \in \{1, 2, \ldots, N\} \quad \& \quad \forall i \in \{1, 2, \ldots, M\}$$

The first constraint guarantees the algebraic connectivity based on Lemma 1. Here, matrix W can be easily found by calculating the orthogonal space of $\mathbf{1} \in \mathbb{R}^{(M+N)\times 1}$. The second and third constraint groups are imposed by the the connectivity model. If an edge $(v_i, v_j) \in E$,

$c_{ij} = 1$ (or $f_{ij} = 1$), then the Euclidean distance between node v_i and v_j has to be less than $2R$. The fourth constraint is the result of considering the following two facts. First, the degree of any intermediate node is larger than 2 because no intermediate node can be a terminal node in the graph. Second, our goal is to reestablish connectivity while keeping the number of existing intermediate nodes fixed. Hence, we need to construct a spanning tree for the graph using the existing intermediate nodes. According to Lin [9], the maximum node degree in a minimum spanning tree is 5.

5.2.3 Solution Alternatives

We open this subsection by providing the following theorem showing that the connectivity maintenance problem formulated in the previous section is NP-hard.

Theorem 5.2. *Problem 5.1 is NP-hard.*

Appendix B.3 contains the proof of Theorem 5.2.

Next, the following lemma simplifies the two groups of nonlinear Euclidean norm constraints in Problem 5.1 to linear matrix inequalities (LMIs).

Theorem 5.3. *Constraint (5.7) is equivalent to a following LMI, given the large positive number* Ω*.*

$$\begin{bmatrix} 4R^2 + \Omega(1 - c_{ij}) & (p_i - q_j) \\ (p_i - q_j)^T & I_{2\times 2} \end{bmatrix} \succeq 0 \qquad (5.12)$$
$$\forall j \in \{1, 2, \ldots, N\} \quad \& \quad \forall i \in \{1, 2, \ldots, M\}$$

Appendix B.4 contains the proof of Theorem 5.3.

Hence, Problem 5.1 can be reduced to to a mixed integer semi-definite programming (MISDP) as the result of eliminating nonlinear constraints. Utilizing the optimization tool YALMIP [80] to apply branch and bound (BnB) algorithm along with SDP solver MOSEK [81] to lower the bounds of BnB iterations, we can then solve MISDP. Our experimental results show that solving Problem 5.1 becomes computationally prohibitive when the number of nodes exceeds 10. Theoretically, the worst case time complexity is

$$\mathcal{O}(2^{MN} \cdot \sqrt{2}^{N^2-N})$$

occuring when all permutations of MN binary variables c_{ij} and $(N^2 - N)/2$ binary variables f_{ij} have to be tried.

We also utilize an alternative solution approach by relaxing binary constraints with continuous constraints $c_{ij} \leq c_{ij}^2$ and $f_{ij} \leq f_{ij}^2$ in the range of $[0, 1]$. We then use PENLAB [82] nonlinear SDP solver to disguise binary constraints. In this case, PENLAB solver is able to generate acceptable results for network configurations with no more than 10 nodes. In [83, 84, 85], the authors use cutting plane algorithms to solve their proposed MISDP problems. As a third alternative, we apply a similar cutting plane solution to our problem but note that this method is only able to solve our problem when the number of nodes is less than 20. Hence, we conclude that there is a need to develop alternative algorithms to solve this problem for larger networks in practical scenarios.

5.3 Location-Aware Connectivity Maintenance under Large Scale Node Mobility

In this section, we focus on solving Problem (5.1) in a practical case for which the positions of network nodes are known. We show that in this case Problem 5.1 becomes a second order cone programming (SOCP) problem and hence can be solved in polynomial time. We note that this location-aware connectivity maintenance scenario is practically viable considering the fact that network nodes can share their locations utilizing their GPS coordinates.

5.3.1 Problem Formulation

Recall that in our problem, the nodes of set V_0 become mobile after network connectivity is initially established. In this subsection, we consider a scenario in which we react to large scale mobility of one node in set V_0 at a time. Note that this scenario consideration does not lead to loss of generality if the nodes of V_1 are relocated fast enough to adapt to network topology changes. Practically speaking, only one node is disconnected from the entire network, as long as the relocation of nodes in set V_1 can occur relatively faster than the rate of change in the locations of nodes in V_0. In this case, the network remains connected among discrete mobility instances representing changes in the locations of nodes in set V_0.

In this case, the topology variation lies on the connectivity condition between this single mobile node referred to as $v_s \in V_0$ and intermediate nodes in V_1. The corresponding part in the augmented Laplacian matrix defined in Eq. (5.4) is row s of matrix C, i.e., $[c_{s1}, c_{s2}, \cdots, c_{sN}]$. Since node locations are known, the mobile node v_s is able to identify the closest node v_d to which it should connect in its new position. Denote $L(k)$ the Laplacian of the topology at discrete mobility instance k. Then, we can input the corresponding binary variables in $L(k)$ to Problem 5.1 which is now reduced to the following SOCP problem.

Problem 5.2.

$$\min_{Q(k)} \quad Tr((Q(k)-Q(k-1))(Q(k)-Q(k-1))^T) \tag{5.13}$$

$$S.T. \qquad \begin{bmatrix} 4R^2+\Omega(1-c_{ij}) & (p_i-q_j) \\ (p_i-q_j)^T & I_{2\times 2} \end{bmatrix} \succeq 0 \tag{5.14}$$

$$\begin{bmatrix} 4R^2+\Omega(1-f_{ij}) & (q_i-q_j) \\ (q_i-q_j)^T & I_{2\times 2} \end{bmatrix} \succeq 0 \tag{5.15}$$

$$\forall j \in \{1,2,\ldots,N\} \quad \& \quad \forall i \in \{1,2,\ldots,M\}$$

Here $Q(k)$ and $Q(k-1)$ represent the positions of intermediate nodes at instances k and $k-1$, respectively. The purpose of the objective function (5.13) is to minimize the total distance traversed by the intermediate nodes so as to minimize the energy cost and reaction time. As noted, this problem is a second order cone programming with $2N$ variables which can be solved through interior point method [86]. As a good solution alternative, one can invoke MOSEK solver under YALMIP environment to solve the problem. Algorithm 4 contains the description of our adaptive node relocation algorithm proposed to solve location-aware connectivity maintenance under large scale mobility.

The detailed description of Algorithm 4 is provided next. The algorithm is in essence a continuous loop in discrete mobility instance k. The mobility instance k is increased as the result of node movement. Per line 5, the algorithm takes action if the network is disconnected after a node moves. The algorithm proceeds with inputting the Laplacian matrix at instance $k-1$ and location of v_s at instance k. Next, connected component sets are formed as the result of v_s moving. The five lines starting from line 9 call for solving Problem 5.2 after identifying the closest node v_d to v_s and forming the resulting Laplacian matrix. These

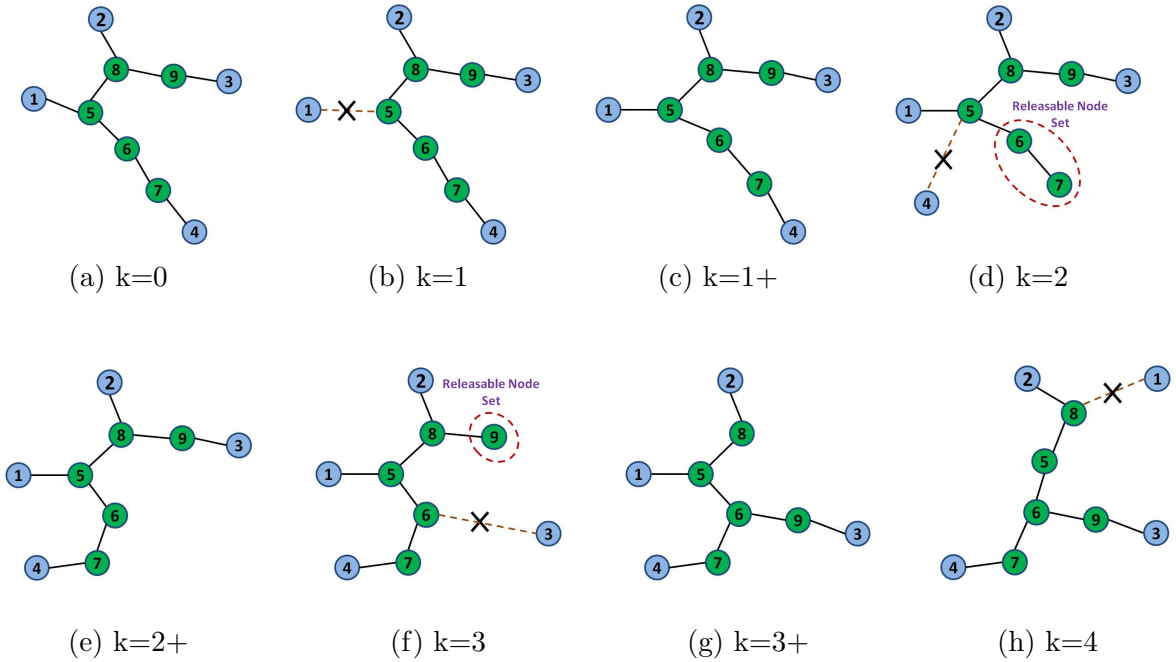
Algorithm 4 Adaptive Node Relocation Algorithm

Set mobility instance counter $k = 0$
Form fully connected component Υ_0
while (TRUE) **do**
  Set $k = k + 1$
  if (disconnected) **then**
    Input $L(k-1)$ and location of v_s at instance k
    Form connected component sets Υ_i with
      $v_s \in \Upsilon_1$ and $\Upsilon_0 = \bigcup_i \Upsilon_i$
    Identify $v_d \in \Upsilon_j$ $(j > 1)$ closest to $v_s \in \Upsilon_1$
    Form $L(k)$
    Solve Problem 5.2 with $L(k)$
    if (connected) **then**
      Set coordinates of nodes of $Q(k)$
    else if (Releasable node set $N_R \neq \emptyset$) **then**
      Change $L(k)$ according to N_R
      Solve Problem 5.2 with $L(k)$
      if (connected) **then**
        Set coordinates of nodes of $Q(k)$
      end if
    else
      Unable to restore connectivity
      Set coordinates of nodes of $Q(k)$ to -1
    end if
    Output coordinates of nodes of $Q(k)$
  end if /* if (disconnected) */
end while

lines inspect the possibility of directly connecting v_s to v_d by directly relocating v_d while still containing it within its connected component. If connectivity cannot be restored, the six lines starting from line 14 attempt at restoring connectivity by relocating the nodes of releasable node set N_R to form a line graph between v_s and v_d . The result is verified after changing $L(k)$ accordingly and solving Problem 5.2. The next four lines starting at line 20 indicate that connectivity cannot be restored after having made the two attempts above. Line 18 and 22 set the coordinates of nodes of $Q(k)$ according to the outcome of one of the three scenarios above. Finally, line 24 outputs the coordinates of nodes $Q(k)$ while the following two lines go to the the top of while loop awaiting the next instance of mobility.

Fig. 5.2 illustrates the functionality of Algorithm 4 in four consecutive discrete mobility instances. Fig. 5.2a shows a connected network at $k = 0$. Fig. 5.2b illustrates an event at $k = 1$ in which $v_s = v_1$ gets isolated from other nodes with $v_d = v_5$. Fig. 5.2c shows that at $k = 1+$ connectivity is restored by relocating node v_5 and forming edge (v_1, v_5) after solving Problem 5.2 (line 11 of the algorithm). Fig. 5.2d then shows another event at $k = 2$ in which $v_s = v_4$ moves away and becomes disconnected from the network. However, relocating node $v_d = v_5$ and forming edge (v_4, v_5) while containing v_5 within its connected component is not an option. At this instance, $N_R = \{v_6, v_7\}$. Fig. 5.2e illustrates that connectivity is restored at $k = 2+$ after relocating the nodes of N_R, i.e., forming a revised $L(k)$ associated with the line graph of (v_5, v_6, v_7, v_4) and solving Problem 5.2. Fig. 5.2f shows a third event at $k = 3$ in which $v_s = v_3$ moves and becomes disconnected from the network. Again, relocating node $v_d = v_6$ and forming edge (v_3, v_6) while containing v_6 within its connected component is not an option. Fig. 5.2g illustrates that connectivity is restored at $k = 3+$ after relocating $v_9 \in N_R$, i.e., forming a revised $L(k)$ associated with the line graph of v_6, v_9, v_3 and solving Problem 5.2. Finally, Fig. 5.2h shows a fourth event at $t = 4$ in which $v_s = v_1$ moves to the upper right corner but neither the edge (v_1, v_8) nor the empty set N_R can be used to reconnect v_1 to the network. At this point, it is possible that further mobility of other nodes at later instances can help restore connectivity using Algorithm 4 or else EGDO algorithm

has to be applied to reconnect the network.



(a) k=0 (b) k=1 (c) k=1+ (d) k=2

(e) k=2+ (f) k=3 (g) k=3+ (h) k=4

Figure 5.2: Sample illustrations describing the functionality of Algorithm 4 in four consecutive discrete mobility instances. The symbol + after a mobility instance denotes the action instance for that mobility instance.

5.3.2 Complexity Analysis

In this subsection, we analyze the complexity of Algorithm 4. In [87, 88], the authors point out that the time complexity for solving SOCP with κ conic constraints, such as (5.14) and (5.15) is in the order of $\mathcal{O}(\sqrt{\kappa})$. Each conic constraint corresponds to an edge connected to at least one intermediate node in the network graph. Therefore, the number of conic constraints is equal to the trace of matrix F in Eq. (5.4). The following theorem finds an upper bound for $Tr(F)$.

Theorem 5.4. *Given a network graph with M pre-deployed clusters and N intermediate nodes, suppose the primitive graph consisting of only M pre-deployed nodes has a Laplacian matrix L_0 with rank$(L_0) = r$. Then, the trace of matrix F in the augmented Laplacian matrix*

L, defined by Eq. (5.4), satisfies the following inequality.

$$Tr(F) \leq M - r + 2N - 2 \tag{5.16}$$

Appendix B.5 includes the proof of Theorem 5.4.

Theorem 5.4 provides us with the maximum number of conic constraints of Problem 5.2. Therefore, the time complexity of solving Problem 5.2, i.e., executing line 11 or line 16 of Algorithm 4, is upper-bounded by

$$\mathcal{O}(\sqrt{M - r + 2N}) \tag{5.17}$$

The complexity of lines $7 - 8$ mainly depends on calculating connected components and line search of the closet nodes. The process of calculating connected components in a graph, undergoing depth first search algorithm, has a time complexity of $\mathcal{O}(|V|+|E|)$. When dealing with a spanning tree graph, the number of edges is in the same order as that of the number of nodes on the tree, i.e., $\mathcal{O}(M + N)$. The highest complexity of searching for the closest node in set V_1 is materialized when it has to traverse the entire set V_1. Therefore, line 9 has a runtime in the order of $\mathcal{O}(N)$. Line 10 mainly assigns numbers to some of c_{ij} and f_{ij} values. Such assignment is made with a complexity in the order of constant time. The worst case time complexity of line 14 in Algorithm 4 is in the order of constant time, as the releasable intermediate nodes are all of degree 2 or lower. Assuming intermediate nodes are distributed uniformly on each edge of the primitive spanning tree consisting of only pre-deployed nodes, this constant is bounded by $N/(M - 1)$. Counting time complexity of all these steps, the total complexity of Algorithm 4 is then in the following order.

$$\mathcal{O}(M + 2N + \sqrt{M - r + 2N}) \tag{5.18}$$

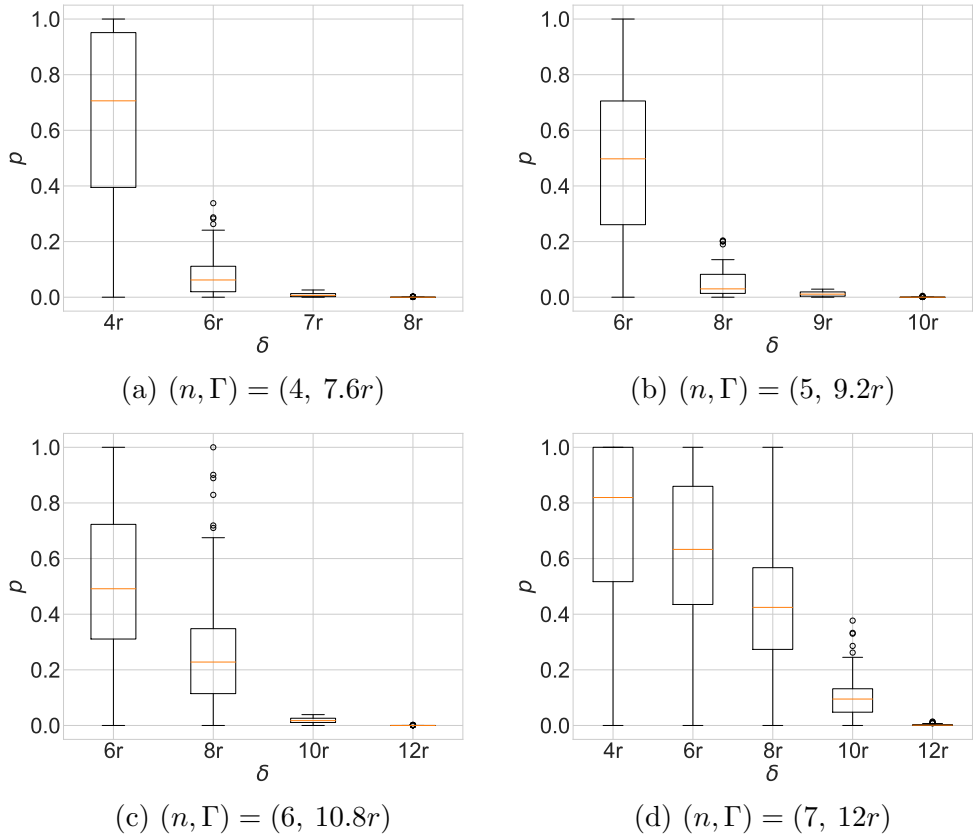
Chapter 6

Simulation Results of Connectivity Maintenance Problem

6.1 Experimental Results

6.1.1 Small Scale Perturbation

The following set of experiments aim at verifying small scale mobility bounds derived in Section 5.1. For each experiment, different mobility distances are applied until reaching the threshold value rendering network disconnected. The results are shown in Fig. 6.1. In each graph, the horizontal axis is the mobility distance measured as a multiple of r and the vertical axis shows the probability of network staying connected after mobility perturbed is applied. The latter is referred to as 'perturbation survivability' and denoted as p. The value of p is calculated by repeatedly applying random-directional mobility at a given distance to pre-deployed nodes and then verifying connectivity. In each perturbation test, the test result is increased by 1 if the network is still connected. In our experiment, we repeat the process



(a) $(n, \Gamma) = (4,\ 7.6r)$ (b) $(n, \Gamma) = (5,\ 9.2r)$

(c) $(n, \Gamma) = (6,\ 10.8r)$ (d) $(n, \Gamma) = (7,\ 12r)$

Figure 6.1: An experimental comparison of perturbation survivability p as a function of mobility distance σ for different combinations of (Γ, n).

1000 times for each connected network and a given mobility distance. Then, we derive the value of p as the number of connected perturbed cases divided by 1000. For each mobility distance, we run the test on 100 different network scenarios and record the data of p in a box plot. Since perturbation survivability of a connected network is also related to node densities, we keep the number of pre-deployed nodes at 30 and vary the field size in order to adapt the density change due to changes in the value of n. This allows us to keep network sparsities and densities comparable in different experiments. The series of experiments start with $n = 4$ and end with $n = 7$. The value of r is set to $30m$. For every given value of n, we gradually increase mobility distance until p becomes zero reflecting that the mobility bound of the underlying network is reached. As observed, simulation results match the analysis of Section 5.1 and particularly Eq. (5.2).

In Fig. 6.1a, the distribution of p for 100 different network scenarios is depicted within each box plot associated with a mobility distance value on the $x-$axis. We can see when the mobility distance is at $4r$, more than half of network scenarios can survive perturbations. When mobility distance increases, the box plot of p shifts down to zero. When the mobility distance is at $7r$, most of the network scenarios cannot survive perturbations. For a mobility distance of $8r$, the distribution of p goes to 0. The results imply that the mobility bound of a network with $R = 55r$ is located between $7r$ and $8r$, while the analytical estimate is $7.6r$ in Eq. (5.2). Following a similar approach for the case of $n = 5$, Fig. 6.1b shows that the point at which p goes to 0 lies between $9r$ and $10r$ matching the estimate of $9.2r$. Fig. 6.1c and Fig. 6.1d show mobility bound values closely approximating the analytical findings of $10.8r$ and $12r$ for the cases of $n = 6$ and $n = 7$, respectively.

6.1.2 Large Scale Node Mobility

In this subsection, we provide numerical evaluation results of connectivity restoration algorithm reported in Section 5.3. The algorithm is evaluated through an experiment consisting of three steps. First and for a given number of nodes in set V_0, we establish connectivity by initially placing intermediate nodes of set V_1 using GDO algorithm. Second, we let a randomly chosen node in V_0 move in a random direction by a certain distance and then apply Algorithm 4 to restore connectivity if the network is disconnected. If connectivity is successfully restored, we count this event as a 'success' event. For each given mobility distance, we randomly generate a set of networks, apply mobility, restore connectivity, and count the number of success events. We define 'Recovery Probability' as the percentage of network success events in an experiment. In the figures below, mobility distance is referred to as σ, the cardinality of set V_0, i.e., the number of pre-deployed nodes is referred to as M, and the cardinality of set V_1, the number of intermediate nodes is referred to as N.

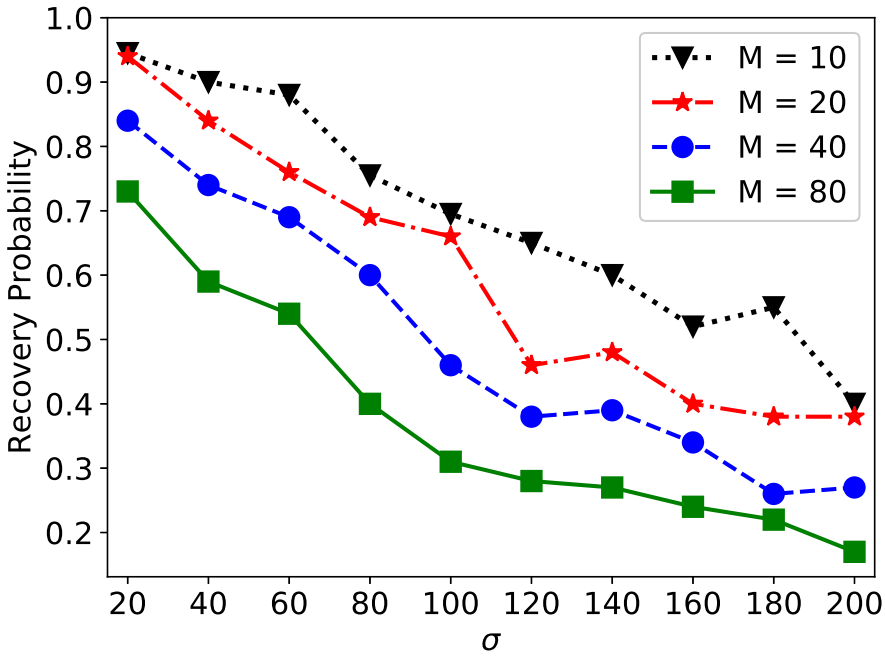


Figure 6.2: Plots of Recovery Probability as a function of mobility distance σ for different numbers of pre-deployed nodes M.

The effect of mobility distance (σ)

Here, we report the effect of mobility distance σ on the Recovery Probability of Algorithm 4. The experiment is done in a field of $10^4 \times 10^4 m^2$. The communication range of a standard node is set to $r = 5m$ and that of a gateway node is set to $R = 455m$. For a given network scenario with M randomly deployed nodes in set V_0, connectivity is first established using GDO algorithm. Then, a pre-deployed node is randomly chosen to become mobile. This procedure is repeated until node mobility results in having a disconnected network. Once we have a disconnected network, Algorithm 4 is applied. In each set of such experiments, M is fixed and 100 different network scenarios are tested to evaluate Recovery Probability. In this case, the value of σ grows from $20r$ to $200r$ within a range beyond the small scale mobility bounds reported in Section 6.1.1.

Fig. 6.2 shows the relationship between Recovery Probability and mobility distance. Each

curve represents a set of experiments with a fixed number of pre-deployed nodes M. It is as expected that the Recovery Probability drops as mobility distance grows. One may also notice that Recovery Probability decreases as M increases. The latter is attributed to the sparsity of network and further investigation results are provided in the following experiment.

The effect of the number of pre-deployed nodes (M)

Here, we present the relationship between the number of pre-deployed nodes and Recovery Probability. In this experiment, we set $d_m = 60r$ and increase M from 20 to 140. Following the same approach as the previous subsection, connectivity is initially established using GDO algorithm and a randomly selected node is made mobile for the purpose of making the network disconnected. Then, Algorithm 4 is launched to restore connectivity.

The results shown in Fig. 6.3 imply that as M increases, Recovery Probability decreases. Noting that all nodes are located in a fixed area field, a smaller number of intermediate nodes are initially required when there are a larger number of pre-deployed nodes. However, a larger number of intermediate nodes offer finer granular controllability of network topology in the case or restoring connectivity. Our statement is verified by Fig. 6.4 showing the number of pre-deployed nodes and the average number of intermediate nodes in the experiments of Fig. 6.3. In Fig. 6.4, the red bar shows that the number of intermediate nodes constantly decreases as the network becomes denser. These experimental results reflect that Algorithm 4 works better in sparse networks.

Runtime evaluations

In Section 5.3.2, we showed that the runtime of Algorithm 4 is in the order of $\mathcal{O}(M + 2N + \sqrt{M - r + 2N})$. Here, we numerically evaluate the runtime and compare the findings with the theoretical analysis. The experiment is implemented in an $8000m \times 8000m$ area, with

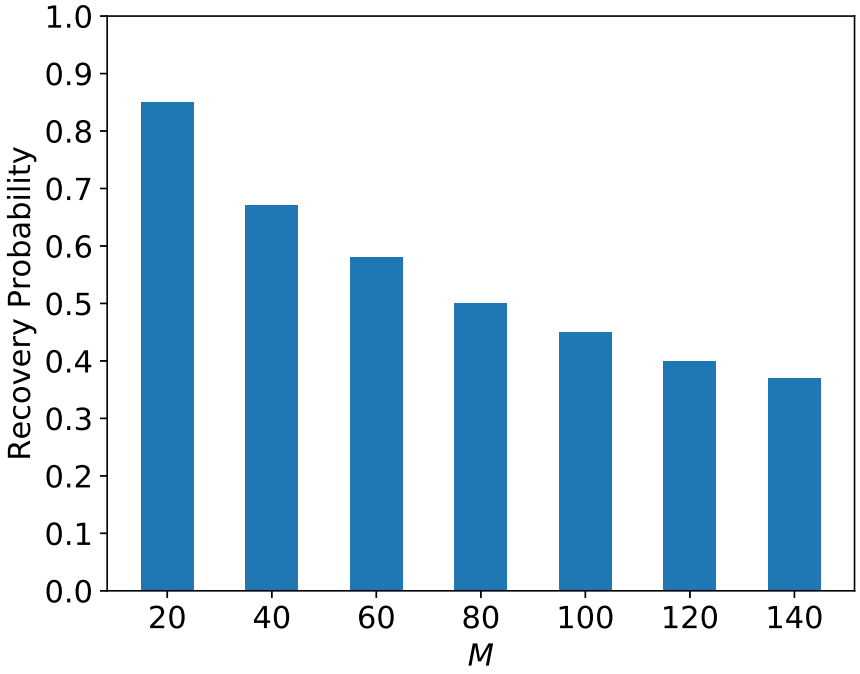


Figure 6.3: Recovery Probability as a function of the number of pre-deployed nodes M.

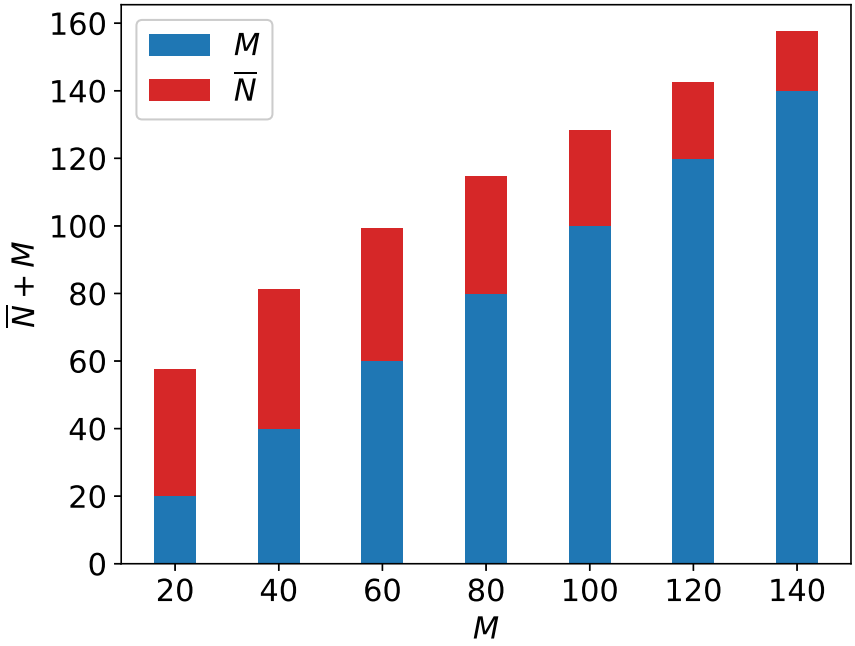


Figure 6.4: Bar plots of the total number of nodes $M + \bar{N}$ (where $\bar{N}$ indicates the *average* number of intermediate nodes) for different choices of pre-deployed nodes M in Fig. 6.3.

$r = 5m$ and $R = 455m$. Similar to previous large scale mobility experiments, we let the number of pre-deployed nodes M vary in the range of $[20, 100]$. For each value of M, we test 100 different network scenarios. In each test, M pre-deployed nodes are randomly placed at the beginning and then network connectivity is established using GDO algorithm. After that, a randomly selected node from the set V_0 undergoes random mobility in the range of $60r$ for the purpose of making the network disconnected. Then, Algorithm 4 is applied to restore connectivity and its runtime is measured.

We then plot the experimental runtime referred to as T, as a function of the runtime order given by (5.18). Fig. 6.5 shows a series of experimental results in which the horizontal axis is $M + 2N + \sqrt{M - r + 2N}$ and vertical axis is the algorithm runtime in seconds. Each subfigure in Fig. 6.5 shows the measure collected in 100 different networks with M pre-deployed nodes. It can be observed that the runtime has a linearly increasing trend with respect to the value of $M + 2N + \sqrt{M - r + 2N}$. The blue line shows the result of data fitting in each subfigure. The slope of each line is reported in Table. 6.1, where $\beta = 1000T/(M + 2N + \sqrt{M - r + 2N})$. Among all subfigures of Fig. 6.5, the steepest slope $\bar{\beta}$ is 5.2 implying thar $T \leq \bar{\beta}(M + 2N + \sqrt{M - r + 2N})/1000$. We can conclude that the runtime of Algorithm 4 is in the order of $\mathcal{O}(M + 2N + \sqrt{M - r + 2N})$.

In Table 6.1, we report typical runtimes associated with restoring connectivity using both Algorithm 4 and EGDO algorithm. The runtime of EGDO algorithm is denoted by T_E in the table. It can be observed that the runtime of EGDO algorithm is typically 2 to 4 times higher than that of Algorithm 4.

Table 6.1: Line slope measures in seconds fitting Fig. 6.5 data.

M	20	30	40	50	60	70	80	90	100	110
β	4.1	3.6	3.8	4.1	4.2	5.2	4.3	4.3	4.5	4.6
T_E	0.92	1.12	1.45	1.73	1.89	2.13	2.09	2.35	2.51	2.55
T	0.54	0.61	0.68	0.74	0.78	0.77	0.83	0.61	0.67	0.61

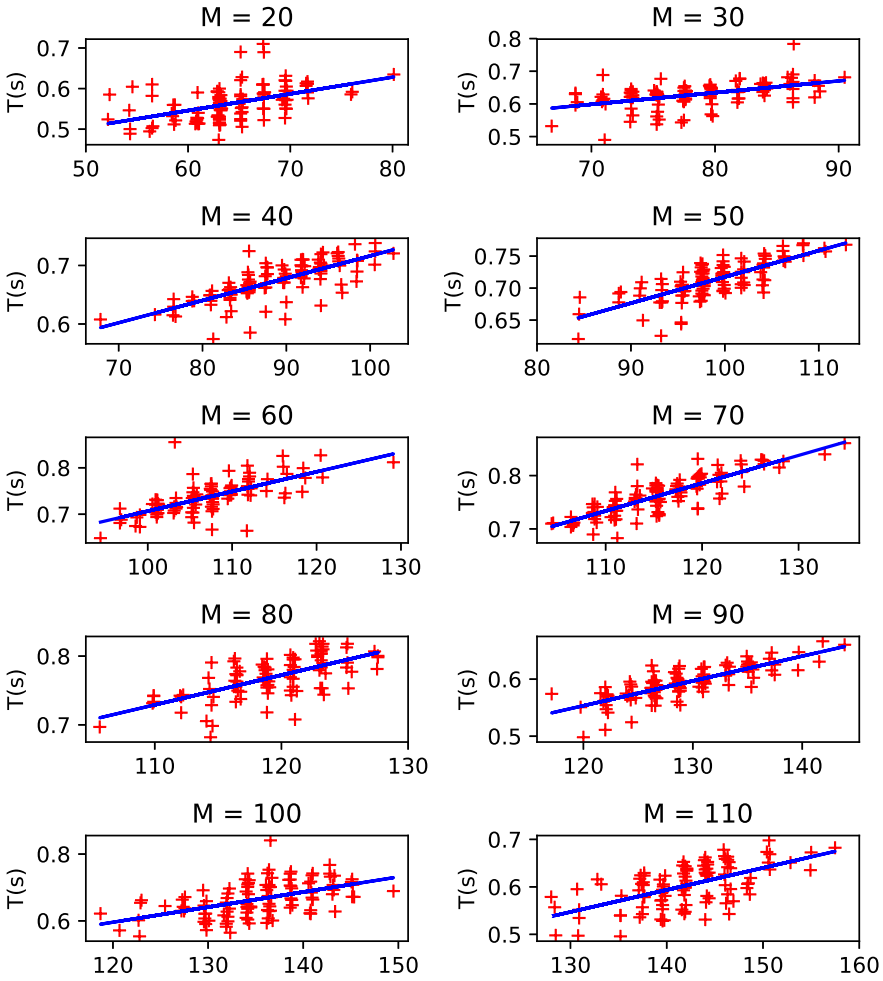


Figure 6.5: Capturings of the runtime of Algorithm 4 versus $M + 2N + \sqrt{M - r + 2N}$.

Chapter 7

Conclusion

7.1 Summary of Main Contribution

This dissertation studied two problems, network connectivity establishment and maintenance, and therefore can be divided into two parts. The first part is composed of Chapter 2, 3 and 4, and the second part consists of Chapter 5 and 6.

In the first part, we investigated robust connectivity establishment in two-tiered heterogeneous network graphs through systematic placement of advantaged nodes. The main contribution of this part is a class of near-optimal yet low complexity algorithms that solves this problem, and the corresponding numerical validations. Our algorithms were developed utilizing a so-called hexagonal coordinate system (HCS) in which we developed an extended algebra. We formulated and solved (within bounds) an NP-hard problem addressing graph connectivity. Further, we developed a class of geometric distance optimization (GDO) algorithms approximating the original problem. Experimental results showed the effectiveness of our proposed GDO algorithms measured in terms of advantaged node cost and robustness of connectivity in sparse networks in comparison with variants of exhaustive search and

Steiner minimum tree (SMT) algorithms. Our experimental results also offered a couple of additional important insights. First, it was commonly observed that our proposed GDO algorithms lost their advantages in comparison with SMT algorithms past a density threshold value due to the higher density of nodes. Second, below the specific sparsity threshold, our proposed algorithms used smaller numbers of AN nodes if we applied HCS representation to SMT algorithms in order to improve robustness.

In the second part, we studied connectivity maintenance scenarios in which an initially connected network consisting of pre-deployed and intermediate nodes was exposed to mobility. Two types of node mobility scenarios were considered. The first scenario analyzed the bounds of mobility when pre-deployed nodes moved at small scales. The bounds of node mobility preserving connectivity were derived through analysis and verified by simulations. The second scenario considered the movement of pre-deployed nodes beyond the bounds of the first scenario thereby breaking connected links and partitioning the connected network. This scenario then considered relocating the existing intermediate nodes in order to reestablish connectivity. Beside the analysis of mobility bound in the first scenario, the main contribution of this part is that for the first time, a general formulation of network maintenance problem, in a network we only have partial control, was proposed in the form of an optimization problem. This problem was later proven to be NP-complete. Then, we considered a practical location-aware special case scenario. We solved the problem of the practical scenario in polynomial time and analyzed the complexity of our solution. We also presented comprehensive performance evaluation results of our proposed algorithm.

7.2 Future Work

7.2.1 Connectivity Establishment and Maintenance in More Heterogeneous Network

In Chapter 2, we introduced the connectivity model where we assumed the intermediate nodes and the gateway nodes are the same. Nowadays, the diversity of devices in wireless networks grows fiercely. Each type of device has different communication capabilities. When dealing with such complex network environment, the assumption we imposed before is not quite practical. The network connectivity establishment and maintenance problem with the following conditions is left unexplored.

1. Intermediate nodes has different types. Each type has different communication range;
2. The number of intermediate nodes of different types are limited;

These two conditions imply the nature of the problem is still a knapsack problem, which is NP-hard. In order to establish robust connectivity, the future work should use the formula $R = (12n+7)r$ with different values of n to approximate the communication range of different types of intermediate nodes. After connectivity is established, the maintenance problem with pre-deployed nodes become mobile is also left unexplored. In this case, which node should be relocated and how to rearrange the network topology is a more sophisticated yet interesting problem.

7.2.2 Formation Control in Multi-agent System

In Chapter 5 and 6, we investigated the scenario when we were able to maneuverer the intermediate nodes to restore connectivity. A similar scenario not fully researched in multi-

agent systems is formation control. When there is such a network consisting of isolated mobile agents and intermediate facilities accommodating connectivity, and the mobile agents want to move to their new locations while keeping global connectivity along the way, the intermediate nodes (or a central hub) should provide a proper motion sequence. In another word, the intermediate nodes need to design a connected graph process so as to make the mobile agents reach their destinations, assuming the final topology can be connected with the same number of intermediate nodes. The deliverable of this research project should be a protocol deciding which node should move and providing its new location during each time period. To develop such a motion protocol is an interesting but unexplored problem.

7.2.3 Communication Energy Optimization

In Chapter 2, we assumed the communication radius of all the intermediate nodes are the same and fixed. In practice, this implies the wireless antennas are working at a fixed transmitting and receiving power. However, after a connected network is established, the wireless nodes may not need to work at the fixed power to guarantee connectivity. Based on the distance between different pairs of connected nodes, the antennas may be able to reduce the communication power but still maintain a connected network. Furthermore, with the given number of intermediate nodes, by modifying their locations locally, an optimal topology consuming the least overall communication power could be found. The problem of optimizing overall communication power by modifying node positions and probably even network topology is interesting but left unexplored.

Bibliography

[1] Jeroen Hoebeke, Ingrid Moerman, Bart Dhoedt, and Piet Demeester. An overview of mobile ad hoc networks: Applications and challenges. *Journal-Communications Network*, 3(3):60–66, 2004.

[2] Yuanjiang Huang, José-Fernán Martínez, Juana Sendra, and Lourdes López. Resilient wireless sensor networks using topology control: A review. *Sensors*, 15(10):24735–24770, 2015.

[3] Yuri Levin and Adi Ben-Israel. A heuristic method for large-scale multi-facility location problems. *Computers & Operations Research*, 31(2):257–272, 2004.

[4] Quanhong Wang, Glen Takahara, Hossam Hassanein, and Kenan Xu. On relay node placement and locally optimal traffic allocation in heterogeneous wireless sensor networks. In *Local Computer Networks, 2005. 30th Anniversary. The IEEE Conference on*, pages 8–pp. IEEE, 2005.

[5] Mohamed Younis and Kemal Akkaya. Strategies and techniques for node placement in wireless sensor networks: A survey. *Ad Hoc Networks*, 6(4):621–655, 2008.

[6] Y. Thomas Hou, Yi Shi, Hanif D. Sherali, and Scott F. Midkiff. On energy provisioning and relay node placement for wireless sensor networks. *Wireless Communications, IEEE Transactions on*, 4(5):2579–2590, 2005.

[7] K. Xu, Q. Wang, H. Hassanein, and G. Takahara. Optimal design of wireless sensor networks: minimum cost with lifetime constraints. *Proc. IEEE WiMob*, 5, 2005.

[8] Senni Perumal and John S Baras. Aerial platform placement algorithm to satisfy connectivity and capacity constraints in wireless ad-hoc networks. In *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, pages 1–5. IEEE, 2008.

[9] Guo-Hui Lin and Guoliang Xue. Steiner tree problem with minimum number of steiner points and bounded edge-length. *Information Processing Letters*, 69(2):53–57, 1999.

[10] E. L. Lloyd and G. Xue. Relay node placement in wireless sensor networks. *IEEE Transactions on Computers*, 56(1):134–138, Jan 2007.

[11] Donghui Chen, Ding-Zhu Du, Xiao-Dong Hu, Guo-Hui Lin, Lusheng Wang, and Guoliang Xue. Approximations for steiner trees with minimum number of steiner points. *Journal of Global Optimization*, 18(1):17–33, 2000.

[12] Xiuzhen Cheng, Ding-Zhu Du, Lusheng Wang, and Baogang Xu. Relay sensor placement in wireless sensor networks. *Wireless Networks*, 14(3):347–355, 2008.

[13] Dingzhu Du, Lusheng Wang, and Baogang Xu. The euclidean bottleneck steiner tree and steiner tree with minimum number of steiner points. In *Computing and Combinatorics*, pages 509–518. Springer, 2001.

[14] Jian Tang, Bin Hao, and Arunabha Sen. Relay node placement in large scale wireless sensor networks. *Computer communications*, 29(4):490–501, 2006.

[15] Quanhong Wang, Kenan Xu, Hossam Hassanein, and Glen Takahara. Minimum cost guaranteed lifetime design for heterogeneous wireless sensor networks (WSNs). In *Performance, Computing, and Communications Conference, 2005. IPCCC 2005. 24th IEEE International*, pages 599–604. IEEE, 2005.

[16] E. N. Gilbert. Random Plane Networks. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):533–543, December 1961.

[17] Ning Li and Jennifer C. Hou. Topology control in heterogeneous wireless networks: Problems and solutions. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 1. IEEE, 2004.

[18] Jianping Pan, Y. Thomas Hou, Lin Cai, Yi Shi, and Sherman X. Shen. Topology control for wireless sensor networks. In *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking*, MobiCom '03, pages 286–299, New York, NY, USA, 2003. ACM.

[19] Bin Hao, Jian Tang, and Guoliang Xue. Fault-tolerant relay node placement in wireless sensor networks: formulation and approximation. In *High Performance Switching and Routing, 2004. HPSR. 2004 Workshop on*, pages 246–250. IEEE, 2004.

[20] X. Han, X. Cao, E. L. Lloyd, and C. C. Shen. Fault-tolerant relay node placement in heterogeneous wireless sensor networks. *IEEE Transactions on Mobile Computing*, 9(5):643–656, May 2010.

[21] Gaurav Gupta and Mohamed Younis. Fault-tolerant clustering of wireless sensor networks. In *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*, volume 3, pages 1579–1584. IEEE, 2003.

[22] Jonathan L. Bredin, Erik D. Demaine, Mohammad Taghi Hajiaghayi, and Daniela Rus. Deploying sensor networks with guaranteed fault tolerance. *IEEE/ACM Transactions on Networking (TON)*, 18(1):216–228, 2010.

[23] Guoqiang Mao. *Connectivity of Communication Networks*. Springer, 2017.

[24] Homayoun Yousefi'zadeh, Hamid Jafarkhani, and Javad Kazemitabar. A study of connectivity in MIMO fading ad-hoc networks. *Journal of Communications and Networks*, 11(1):47–56, 2009.

[25] Sanaz Barghi, Hamid Jafarkhani, and Homayoun Yousefi'zadeh. MIMO-assisted MPR-aware MAC design for asynchronous WLANs. *IEEE/ACM Transactions on Networking*, 19(6):1652–1665, 2011.

[26] Xiaolong Li and Homayoun Yousefi'zadeh. Robust EKF-based wireless congestion control. *IEEE Transactions on Communications*, 61(12):5090–5102, 2013.

[27] Ahmed S. Ibrahim, Karim G. Seddik, and K. J. Ray Liu. Connectivity-aware network maintenance and repair via relays deployment. *IEEE Transactions on Wireless Communications*, 8(1):356–366, 2009.

[28] J. P. Macker, W. Chao, R. Mittu, and M. Abramson. Multi-agent systems in mobile ad hoc networks. In *MILCOM 2005 - 2005 IEEE Military Communications Conference*, Oct 2005.

[29] J. Fink, A. Ribeiro, and V. Kumar. Robust control for mobility and wireless communication in cyberphysical systems with application to robot teams. *Proceedings of the IEEE*, 100(1):164–178, 2012.

[30] Jorge Cortes, Sonia Martinez, Timur Karatas, and Francesco Bullo. Coverage control for mobile sensing networks. *IEEE Transactions on robotics and Automation*, 20(2):243–255, 2004.

[31] Tahiry Razafindralambo and David Simplot-Ryl. Connectivity preservation and coverage schemes for wireless sensor networks. *IEEE Transactions on Automatic Control*, 56(10):2418–2428, 2011.

[32] Benjie Chen, Kyle Jamieson, Hari Balakrishnan, and Robert Morris. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *Wireless networks*, 8(5):481–494, 2002.

[33] Guoliang Xing, Chenyang Lu, Xiaohua Jia, and Robert Pless. Localized and configurable topology control in lossy wireless sensor networks. *Ad Hoc Networks*, 11(4):1345–1358, 2013.

[34] F. O. Aron, T. O. Olwal, A. Kurien, and M. O. Odhiambo. Energy efficient topology control algorithm for wireless mesh networks. pages 135–140, Aug 2008.

[35] M. C. De Gennaro and A. Jadbabaie. Decentralized control of connectivity for multi-agent systems. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 3628–3633, Dec 2006.

[36] M. Franceschelli, A. Gasparri, A. Giua, and C. Seatzu. Decentralized laplacian eigenvalues estimation for networked multi-agent systems. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 2717–2722, Dec 2009.

[37] Peng Yang, Randy A. Freeman, Geoffrey J. Gordon, Kevin M. Lynch, Siddhartha S. Srinivasa, and Rahul Sukthankar. Decentralized estimation and control of graph connectivity for mobile sensor networks. *Automatica*, 46(2):390–396, 2010.

[38] Lorenzo Sabattini, Nikhil Chopra, and Cristian Secchi. Decentralized connectivity maintenance for cooperative control of mobile robotic systems. *The International Journal of Robotics Research*, 32(12):1411–1423, 2013.

[39] Kai Ding, Homayoun Yousefi'zadeh, and Faryar Jabbari. A robust advantaged node placement strategy for sparse network graphs. *IEEE Transactions on Network Science and Engineering*, 2017.

[40] Kai Ding and Homayoun Yousefi'zadeh. A systematic node placement strategy for multi-tier heterogeneous network graphs. In *2016 IEEE Wireless Communications and Networking Conference*, pages 1–6, April 2016.

[41] Majid Raissi-Dehkordi, Karthikeyan Chandrashekar, and John S. Baras. UAV placement for enhanced connectivity in wireless ad-hoc networks. Technical report, 2004.

[42] Thomas Clouqueur, Veradej Phipatanasuphorn, Parameswaran Ramanathan, and Kewal K Saluja. Sensor deployment strategy for target detection. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 42–48. ACM, 2002.

[43] Dario Pompili, Tommaso Melodia, and Ian F. Akyildiz. Deployment analysis in underwater acoustic wireless sensor networks. In *Proceedings of the 1st ACM international workshop on underwater networks*, pages 48–55. ACM, 2006.

[44] E. S. Biagioni and G. Sasaki. Wireless sensor placement for reliable and efficient data collection. In *36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of the*, pages 10 pp.–, Jan 2003.

[45] S. Toumpis and G. A. Gupta. Optimal placement of nodes in large sensor networks under a general physical layer model. In *2005 Second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, IEEE SECON 2005.*, pages 275–283, Sept 2005.

[46] K. Xu, H. Hassanein, G. Takahara, and Q. Wang. Relay node deployment strategies in heterogeneous wireless sensor networks. *IEEE Transactions on Mobile Computing*, 9(2):145–159, Feb 2010.

[47] Yunxia Chen, Chen-Nee Chuah, and Qing Zhao. Sensor placement for maximizing lifetime per unit cost in wireless sensor networks. In *MILCOM 2005 - 2005 IEEE Military Communications Conference*, pages 1097–1102 Vol. 2, Oct 2005.

[48] Hanan Shpungin and Michael Segal. On minimizing the total power of k-strongly connected wireless networks. *Wireless Networks*, 16(4):1075–1089, 2010.

[49] Jian Tang, Bin Hao, and Arunabha Sen. Relay node placement in large scale wireless sensor networks. *Computer communications*, 29(4):490–501, 2006.

[50] Michael M. Zavlanos and George J. Pappas. Potential fields for maintaining connectivity of mobile networks. *IEEE Transactions on Robotics*, 2007.

[51] D.V. Dimarogonas and K.H. Johansson. Bounded control of network connectivity in multi-agent systems. *IET Control Theory & Applications*, 4(8):1330–1338, 2010.

[52] Z. Han, A. L. Swindlehurst, and K. J. R. Liu. Optimization of MANET connectivity via smart deployment/movement of unmanned air vehicles. *IEEE Transactions on Vehicular Technology*, 58(7):3533–3546, Sept 2009.

[53] Michael M. Zavlanos, Magnus B. Egerstedt, and George J. Pappas. Graph-theoretic connectivity control of mobile robot networks. *Proceedings of the IEEE*, 99(9):1525–1540, 2011.

[54] S. Alireza Motevallian, Changbin Yu, and Brian D.O. Anderson. Robustness to the loss of multiple nodes in the localizability of sensor networks. *IFAC Proceedings Volumes*, 44(1):7836–7841, 2011.

[55] Shiguang Wang, Xufei Mao, Shao Jie Tang, Xiang Yang Li, Jizhong Zhao, and Guojun Dai. On movement-assisted connectivity restoration in wireless sensor and actor networks. *IEEE Transactions on Parallel and Distributed Systems*, 22(4):687–694, 2011.

[56] Mustafa Y. Sir, Izzet F. Senturk, Esra Sisikoglu, and Kemal Akkaya. An optimization-based approach for connecting partitioned mobile sensor/Actuator Networks. *2011 IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS 2011*, pages 525–530, 2011.

[57] Demetri P. Spanos and Richard M. Murray. Robust connectivity of networked vehicles. In *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, volume 3, pages 2893–2898. IEEE, 2004.

[58] Michael M. Zavlanos and George J. Pappas. Controlling connectivity of dynamic graphs. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, pages 6388–6393. IEEE, 2005.

[59] Nathan Michael, Michael M. Zavlanos, Vijay Kumar, and George J. Pappas. Maintaining connectivity in mobile robot networks. In *Experimental Robotics*, pages 117–126. Springer, 2009.

[60] Lorenzo Sabattini, Cristian Secchi, Nikhil Chopra, and Andrea Gasparri. Distributed control of multirobot systems with global connectivity maintenance. *IEEE Transactions on Robotics*, 29(5):1326–1332, 2013.

[61] Hasan A. Poonawala and Mark W. Spong. On maintaining visibility in multi-robot-networks with limited field-of-view sensors. In *American Control Conference (ACC), 2017*, pages 4983–4988. IEEE, 2017.

[62] Meng Ji and Magnus Egerstedt. Distributed coordination control of multiagent systems while preserving connectedness. *IEEE Transactions on Robotics*, 23(4):693–703, 2007.

[63] Yongcan Cao and Wei Ren. Distributed coordinated tracking via a variable structure approach-part i: Consensus tracking. In *American Control Conference (ACC), 2010*, pages 4744–4749. IEEE, 2010.

[64] Michael M. Zavlanos and George J. Pappas. Distributed connectivity control of mobile networks. *IEEE Transactions on Robotics*, 24(6):1416–1428, 2008.

[65] Derya Aksaray and Dimitri N. Mavris. Maintaining connectivity for networked mobile systems in the presence of agent loss. In *AIAA Guidance, Navigation, and Control (GNC) Conference*, page 4886, 2013.

[66] Feng Xiao, Long Wang, and Tongwen Chen. Connectivity preservation for multi-agent rendezvous with link failure. *Automatica*, 48(1):25–35, 2012.

[67] Nilanjan Chakraborty and Katia Sycara. Reconfiguration algorithms for mobile robotic networks. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 5484–5489. IEEE, 2010.

[68] A. S. Ibrahim, K. G. Seddik, and K. J. R. Liu. Connectivity-aware network maintenance and repair via relays deployment. *IEEE Transactions on Wireless Communications*, 8(1):356–366, Jan 2009.

[69] Mohammad Rafiee and Alexandre M Bayen. Optimal network topology design in multi-agent systems for efficient average consensus. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 3877–3883. IEEE, 2010.

[70] Olivier Dousse, François Baccelli, and Patrick Thiran. Impact of interferences on connectivity in ad hoc networks. *IEEE/ACM Transactions on Networking (TON)*, 13(2):425–436, 2005.

[71] Yicheng Lin, Wei Yu, and Yves Lostanlen. Optimization of wireless access point placement in realistic urban heterogeneous networks. In *Global Communications Conference (GLOBECOM), 2012 IEEE*, pages 4963–4968. IEEE, 2012.

[72] T. Zhang and K. Ding. A new proof of honeycomb conjecture by fractal geometry methods. *Frontiers of Mechanical Engineering*, 8(4):367–370, 2013.

[73] T. Zhang and K. Ding. Hierarchical fractal structure of perfect single-layer grapheme. *Frontiers of Mechanical Engineering*, 8(4):371–382, 2013.

[74] Ivan Stojmenovic. Honeycomb networks: Topological properties and communication algorithms. *IEEE Transactions on parallel and distributed systems*, 8(10):1036–1042, 1997.

[75] Ding-Zhu Du and Panos M. Pardalos. *Handbook of combinatorial optimization: supplement*, volume 1. Springer Science & Business Media, 2013.

[76] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. Network flows. 2014.

[77] Ning Li and Jennifer C. Hou. Improving connectivity of wireless ad hoc networks. In *Mobile and Ubiquitous Systems: Networking and Services, 2005. MobiQuitous 2005. The Second Annual International Conference on*, pages 314–324. IEEE, 2005.

[78] Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak mathematical journal*, 23(2):298–305, 1973.

[79] Chris Godsil and Gordon F Royle. *Algebraic graph theory*, volume 207. Springer Science & Business Media, 2013.

[80] J. Lofberg. YALMIP: a toolbox for modeling and optimization in MATLAB. In *2004 IEEE International Conference on Robotics and Automation (IEEE Cat. No.04CH37508)*, pages 284–289, Sept 2004.

[81] Erling D. Andersen and Knud D. Andersen. The MOSEK interior point optimizer for linear programming: an implementation of the homogeneous algorithm. In *High performance optimization*, pages 197–232. Springer, 2000.

[82] Jan Fiala, Michal Kočvara, and Michael Stingl. Penlab: A matlab solver for nonlinear semidefinite optimization. *arXiv preprint arXiv:1311.5240*, 2013.

[83] Harsha Nagarajan, Sivakumar Rathinam, Swaroop Darbha, and Kumbakonam Rajagopal. Algorithms for synthesizing mechanical systems with maximal natural frequencies. *Nonlinear Analysis: Real World Applications*, 13(5):2154–2162, 2012.

[84] Harsha Nagarajan, Peng Wei, Sivakumar Rathinam, and Dengfeng Sun. Heuristics for Synthesizing Robust Networks with a Diameter Constraint. 2014, 2014.

[85] Harsha Nagarajan, Sivakumar Rathinam, and Swaroop Darbha. Synthesizing Robust Communication Networks for Unmanned Aerial Vehicles With Resource Constraints. *Journal of Dynamic Systems, Measurement, and Control*, 137(6):061001, 2015.

[86] Alexander Domahidi, Eric Chu, and Stephen Boyd. ECOS: An SOCP solver for embedded systems. In *Control Conference (ECC), 2013 European*, pages 3071–3076. IEEE, 2013.

[87] Farid Alizadeh and D. Goldfarb. Second-order cone programming. *Mathematical Programming*, 95(1):3–51, 2003.

[88] Miguel Sousa Lobo, Lieven Vandenberghe, Stephen Boyd, and Hervé Lebret. Applications of second-order cone programming. *Linear Algebra and its Applications*, 284(1-3):193–228, 1998.

[89] Jonathan L Gross and Jay Yellen. *Graph theory and its applications*. CRC press, 2005.

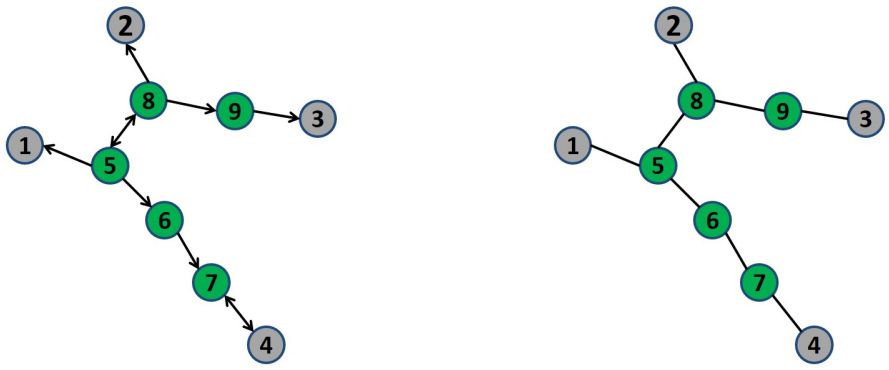
[90] Douglas Brent West et al. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, 2001.

[91] Damon Mosk-Aoyama. Maximum algebraic connectivity augmentation is NP-hard. *Operations Research Letters*, 36(6):677–679, 2008.

Appendices

A A Brief Introduction to Algebraic Graph Theory

Graphs are usually used to model the relations between objects. The objects could be people in social networks, sensors in wireless sensor networks, mobile devices in mobile wireless networks, protein structures, etc. The relations between them could be connected or not connected, directional interacted and weighted interaction. *Node*(or *Vertex* in some references) *Edge* are the two building blocks of a graph, which often denoted as $G(V, E)$ where V is the node set and E is the edge set. If the direction of edges are specified, then the graph G is called *directed* graph or *digraph*, as shown by A.1a. Oppositely, if the direction of edges are not specified, then the graph is called undirected graph, as shown by A.1b. In



(a) Directed graph (b) Undirected graph

Figure A.1: Directed graph v.s. undirected graph

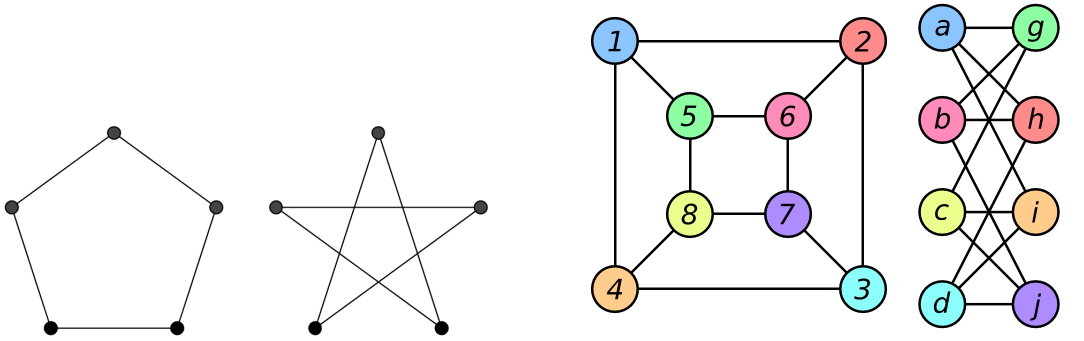
the research are of network flows, a *weight* is usually associated to each edge. The weights are usually real numbers representing materials flowing from one node to another. If not specified, the weights are regarded as all 1. This an arbitrary choice. We should keep in mind that the weights always have impacts to some of the results derived under the assumption of a weightless graph. Another concept we need to get familiar with is *status*. The status of a node can be the position of the node, velocity, voltage level, energy level, amount of currency stored, etc. Later when we need to describe the dynamics of a network graph with state space, the concept of node status will be in use.

If the set V and E evolves with respect to time, for instance, the position of one or more vertices changes, or some edges would disappear and appear under some condition, then the graph G is referred to as dynamic graph. The topology of a graph can be described by a symmetric matrix, namely, adjacency matrix. Given an undirected graph $G(V, E)$ with n nodes v_i, where $i = 1, 2, \ldots, n$, the adjacency matrix $A \in \mathbb{R}^{n\times n}$, with a_{ij} being its elements, is defined as

$$a_{ij} = \begin{cases} 0, \text{ if } v_i \text{ and } v_j \text{ are not connected} \\ 1, \text{ there exist an edge between } v_i \text{ and } v_j \end{cases}$$

$D = \mathbf{diag}\{d_1, d_2, \ldots, d_n\}$ is the degree matrix of graph G, where $d_i = \sum_{j=1}^{n} a_{ij}$ represents the degree of v_i. One may wonder why we need to represent a graph with matrix, which is likely to drag us into the swamp of linear algebra. Before we plunge into linear algebra, let's talk about the motivation, from the point view of electrical engineering.

One may recall the excitement when Fourier Transform was first introduced in studying signal processing. After suffering the calculation of convolution for a few weeks, we were given the new tool to easily compute convolution between two signals in time domain by doing multiplication in frequency domain. Similarly, integral of a signal is done by division in frequency domain and differentiation by multiplication. What's more, we can analyze



(a) Two identical graphs with 5 nodes. (b) Two identical graphs with 8 nodes.

Figure A.2: Isomorphism

signals in frequency domain and efficiently filter out noisy or unrelated signals, which is hardly doable in time domain. By and large, all these nice features came from observing the same object from a quite different view. Another example in is duality, such as prime and dual problems in linear programming, and controllability and observability in linear systems.

Coming back to graphs, a graph can be really complex if the number of nodes and edges becomes large. In that case, some features of the graph are difficult to analyze by barely looking at the topology. What was worse, two identical graphs may show as different topology. For example, in Fig. A.2, we show two pairs of identical graphs with different topologies. Fig. A.2a shows a graph with 5 nodes. The layout on the left hand side is a pentagon, each node is connected to 2 edges and all edges has no cross in 2D space. The graph on the right hand side has the same node and edge set. But the edges are going across each other. The left one is called 'Planar Graph' or graph embedded in 2D space [89]. Fig. A.2b show a graph that can be realized with a bipartite. Here 'bipartite' represents a graph whose node set can be divided into two subsets. The nodes within each subset are not neighbors but each node is connected to at least one node in the other set [90]. This concept is widely used in neuron network. By looking at the two graphs in Fig. A.2a for a few minutes, we are able to recognize their identity. However, the fact that Fig.A.2b shows two identical graphs is not obvious to us. Limitations of analyzing graphs in topology domain motivates the work of

developing a different tool. Matrix and linear algebra turned out to be an effective one. This tool, named "algebraic graph theory", calls for analyzing properties of a graph by studying the spectrum of its corresponding adjacency matrix and Laplacian matrix.

A.1 Adjacency Matrix

Adjacency matrix is the most fundamental matrix in algebraic graph theory. Its definition is as following.

$$a_{ij} = \begin{cases} 0, \text{ if } v_i \text{ and } v_j \text{ are not connected} \\ w_{ij}, \text{ there exist an edge between } v_i \text{ and } v_j \end{cases} \tag{A.1}$$

w_{ij} is the weight on edge (v_i, v_j). Usually, when representing connectivity condition only, the weight on each edge is binary. When it comes to digraphs, w_{ij} represents the weight on the edge going into node v_i. The adjacency matrix for Fig. A.1a is given in (A.2)

$$A_a = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{A.2}$$

The adjacency matrix for Fig. A.1b is given in (A.3). One should notice the adjacency matrix for digraph is not necessarily symmetric.

$$A_b = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{A.3}$$

The following remark is one of the most interesting and useful property of adjacency matrix.

Remark. *The number of walks of length l in a graph G, from v_i to v_j is the element at the position (i, j) of the matrix $\mathbf{A}^l$.*

Let's take Fig. A.2b as an example and look at this property. The adjacency matrix of the

left graph in Fig. A.2b is

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

By inspection, we found there are 6 paths of length 3 from node v_5 to v_3, and the element $A(5,3)$ is 6.

$$A^3 = \begin{bmatrix} 0 & 7 & 0 & 7 & 7 & 0 & 6 & 0 \\ 7 & 0 & 7 & 0 & 0 & 7 & 0 & 6 \\ 0 & 7 & 0 & 7 & 6 & 0 & 7 & 0 \\ 7 & 0 & 7 & 0 & 0 & 6 & 0 & 7 \\ 7 & 0 & 6 & 0 & 0 & 7 & 0 & 7 \\ 0 & 7 & 0 & 6 & 7 & 0 & 7 & 0 \\ 6 & 0 & 7 & 0 & 0 & 7 & 0 & 7 \\ 0 & 6 & 0 & 7 & 7 & 0 & 7 & 0 \end{bmatrix}$$

A.2 Laplacian Matrix

In algebraic graph theory, a more important matrix than adjacency matrix is Laplacian matrix, defined as $L = D - A$, where D is the degree matrix. In this dissertation, we use S_+^n

to represent the set of symmetric positive semidefinite(PSD) matrices of dimension n. Being a real symmetric matrix, L carries the following properties that are related to our class.

1. Due to its real symmetry, all eigenvalues of L, ordered such that $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$, are real numbers.

2. According to Gershgorin's theorem, all eigenvalues of L are at least zero. Consequently, we have $L \in S_+^n$, with a complete orthonormal eigenspace, referred to as $\{u_1, u_2, \ldots, u_n\}$. This is also easy to see through the quadratic form

$$x^T L x = \sum_{(v_i, v_j) \in E} (x_i - x_j)^2$$

3. $L\mathbf{1} \equiv \mathbf{0}$, therefore $\lambda_1 = 0$ and $\mathbf{1}$ is an eigenvalue-eigenvector pair of any Laplacian matrix. Here $\mathbf{1}$ is the vector of all ones.

4. $\forall u_i$(note that $u_1 = \mathbf{1}$), we have $u_i^T u_j = 0 (i \neq j)$ and $u_i^T u_i = 1$. In particular, for $i \geq 2$, $u_i \in \mathbf{1}^\perp$.

5. The second smallest eigenvalue, λ_2, is also known as the algebraic connectivity of a graph. $\lambda_2 > 0$ if and only if the graph is connected.

6. Since λ_2 is the smallest eigenvalue except 0, λ_2 can be calculated through Raleigh quotient

$$\lambda_2 = u_2^T L u_2 = \min_{u \in \mathbf{1}^\perp, u \neq \mathbf{0}} \frac{u^T L u}{u^T u}$$

The first property is trivial. We give Gershgorin's theorem as a reference below for the second property.

Theorem A.1 (Gershgorin's Theorem)**.** *Every eigenvalue of a matrix* $\mathbf{M} \in \mathbb{R}^{n\times n}$*, with* m_{ij} *being its elements, satisfies*

$$|\lambda_i - m_{ii}| \leq \sum_{j\neq i} |m_{ij}| \quad i \in \{1, 2, \ldots, n\}$$

B Missing proofs in Chapter 5

B.1 Proof of Lemma 1

Lemma 1. *The eigenvalue* $\lambda_2 > 0$ *if and only if* $\exists W$ *such that* $W^T LW$ *is positive definite where* $W = [w_1, w_2, \ldots, w_{n-1}] \in \mathbb{R}^{n\times(n-1)}$ *with* $w_i^T \mathbf{1} = 0\ \forall i \in \{1, 2, \ldots, n-1\}$ *and* $w_i^T w_j = 0$ $\forall i \neq j$*.*

Proof. For any graph, the Laplacian matrix L is a positive semidefinite matrix. It can be proven that the eigenvector corresponding to $\lambda_1 = 0$ is $\mathbf{1}$ and L has an orthonormal eigenspace. Therefore, we claim $\lambda_2(L) > 0$ if and only if $w^T Lw > 0$ for any vector $w \in \mathbf{1}^\perp$.

To prove this assuming u_i is the eigenvector corresponding to $\lambda_i(L)$ with $i > 1$, we have $u_i \perp \mathbf{1}$. On one hand, $\lambda_i > 0$ implies $u_i^T Lu_i > 0$. On the other hand, the dimension of the vector space $\mathbf{1}^\perp = \{u | u \perp \mathbf{1}\}$ is at most $(n-1)$. Therefore, if we can find a full-rank matrix $W = [w_1, w_2, \ldots, w_{n-1}] \in \mathbb{R}^{n\times(n-1)}$ with $w_i^T \mathbf{1} = 0$, then any eigenvector $u \in \mathbf{1}^\perp$ is in $span(W)$ and can be expressed as

$$u = \sum_{i=1}^{n-1} \alpha_i w_i \tag{B.4}$$

where $\alpha_i \in \mathbb{R}$ and not all α_i's are zero. Thus, $u^T Lu > 0$ if and only if $W^T LW \succ 0$. We conclude that $\lambda_2(L) > 0$ if and only if $W^T LW \succ 0$. □

B.2 Proof of Theorem 5.1

Theorem 5.1. *In the case of the smallest safety margin, an edge can tolerate a maximum mobility distance of $\delta = r/2$ for any given value of n satisfying $R = (12n + 7)r$.*

Proof. It can be seen in Fig. 5.1b that the smalles safety margin $\delta = 2R - |JK|$ where $|JK|$ denotes center-to-center Euclidean distance between node J and K. We calculate $|JK|$ in $\triangle JKT$ with $|KT| = r$. Since $|TY|$ and $|KX|$ are parallel and equal, $|TY| = R$. Thus, $|JT| = |JY| + |YT| = 2R$. Applying the Law of Cosines to $\triangle JKT$ implies

$$\begin{aligned} |JK|^2 &= |JT|^2 + |TK|^2 - 2|JT||TK| \cos\frac{\pi}{3} \\ &= 4R^2 + r^2 - 2Rr \end{aligned} \tag{B.5}$$

and

$$\delta = 2R - |JK| = \frac{4R^2 - |JK|^2}{2R + |JK|} = r \cdot \frac{1 - \frac{r}{2R}}{1 + \frac{|JK|}{2R}} \tag{B.6}$$

Define

$$z = \frac{r}{2R} = \frac{1}{2(12n + 7)}, \quad 0 < z \leq \frac{1}{14}, \ (n = 0, 1, 2, \cdots) \tag{B.7}$$

Let $\delta = r \cdot f(z)$ with

$$f(z) = \frac{1 - \frac{r}{2R}}{1 + \frac{|JK|}{2R}} = \frac{1 - \frac{r}{2R}}{1 + \frac{\sqrt{4R^2 + r^2 - 2Rr}}{2R}} = \frac{1 - z}{1 + \sqrt{1 + z^2 - z}} \tag{B.8}$$

To find the maximum of $f(z)$ on its domain, we take derivative

$$f'(z) = \frac{-(1 + \sqrt{1 + z^2 - z}) - \frac{(1-z)}{2\sqrt{1+z^2-z}}}{(1 + \sqrt{1 + z^2 - z})^2} < 0 \tag{B.9}$$

Thus, $f(z)$ is monotonically decreasing over its domain implying $\max f(z) = 1/2$ at $z = 0$. We conclude that a maximum mobility distance of $r/2$ can be tolerated for any given value of n satisfying $R = (12n + 7)r$. □

B.3 Proof of theorem 5.2

We prove that Problem 5.1 is NP-hard by showing that there exists a polynomial reduction from a known NP-complete problem to Problem 5.1.

In [91], it is proven that the maximum algebraic connectivity augmentation problem in its deision version and as stated below, is NP-complete.

Problem B.1. *Given an undirected graph $G(V, E)$, let $G^c(V, E^c)$ denote the complementary graph of G where $E^c = \{(v_x, v_y)|v_x, v_y \in V, (v_x, v_y) \notin E\}$. Then, the maximum algebraic connectivity augmentation problem is defined as follows.*
Instance: Given a non-negative integer k and a non-negative threshold τ.
Problem: Is there a subset $S \subseteq E^c$ of cardinality $|S| \leq k$ such that the graph $H(V, E \cup S)$ satisfies $\lambda_2(H) \geq \tau$?

To better assist the proof, we also put Problem 5.1 in its decision version here.

Problem B.2. *Given an undirected graph $G(V, E)$ with two types of nodes belonging to sets V_0 and V_1, let the known positions of nodes in V_0 be denoted as $P \in \mathbb{R}^{M\times 2}$ and the nodes in V_0 form a disconnected graph. Denote the positions of nodes in V_1 as $Q \in \mathbb{R}^{N\times 2}$ and let $2R$ represent the communication radius of a node. Further, let L be the augmented Laplacian matrix with c_{ij}'s representing the existence of edges $\{(v_i, v_j)|v_i \in V_0, v_j \in V_1\}$ and f_{ij}'s representing the existence of edges $\{(v_i, v_j)|v_i, v_j \in V_1\}$.*
Instance: Given a disconnected network $G(V_0 \cup V_1, E)$ as described above.
Problem: Is there a set of c_{ij}'s and f_{ij}'s as well as Q values that satisfy the constraints in

Problem 5.1?

Theorem 5.2. *Problem 5.1 is NP-hard.*

Proof. To prove the NP-hardness of Problem 5.1, we need to show its decision version, Problem B.2, is a reduction from Problem B.1. One may notice that the most significant distinction between these two problems lies on the norm constraints (5.7) and (5.8) in Problem 5.1. In order to construct an equivalent instance to Problem B.1, we choose R as follows.

$$R = \begin{cases} 1/(1-c_{ij})^2, & \text{for } R \text{ in (5.7)} \\ 1/(1-f_{ij})^2, & \text{for } R \text{ in (5.8)} \end{cases} \tag{B.10}$$

In this case, constraints (5.7) and (5.8) are automatically satisfied. Then, the optimization Problem 5.1 becomes the following feasibility problem.

$$\begin{aligned} \min_{C,F} \quad & 0 \\ \text{S.T.} \quad & W^T L W \succ 0 \\ & 2 \le f_{ii} \le 5, \quad \forall i \in 1, 2, \ldots, N \\ & c_{ij} \in \{0, 1\} \\ & f_{ij} \in \{0, 1\} \end{aligned}$$

This above feasibility problem has a solution if and only if the following optimization problem

is feasible.

$$
\begin{aligned}
\min_{C,F} \quad & -\theta \\
\text{S.T.} \quad & W^T L W \succ \theta \cdot I_{M+N-1} \\
& 2 \le f_{ii} \le 5, \qquad \forall i \in 1, 2, \ldots, N \\
& c_{ij} \in \{0, 1\} \\
& f_{ij} \in \{0, 1\} \\
& \theta \ge 0
\end{aligned}
$$

To be consistent with Problem B.1, we can change the objective function into a maximization problem by considering its opposite value. Along with Lemma 1 and property (6) in Section 5.2.1, the above optimization problem equals to

$$
\begin{aligned}
\max_{C,F} \quad & \lambda_2(L) && \text{(B.11)} \\
\text{S.T.} \quad & 2 \le f_{ii} \le 5 \quad \forall i \in 1, 2, \ldots, N && \text{(B.12)} \\
& c_{ij} \in \{0, 1\} && \text{(B.13)} \\
& f_{ij} \in \{0, 1\} && \text{(B.14)} \\
& \lambda_2 \ge 0 && \text{(B.15)}
\end{aligned}
$$

This above optimization problem is in the standard form of a maximum algebraic connectivity problem, with an extra constraint (B.12).

We choose $k = M+N-1$ and τ be an arbitrarily small positive number. With the equivalent transformations illustrated, the maximum algebraic connectivity problem can be reduced to an instance of Problem B.2 with R chosen as in (B.10). The extra constraint (B.12) enforces that Problem B.2 is at least as hard as a standard maximum algebraic connectivity problem,

which concludes the proof. □

B.4 Proof of Theorem 5.3

Theorem 5.3. *Constraint (5.7) is equivalent to a following LMI, given the large positive number* Ω.

$$\begin{bmatrix} 4R^2 + \Omega(1 - c_{ij}) & (p_i - q_j) \\ (p_i - q_j)^T & I_{2\times 2} \end{bmatrix} \succeq 0 \tag{5.12}$$
$$\forall j \in \{1, 2, \ldots, N\} \quad \& \quad \forall i \in \{1, 2, \ldots, M\}$$

Proof. The transformation from constraint (5.7) to (5.12) contains two steps. First, we show constraint (5.7) is equivalent to the following inequality.

$$||p_i - q_j||^2 \leq 4R^2 + \Omega(1 - c_{ij}) \tag{B.16}$$

In (B.16), when $c_{ij} = 1$, the edge $(v_i, v_j) \in E$. This inequality becomes $||p_i - q_j||^2 \leq 4R^2$ which implies the distance between client node v_i and intermediate node v_j is less than $2R$. When $c_{ij} = 0$, there is no edge between v_i and v_j. Since Ω is a large positive number ($\Omega \approx \infty$), inequality (B.16) becomes $||p_i - q_j||^2 \leq \infty$, which is always true if the nodes are allowed to move within a fixed boundary field.

Next, we show (B.16) is equivalent to (5.12). Relying on Schur Complement,

$$\begin{bmatrix} A & B \\ B^T & C \end{bmatrix} \succeq 0 \Leftrightarrow \begin{cases} C \succ 0, \\ A - BC^{-1}B^T \succeq 0 \end{cases} \tag{B.17}$$

Let $A = 4R^2 + \Omega(1 - c_{ij})$, $C = I_{2\times 2} \succ 0$, and $B = (p_i - q_j)$. If (B.16) holds, we have

$$A - BC^{-1}B^T = 4R^2 + \Omega(1 - c_{ij}) - (p_i - q_j)(p_i - q_j)^T \geq 0$$

Since both conditions are satisfied, we conclude that inequality (B.16) holds if and only if the LMI condition of (5.12) is satisfied. □

B.5 Proof of Theorem 5.4

Theorem 5.4. *Given a network graph with M pre-deployed clusters and N intermediate nodes, suppose the primitive graph consisting of only M pre-deployed nodes has a Laplacian matrix L_0 with rank$(L_0) = r$. Then, the trace of matrix F in the augmented Laplacian matrix L, defined by Eq. (5.4), satisfies the following inequality.*

$$Tr(F) \leq M - r + 2N - 2 \tag{5.16}$$

Proof. From algebraic graph theory, we know that $(M - r)$ is equal to the number of connected components of V_0. Given a disconnected network with $(M-r)$ components consisting of M pre-deployed clusters and N intermediate nodes, the connected graph with the least number of edges is a spanning tree. Therefore, we need to construct a spanning tree with $(M - r + N - 1)$ edges in order to establish connectivity for a graph with $(M - r) + N$ components. Among these edges, at least $(M - r)$ edges need to be connected to pre-deployed nodes, which is to say that at least $(M - r)$ elements in matrix C are equal to 1. This happens when all components formed by the pre-deployed clusters are terminals and connected to only one intermediate node. In this case, the remaining $(N - 1)$ edges are connected to the nodes of set V_1. Hence, there are $(N - 1)$ f_{ij}'s equal to 1.

The trace of matrix F can then be calculated noticing that each $c_{ij} = 1$ increases the total

vertex degree of V_1 by 1 while each $f_{ij} = 1$ increases the total vertex degree of V_1 by 2. According to the Handshaking theorem, we have

$$Tr(F) = \sum_{j=1}^{N} d_j = (M - r) + 2(N - 1), \quad v_j \in V_1$$

However if some of the pre-deployed nodes in V_0 are not terminal, then more than one intermediate nodes are connected to them. In this case, some of the f_{ij}'s are altered by the same number of c_{ij}'s reducing $Tr(F)$. Thus, the above case with all pre-deployed components being terminal is an upper bound on $Tr(F)$, i.e.,

$$Tr(F) \leq M - r + 2N - 2$$

□