Resource Allocation in Communication and Social Networks

by

Shahrzad Shirazipourazad

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved February 2014 by the
Graduate Supervisory Committee:

Arunabha Sen, Chair
Guoliang Xue
Andrea Richa
Srikanth Saripalli

ARIZONA STATE UNIVERSITY

May 2014

ABSTRACT

As networks are playing an increasingly prominent role in different aspects of our lives, there is a growing awareness that improving their performance is of significant importance. In order to enhance performance of networks, it is essential that scarce networking resources be allocated smartly to match the continuously changing network environment. This dissertation focuses on two different kinds of networks - communication and social, and studies resource allocation problems in these networks. The study on communication networks is further divided into different networking technologies - wired and wireless, optical and mobile, airborne and terrestrial.

Since nodes in an airborne network (AN) are heterogeneous and mobile, the design of a reliable and robust AN is highly complex. The dissertation studies connectivity and fault-tolerance issues in ANs and proposes algorithms to compute the critical transmission range in fault free, faulty and delay tolerant scenarios.

Just as in the case of ANs, power optimization and fault tolerance are important issues in wireless sensor networks (WSN). In a WSN, a tree structure is often used to deliver sensor data to a sink node. In a tree, failure of a node may disconnect the tree. The dissertation investigates the problem of enhancing the fault tolerance capability of data gathering trees in WSN.

The advent of OFDM technology provides an opportunity for efficient resource utilization in optical networks and also introduces a set of novel problems, such as routing and spectrum allocation (RSA) problem. This dissertation proves that RSA problem is NP-complete even when the network topology is a chain, and proposes approximation algorithms.

In the domain of social networks, the focus of this dissertation is study of influence propagation in presence of active adversaries. In a social network multiple vendors may attempt to influence the nodes in a competitive fashion. This dissertation investigates the scenario where the first vendor has already chosen a set of nodes and the second vendor, with the knowledge of the choice of the first, attempts to identify a smallest set of nodes so that after the influence propagation, the second vendor's market share is larger than the first.

To my dear family.

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation and thanks to my advisor, Professor Arunabha Sen for giving me the chance to be a member of his research team and for providing me with invaluable guidelines, great ideas and feedback, constructive supervision and infinite support throughout my doctoral studies. I would also like to thank Professor Andrea Richa, Professor Guoliang Xue and Professor Srikanth Saripalli for their helpful comments on my work and serving on my committee.

I am immensely grateful to my seniors Dr. Pavel Ghosh and Dr. Sujogya Banerjee, whose precious advices and guidance made my initial years of doctoral studies smooth. I would also like to thank my lab-mates Zahra, Brian, Harsh, Arun, and Anisha with whom I had a great experience. I would also like to thank all my friends at Arizona State University especially Zahra Abbasi, Ali Abbasi, Yasaman Khodadegan, Moeed Haghnevis, Mahdi Hamzeh, Mitra Eftekhar, Zahra Derakhshandeh, Babak Esmaeili, Mojtaba Rahmati, Afsaneh Nasseri, Mikal Askarian, Pegah Shekh Olia, Ramtin Karmani, Soroosh Gholami, and many others who made this experience enjoyable.

I would like to thank my dear relative Nasrin Rouintan and her nice family in Arizona for their support and being with whom I always felt like being at home with a new great family.

Words cannot express how grateful I am to my parents for their inseparable supports and prayers. It is from them I learned the knowledge of life and grew up to be the person I am today. I would also like to thank my brother and his family, my sister, my parents in law, my sister in law and her family for all their encouragements and motivating me to strive towards my goal.

Finally, I would like to specially thank my husband Sina for being a continuous source of strength and support in every aspect of my life. It was because of him I ventured into this unexplored part of the world thousands of miles away from home and became successful in achieving our dreams. His philosophy, guidance, time and moral support was always by my side whenever I needed them most.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Chapter 1

INTRODUCTION

We live in a networked world. Our computers, power grids, transportation systems, water distribution systems, biological systems are networked and even we are networked through our social networks. Because of the widespread use of networks and their increasingly prominent role in various aspects of our lives, improving their performance will positively influence our lives. In order to enhance performance of networks, it is essential that scarce networking resources be allocated efficiently to match the continuously changing network environment. The dissertation focuses on two different kinds of networks - communication and social, and studies resource allocation problems in these networks. The studies on communication networks is further divided into different networking technologies - wired and wireless, optical and mobile, airborne and terrestrial.

1.1 Motivation and Challenges

Airborne networks (AN) have drawn attention of researchers in the past few years due to their importance in civil and military purposes and due to the several complex issues in these domains. Airborne networks are heterogeneous mobile ad hoc networks consisting of air vehicles as well as space and ground vehicles. Mobility pattern of nodes in a mobile network has significant impact on the coverage and connectivity properties of the network. It has been shown that infrastructure-less mobile ad hoc networks have limitations with respect to data transmission, distance, interference and scalability [1]. Accordingly, the authors in [2, 1] suggested the addition of a mobile wireless backbone of base stations (analogous to cellular telephony or the Internet backbones), in which topologies and mobility can be controlled for purposes of assured communications. End to end connectivity of the backbone nodes are crucial in providing the communication among the hosts. Network connectivity can be eas-

ily achieved if the transmission range of the airborne networking platforms (ANPs) is very large. However, large transmission range results in high power consumption and interference. Since in mobile wireless networks power is scarce, utilizing power efficiently is very critical. These requirements and limitations give rise to the following resource optimization problem in airborne networks: minimizing the ANPs transmission range such that the backbone network remains *connected at all times*, even though the topology of the network changes with the movement of the ANPs. Since ANPs are prone to failure because of attacks like EMP attack or jamming, one important issue is to improve the robustness of the backbone network against these attacks. Such attacks will impact specific geographic regions at specific times and if an ANP is within the fault region during the time of attack, it will fail. This research also considers the AN scenario where a region may fail and studies the following problem: minimizing the ANPs transmission range such that the network remains connected *irrespective of location of the fault region* and the *time of failure*. Even though connectivity of the backbone network at all times is an important and desired requirement, it may not be possible to equip the ANPs with radios with transmission range large enough to keep the network connected at all times. In such a scenario the network may operate in a disconnected mode for some part of time. On the other hand, based on the type of data that should be transmitted between ANPs, data transmissions may be tolerant to some amount of delay. Hence, ANPs may not need to have end-to-end paths all the time but they should be able to transmit data to each other within bounded time. Another challenge that is considered in this thesis is the problem of computation of critical transmission range in delay tolerant airborne networks.

Just as in the case of airborne networks, power optimization and fault tolerance are equally important in wireless sensor networks. In sensor networks often a tree

structure is used to deliver collected sensors data to a sink node. A tree is adequate for data gathering from all sensor nodes as long as no node in the tree fails. Since the connectivity of the tree is one, failure of any one node disconnects the tree and may disable the sink node from collecting data from some of the sensor nodes. Hence, reinforcing the data gathering tree against node failures in sensor networks is a critical issue that should be addressed. This dissertation studies the problem of enhancing the fault tolerance capability of a data gathering tree by adding a few additional links so that the failure of any one sensor or a pair of adjacent sensors would not disconnect the tree. Assuming that the addition of each link to the tree involves some cost, this research studies the problem of least-cost augmentation of the tree, so that even after failure of a single node or two adjacent nodes, all the surviving nodes will remain connected to the sink node.

The continuous growth of data traffic in Internet necessitates introduction of innovative and efficient technology solutions in optical networks of the future. The orthogonal frequency division multiplexing (OFDM) technology provides an opportunity for efficient resource utilization in optical networks. It allows allocation of multiple sub-carriers to meet traffic demands of varying size. Utilizing OFDM technology, a spectrum efficient and scalable optical transport network called SLICE was proposed recently. The SLICE architecture enables sub-wavelength, super-wavelength resource allocation and multiple rate data traffic that results in efficient use of spectrum. However, the benefit is accompanied by additional complexities in resource allocation. In SLICE architecture, in order to minimize utilized spectrum, one has to solve the *routing and spectrum allocation* (RSA) problem, a generalization of the *routing and wavelength allocation* (RWA) problem. The RSA problem in optical networks with SLICE architecture is another focus of this dissertation.

In the domain of social networks, the focus of this dissertation is study of influence propagation in presence of active adversaries. It has been observed that individuals' decisions to adopt a product or innovation are often influenced by the recommendations of their friends and acquaintances. Motivated by this observation, the last few years have seen a number of studies on influence maximization problem in social networks. One major goal of several of these studies is identification of *k most influential nodes* in a network. It may be noted that most of the studies on influence propagation are geared toward a *non-adversarial environment*, where only one vendor (player) is attempting to influence the nodes of a social network to buy her product. However, in a realistic market scenario, most often there exists multiple players, each attempting to sell their competing products or innovations. This dissertation investigates the scenario where in a social network, the first player has already chosen a set of k nodes and the second player, with the knowledge of the choice of the first, attempts to identify a smallest set of nodes so that when the influence propagation process ends, the number of nodes influenced by the second player is larger than the number of nodes influenced by the first.

1.2 Contributions

The contributions of this dissertation can be categorized into four main fields of (i) robust design of Airborne Networks, (ii) fault tolerant design of wireless sensor networks, (iii) routing and spectrum allocation in spectrum sliced optical networks, and (iv) influence propagation in social networks in adversarial setting. The dissertation contributions in each field is described in the following parts.

1.2.1 Robust Design of Airborne Networks

The connectivity of the backbone network is an important metric in reliable and robust design of airborne networks. In this dissertation the problem of connectivity of airborne networks is studied in three different scenarios:

- Non-faulty: In non-faulty scenario it is assumed that all the network nodes (airborne platforms) are operational all the times during the network operation time. The *critical transmission range*, CTR, is defined to be the minimum transmission range of the ANPs to ensure that the dynamic network formed by the movement of the ANPs remains connected at all times. In this dissertation, the problem of computation of critical transmission range is studied. An algorithm to compute CTR when the flight paths are known is proposed. As a part of the design of this algorithm, a technique to compute the *dynamic topology* of the AN at any instance of time is developed.

- Faulty: In faulty scenario, it is considered that some of the network nodes are faulty, and faulty nodes are *spatially correlated* (or *region-based*). In other words, faulty nodes due to an attack are confined to a *region*. In this scenario, the design requirement is that the network should remain connected even if a region fails. The *critical transmission range in faulty scenario (CTR_f)* is defined to be the *smallest transmission range necessary to ensure network connectivity, irrespective of (a) the location of the fault region and (b) the time of the failure*. In this dissertation, an algorithm is proposed to compute CTR_f. As a part of design of this algorithm, techniques are developed to (i) compute all the fault regions that need to be considered to *ensure overall connectivity at all times*, and (ii) compute the set of dynamic nodes that might be affected by the *failure of a specific region at a specific time*.

- Delay tolerant: In the two previous scenarios the connectivity requirement is very strict and the backbone network is needed to be connected all the times. However, it may not be possible to equip the ANPs with radios with transmission range at least as large as the CTR. In such a scenario the network may operate in a disconnected mode for some part of time. On the other hand,

based on the type of data that should be transmitted between ANPs, data transmissions may be tolerant to some amount of delay. Hence, ANPs may not need to have end-to-end paths all the time but they should be able to transmit data to each other within bounded time. In this dissertation the problem of computation of critical transmission range in delay tolerant airborne networks is studied. More specifically, the critical transmission range in delay tolerant network (CTR_D) is defined to be the minimum transmission range necessary to ensure that every pair of nodes in the backbone network can communicate with each other within a bounded time. A solution is proposed to compute CTR_D.

1.2.2 Fault Tolerant Design of Wireless Sensor Networks

In this dissertation the problem of enhancing the fault tolerance capability of data gathering tree in sensor networks is investigated. Assuming that addition of each link to the tree involves a cost, the following problem is studied: least cost augmentation of the tree (TCA) so that even after failure of a single node or a pair of adjacent nodes, all the surviving nodes will remain connected to the sink node. It is proven that the least cost tree augmentation problem is NP-complete under both types of fault scenarios. Moreover, two approximation algorithms, one for single node failure and the other for a pair of adjacent node failure, with performance bounds of two and four are, respectively, proposed.

1.2.3 Routing and Spectrum Allocation in Spectrum Sliced Optical Networks

A thorough study of routing and spectrum allocation RSA problem in spectrum sliced optical networks is performed. In this dissertation, it is proved that RSA problem is NP-complete even when the optical network topology is as simple as a *chain* or a *ring*. The *Spectrum Constrained RSA* (SCRSA) problem is introduced where the goal is to satisfy as many requests as possible, subject to the constraint that only a finite size spectrum is available for satisfying connection requests. Then, approx-

imation algorithms are proposed for both the RSA and SCRSA problems when the network topology is a *binary tree* or a ring. Moreover, the on-line version of RSA problem is studied where the requests are not known in advance and an algorithm for the on-line version of RSA problem in the *ring network* with a bounded *competitive ratio* is developed. Finally, heuristics are provided for off-line and on-line RSA problem in the network with arbitrary topology and the effectiveness of the heuristics is measured with extensive simulation. Simulation results demonstrate that the heuristics significantly outperforms several other heuristics proposed recently for the RSA problem.

1.2.4 Influence Propagation in Social Networks in Adversarial Setting

A comprehensive study of infulence propagation problem in social networks has been conducted in this dissertation. A new influence propagation problem in an adversarial setting is introduced where the goal of the second player is to defeat the first within D time steps and least amount of cost (i.e., number of seed nodes). Also, NP-Hardness proof for the problem under two different influence propagation models is provided. Moreover, an approximation algorithm for the problem with a tight performance bound of $O(\log(n))$ is proposed. Finally, the approximation algorithm is evaluated through experiments using collaboration network data.

Several parts of the research work presented in this dissertation, which have been already published, accepted or are under review in international conferences and journals are listed as follow:

- S. Shirazipourazad, A. Sen, and S. Bandyopadhyay, Fault-tolerant Design of Wireless Sensor Networks with Directional Antennas, accepted in Pervasive and Mobile Computing.

- S. Shirazipourazad, Z. Derakhshandeh, and A. Sen, Analysis of On-line Routing and Spectrum Allocation in Spectrum-sliced Optical Networks, IEEE International Conference on Communications (ICC), Budapest, Hungary, 2013.

- S. Shirazipourazad, C. Zhou, Z. Derakhshandeh, and A. Sen, On Routing and Spectrum Allocation in Spectrum-sliced Optical Networks, IEEE International Conference on Computer Communications (INFOCOM)(Mini-Conference), Turin, Italy, 2013.

- S. Shirazipourazad, A. Sen, and S. Bandyopadhyay, Fault-tolerant Design of Wireless Sensor Networks with Directional Antennas, International Conference on Distributed Computing and Networking (ICDCN), Mumbai, India, 2013 (Best Paper Award).

- S. Shirazipourazad, B. Bogard, H. Vachhani, A. Sen, and P. Horn, Influence Propagation in Adversarial Setting: How to Defeat Competition with Least Amount of Investment", in Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM), Maui, HI, USA, 2012

- S. Shirazipourazad, P. Ghosh, and A. Sen, On Connectivity of Airborne Networks in Presence of Region-based Faults, IEEE Military Communications Conference (MILCOM), Baltimore, MD, 2011.

1.3 Dissertation Outline

The rest of the dissertation is organized as follows: In Chapter 2, the reliable and robust design of airborne networks has been studied and several problems on airborne network connectivity have been discussed. In Chapter 3, the tree connectivity augmentation problem in sensor networks has been presented. Chapter 4 studies

the routing and spectrum allocation problem in optical networks. In chapter 5, the identification of influential nodes in social networks in adversarial setting has been discussed. Finally, Chapter 6 concludes the dissertation along discussions of future research directions.

Chapter 2

ROBUST AIRBORNE NETWORK DESIGN

An Airborne Network (AN) is a mobile ad hoc network that utilizes a heterogeneous set of physical links (RF, Optical/Laser and SATCOM) to interconnect a set of terrestrial, space and highly mobile airborne platforms (satellites, aircrafts and Unmanned Aerial Vehicles (UAVs)). Airborne networks can benefit many civilian applications such as air-traffic control, border patrol, and search and rescue missions. The design, development, deployment and management of a network where the nodes are mobile are considerably more complex and challenging than a network of static nodes. This is evident by the elusive promise of the Mobile Ad-Hoc Network (MANET) technology where despite intense research activity over the previous years, mature solutions are yet to emerge [3, 4]. One major challenge in the MANET environment is the unpredictable movement pattern of the mobile nodes and its impact on the network structure. In case of an AN, there exists considerable control over the movement pattern of the mobile platforms. A senior Air Force official can specify the controlling parameters, such as the *location, flight path and speed* of the ANPs, to realize an AN with desired functionalities. Such control provides the designer with an opportunity to develop a topologically stable network, even when the nodes of the network are highly mobile. It is increasingly being recognized in the networking research community that the level of *reliability* needed for continuous operation of an AN may be difficult to achieve through *a completely mobile, infrastructure-less network* [1]. In order to enhance *reliability* and *scalability* of an AN, Milner *et al.* in [1] suggested the formation of a *backbone network* with Airborne Networking Platforms (ANPs). In order to deal with the reliability and scalability issues in an AN, we consider an architecture for an AN where a set of ANPs form the *backbone* of the AN. This set of ANPs may be viewed as *mobile base stations* with *predictable and well-structured*

flight paths and the combat aircrafts on a mission as *mobile clients*. We want that the backbone network remains connected all the times even though the topology of the network changes with the movement of the ANPs. Network connectivity can be easily achieved if the transmission range of the ANPs is very large. However large transmission range also implies high power consumption. In order to minimize power consumption and hence extend network lifetime, we would like to find the smallest transmission range to ensure network connectivity. We define the *critical transmission range* (CTR) to be the minimum transmission range of the ANPs to ensure that the dynamic network formed by the movement of the ANPs remains connected at all times. We present an algorithm to compute CTR when the flight paths are known. As a part of design of this algorithm, we develop techniques to compute the *dynamic topology* of the AN at any instance of time.

Using CTR as the transmission range of all nodes, the network is connected as long as all the network nodes (i.e., the ANPs) are operational. However, the ANPs are vulnerable to Electromagnetic Pulse (EMP) attacks or jamming. Such an attack will impact specific geographic regions at specific times and if an ANP is within the fault region during the time of attack, it will not be able to carry out its normal communication functions. We will refer to these ANPs as faulty nodes of the network. In this research, we also consider the AN scenario where some of the network nodes are faulty. We consider faulty nodes are *spatially correlated* (or *region-based*), that is faulty nodes due to an attack are confined to a *region*. We want that the network remains connected *irrespective of location of the fault region* and the *time of failure*.

We define *critical transmission range in faulty scenario (CTR_f)* to be the *smallest transmission range necessary to ensure network connectivity, irrespective of (a) the location of the fault region and (b) the time of the failure*. We would like to find CTR_f. As a part of design of this algorithm, we develop techniques to (i) compute

all the fault regions that need to be considered to *ensure overall connectivity at all times*, (ii) compute the set of dynamic nodes that might be affected by the *failure of a specific region at a specific time*, and finally, (iii) compute CTR_f.

In previous problems the connectivity requirement is very strict and the backbone network is needed to be connected all the times. However, it may not be possible to equip the ANPs with radios with transmission range at least as large as the CTR. In such a scenario the network may operate in a disconnected mode for some part of time. On the other hand, based on the type of data that should be transmitted between ANPs, data transmissions may be tolerant to some amount of delay. Hence, ANPs may not need to have end-to-end paths all the time but they should be able to transmit data to each other within bounded time. These requirements lead us to study the problem of computation of critical transmission range in delay tolerant airborne networks. More specifically, the critical transmission range in delay tolerant network (CTR_D) is defined to be the minimum transmission range necessary to ensure that every pair of nodes in the backbone network can communicate with each other within a bounded time. In this chapter we formulate CTR_D and propose a solution to compute CTR_D.

The rest of the chapter is organized as follows. We discuss related work in section 2.1. In section 2.2 we present the AN architecture. We present dynamic topology computation of the AN in section 2.3. In section 2.4 we present an algorithm to compute CTR in fault free scenario. We discuss the faulty scenario and propose an algorithm to compute CTR_f in section 2.5. Connectivity problem in delay tolerant airborne network is formulated and studied in section 2.6. Experimental results are presented in section 2.7. The section 2.8 concludes the chapter.

2.1 Related Works

Due to the Joint Aerial Layer Networking (JALN) activities of the U.S. Air Force, design of a robust and resilient Airborne Network (AN) has received considerable attention in the networking research community in recent years. It has been investigated that the flat ad hoc networks have limitations with respect to data transmission, distance, interference and scalability [1, 5, 6]. Accordingly, [2, 1, 5] suggested the addition of a mobile wireless backbone of base stations (analogous to cellular telephony or the Internet backbones), in which topologies and mobility can be controlled for purposes of assured communications.

There exists considerable amount of studies on topology control using power control in MANETs [7, 8, 9, 10, 5]. The goal of the proposed algorithms is to assign power values to the nodes to keep the network connected while reducing the power consumption. The authors of [7, 8] proposed distributed heuristics for power minimization in mobile ad hoc networks and offer no guarantees on the worst case performance. Santi in [10] studied the minimum transmission range required to ensure network connectivity in mobile ad hoc networks. He proved that the critical transmission range for connectivity (CTR) is $c\sqrt{\frac{\ln n}{\pi n}}$ for some constant c where mobility model is obstacle free and nodes are allowed to move only within a certain bounded area. In these studies the mobility patterns are not known unlike the problems studied in this chapter where it is assumed that the flight paths of ANPs are predictable. Moreover, this research studies the computation of minimum transmission range in presence of region-based faults and in delay tolerant scenario where it is not the case in previous studies.

In recent times, there has been considerable interest in studying localized, i.e., spatially correlated or region-based faults in networks [11, 12, 13, 14]. In order to

capture the notion of locality in measuring the fault-tolerance capability of a network, a new variant of connectivity metric called region-based connectivity was first introduced by Sen et. al. [11]. Region-based connectivity, for multiple spatially correlated faults, has been studied in [12]. The region-based connectivity of a network can be informally defined to be the minimum number of nodes that has to fail within any region of the network before it is disconnected. Neumayer et. al [13] gave an analysis on identifying the most vulnerable parts of the network when the faults are geographically correlated. That is, the analysis gives locations of disasters that would have the maximum disruptive effect on the network in terms of capacity and connectivity. In [14] Neumayer et. al. evaluates average two-terminal reliability of a fiber-optic network in polynomial time under the presence of such geographically correlated faults. The networks studied in [11, 12, 14, 13] are all static. However, ANs under study in this research are dynamic.

There may be times that the backbone network may have to operate in a disconnected mode. The last few years have seen considerable interest in the networking research community in delay tolerant networks (DTN) design [15]. The authors of [16] survey challenges in enhancing the survivability of mobile wireless networks. This paper mentions that one of the aspects that can significantly enhance network survivability is the design for end-to-end communication in environments where the path from source to destination is not wholly available at any given instant of time. In this design adjusting the transmit power of the nodes plays an important role. Existing DTN research mainly focuses on routing problem in DTN [17, 18]. The paper [19] provides a survey on routing algorithms for DTN. For the routing algorithms to be effective, every pair of nodes should be able to communicate with each other over time. Therefore, the time evolving DTN should be connected over time. Few papers [20, 21] have studied the problem of topology control in DTNs. In these papers, the

time evolving network is modeled by space-time graph and it is assumed that the space-time graph is initially connected and the problem is to find the minimum cost connected subgraph of the original graph.

These papers have not studied the computation of minimum transmission range of nodes in DTN networks such that the time evolving network is connected over time and to the best of our knowledge, this is the first research work that studies this problem.

2.2 System Model and Architecture

In the previous section, we argued that the level of *reliability* needed for continuous operation of an AN may be difficult to achieve through *a completely mobile, infrastructure-less network* and wherever possible a *backbone network* with Airborne Networking Platforms (ANPs) should be formed to enhance reliability. In order to achieve this goal, we propose an architecture of an AN where a set of ANPs form a backbone network and provide reliable communication services to combat aircraft on a mission. In this architecture, the nodes of the backbone networks (ANPs) may be viewed as *mobile base stations* with *predictable and well-structured flight paths* and the combat aircrafts on a mission as *mobile clients*. A schematic diagram of this architecture is shown in Fig. 2.1. In the diagram, the black aircrafts are the ANPs forming the infrastructure of the AN (although in Fig. 2.1, only aircrafts are shown as ANPs, the UAVs and satellites can also be considered as ANPs). We assume that the ANPs follow a circular flight path. The circular flight paths of the ANPs and their coverage area (shaded spheres with ANPs at the center) are also shown in Fig. 2.1. Thick dashed lines indicate the communication links between the ANPs. The figure also shows three fighter aircrafts on a mission passing through space known as *air corridor*, where network coverage is provided by ANPs 1 through 5. As the fighter aircrafts move along their flight trajectories, they pass through the coverage area of

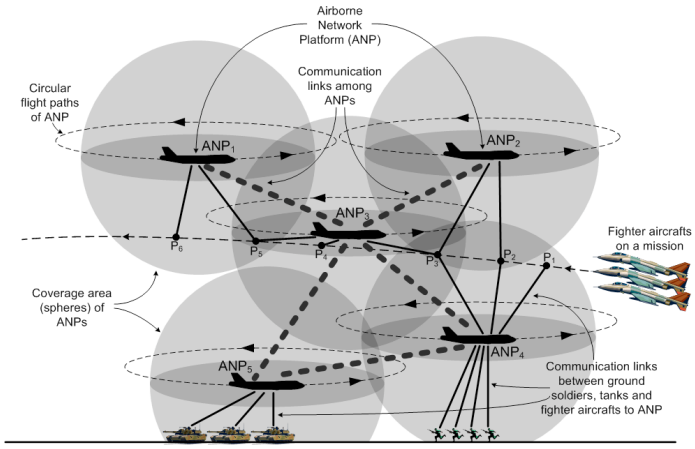


Figure 2.1: A Schematic View of an Airborne Network

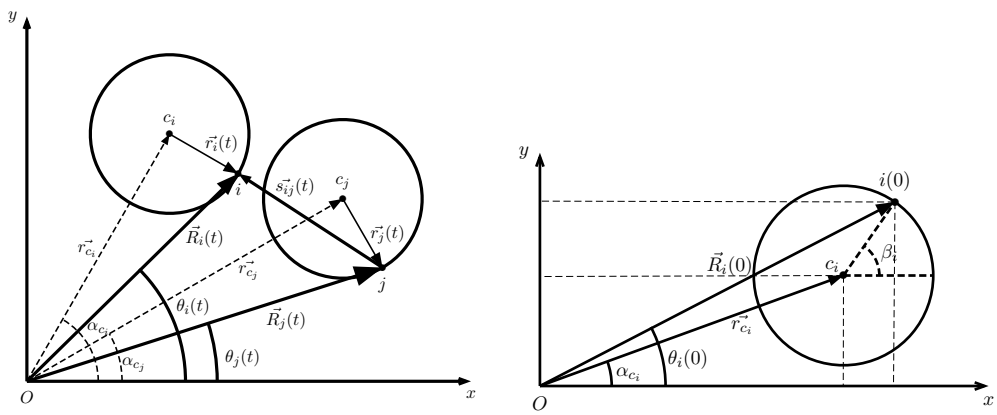
multiple ANPs and there is a smooth hand-off from one ANP to another when the fighter aircrafts move from the coverage area of one ANP to that of another. At points P1, P2, P3, P4, P5 and P6 on their flight path in Fig. 2.1, the fighter aircrafts are connected to the ANPs (4), (2, 4), (2, 3, 4), (3), (1, 3) and (1), respectively.

In this research, we make a simplifying assumption that two ANPs can communicate with each other whenever the distance between them does not exceed the specified threshold (transmission range of the on board transmitter). We are well aware of the fact that successful communication between two airborne platforms depends not only on the distance between them, but also on various other factors such as (i) the line of sight between the platforms [22], (ii) changes in the atmospheric channel conditions due to turbulence, clouds and scattering, (iii) the banking angle, the wing obstruction and the dead zone produced by the wake vortex of the aircraft [23] and (iv) Doppler effect. Moreover, the transmission range of a link is not a constant and is impacted by various factors, such as transmission power, receiver sensitivity, scattering loss over altitude and range, path loss over propagation range, loss due to turbulence and the transmission aperture size [23]. However, the distance between the ANPs remains a very important parameter in determining whether communication

between the ANPs can take place, and as the goal of this research is to understand the basic and fundamental issues of designing an AN with twin invariant properties of coverage and connectivity, we feel such simplifying assumptions are necessary and justified. Once the fundamental issues of the problem are well understood, factors (i) - (iv) can be incorporated into the model to obtain a more accurate solution.

For simplicity of analysis, we make two more assumptions. We assume that (i) all ANPs are flying at the same altitude and (ii) they follow a circular flight path. The first assumption allows us to reduce the problem from three dimension to two. However, none of these two assumptions are critical and our analysis technique can easily be extended to scenarios where the ANPs are not flying at the same altitude and they are not following a circular flight path. As a consequence of assumption (i), we can view the n backbone nodes (ANPs) as moving points on a 2 dimensional plane. Let $(x_i(t), y_i(t))$ be the coordinates of the node i at time t. The network of flying ANPs gives rise to a *dynamic graph* $G(t) = (V, E(t))$ where $V = \{1, 2, \ldots, n\}$ is the set of nodes indexed by the ANPs and $E(t)$ is the set of edges at time t. There is an edge between two nodes if their Euclidean distance, s_{ij} is less than the transmission range Tr at time t, i.e., $E(t) = \{(i, j)|s_{ij}(t) < Tr\}$. It may be noted that the dynamic graph $G(t) = (V, E(t))$ is completely defined by the following five controlling parameters.

1. a set of points $\{c_1, c_2, \ldots, c_n\}$ on a two dimensional plane (representing the centers of circular flight paths),
2. a set of radii $\{r_1, r_2, \ldots, r_n\}$ representing the radii of circular flight paths,
3. a set of points $\{p_1, p_2, \ldots, p_n\}$ representing the initial locations of the platforms
4. a set of velocities $\{v_1, v_2, \ldots, v_n\}$ representing the speeds of the platforms, and
5. transmission range Tr of the transceivers on the airborne platforms.



(a) (b)

Figure 2.2: (a) Initial phase angle β_i of point i; at time 0 point is shown as $i(0)$, (b) Vector representations ($\mathbf{R_i}(t)$ and $\mathbf{R_j}(t)$) of two points i and j at time t moving along two circular orbits: $r_{c_i} = 15,\ r_{c_j} = 27,\ \angle c_i O x = \alpha_{c_i} = \frac{\pi}{3},\ \angle c_j O x = \alpha_{c_j} = \frac{\pi}{6}$

In next section we explain the computation of dynamic topology of graph $G(t) = (V, E(t))$ when all five controlling parameters are given.

2.3 Dynamic Topology Computation

In this section we answer the following question. Given all five problem parameters including the transmission range of the ANPs, how do you determine if the resulting dynamic graph is connected at all times?

Suppose that two ANPs, represented by two points i and j (either in two or in three dimensional space, the two dimensional case corresponds to the scenario where the ANPs are flying at same altitude) are moving along two circular orbits with centers at c_i and c_j with orbit radius r_i and r_j as shown in Fig. 2.2(a) with velocities v_i and v_j (with corresponding angular velocities ω_i and ω_j), respectively.

A moving node i is specified by the radius vector $\mathbf{R_i}(t)$ directed from some origin point O, and similarly $\mathbf{R_j}(t)$ for point j. Therefore the distance $s_{ij}(t)$ between the nodes $i - j$ at time t is given by:

$$s_{ij}^2(t) = (\mathbf{R}_i(t) - \mathbf{R}_j(t))^2 = R_i^2(t) + R_j^2(t) - 2\mathbf{R}_i(t) \cdot \mathbf{R}_j(t) \qquad \text{(i)}$$

As mentioned earlier, we have assumed that the communication between the ANPs is possible if and only if the Euclidean distance between them does not exceed the communication threshold distance Tr. This implies that the link between the nodes i and j is alive (or active) when

$$s_{ij}(t) \leq Tr \tag{ii}$$

In the analysis that follows, we have assumed that ANPs are flying at the same altitude, i.e., we focus our attention to the two dimensional scenario. However, this analysis can easily be extended to the three dimensional case to model the scenario where the ANPs are flying at different altitude. In this case we can view the ANPs as points on a two-dimensional plane moving along two circular orbits, as shown in Fig. 2.2(a). In Fig. 2.2(a), the vectors from the origin O to the centers of the orbits c_i and c_j are given as $\mathbf{r_{c_i}}$ and $\mathbf{r_{c_j}}$. The cartesian co-ordinates of the centers can be readily obtained as $\mathbf{r_{c_i}} = (r_{c_i} cos\ \alpha_{c_i}, r_{c_i} sin\ \alpha_{c_i})$ and $\mathbf{r_{c_j}} = (r_{c_j} cos\ \alpha_{c_j}, r_{c_j} sin\ \alpha_{c_j})$. Accordingly, $\mathbf{R_i}(t)$ can be expressed in polar coordinates: $(R_i(t), \theta_i(t))$ with respect to origin point O, as shown in Fig. 2.2(a), and similarly for $\mathbf{R_j}(t)$. The initial location of the points $\mathbf{R_i}(0)$ and $\mathbf{R_j}(0)$ are given. From Fig. 2.2(b), the phase angle β_i for node i with respect to the center of orbit c_i, can be calculated as (by taking projection on the axes):

$$tan\ \beta_i = \frac{R_i(0) cos\ \theta_i(0) - r_{c_i} cos\ \alpha_{c_i}}{R_i(0) sin\ \theta_i(0) - r_{c_i} sin\ \alpha_{c_i}} \tag{iii}$$

From Fig. 2.2(a),

$$\mathbf{R}_i(t) = \mathbf{r}_{c_i} + \mathbf{r}_i(t) \tag{iv}$$

where $\mathbf{r}_i(t) = (r_i \cos\ (\beta_i + \omega_i t), r_i \sin\ (\beta_i + \omega_i t))$ (since angle made by i at time t w.r.t. c_i is given by $(\beta_i + \omega_i t)$). Therefore, the angle between $\mathbf{r}_i(t)$ and $\mathbf{r}_{c_i}$ is $(\beta_i - \alpha_{c_i} + \omega_i t)$. Hence,

$$R_i^2(t) = r_{c_i}^2 + r_i^2 + 2 r_{c_i} r_i \cos\ (\beta_i - \alpha_{c_i} + \omega_i t) \tag{v}$$

Now taking the projection of $\mathbf{R}_i(t) = \mathbf{r}_{c_i} + \mathbf{r}_i(t)$ on the x and y axes, we get

$$R_i(t)\cos\theta_i(t) = r_{c_i}\cos\ \alpha_{c_i} + r_i\cos\ (\beta_i + \omega_i t), \tag{vi}$$

$$R_i(t)\sin\ \theta_i(t) = r_{c_i}\sin\ \alpha_{c_i} + r_i\sin(\beta_i + \omega_i t) \tag{vii}$$

Recalling $\cos(A - B) = \cos A\cos B + \sin A\sin B$, and simplifying we get

$$\begin{aligned} R_i(t)R_j(t)\cos(\theta_i(t) - \theta_j(t)) &= r_{c_i}r_{c_j}\cos\ \alpha_{c_ic_j} + r_ir_j\cos(\beta_{ij} + (\omega_i - \omega_j)t) \\ &+ r_{c_i}r_j\cos(\alpha_{c_i} - \beta_j - \omega_j t) \\ &+ r_{c_j}r_i\cos(\alpha_{c_j} - \beta_i - \omega_i t) \end{aligned} \tag{viii}$$

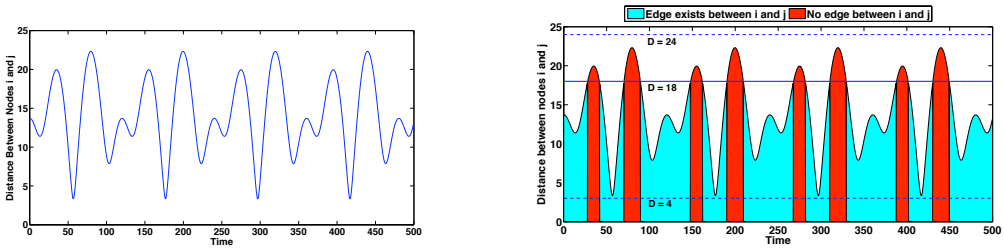
where $\alpha_{c_{ij}} = \alpha_{c_i} - \alpha_{c_j}$ and $\beta_{ij} = \beta_i - \beta_j$. Combining equation i with equations v and viii, we have:

$$\begin{aligned} s_{ij}^2(t) &= r_{c_i}^2 + r_i^2 + 2r_{c_i}r_i\cos(\beta_i - \alpha_{c_i} + \omega_i t) + r_{c_j}^2 + r_j^2 + 2r_{c_j}r_j\cos(\beta_j - \alpha_{c_j} + \omega_j t) \\ &-2[\ r_{c_i}r_{c_j}\cos\alpha_{c_ic_j} + r_ir_j\cos(\beta_{ij} + (\omega_i - \omega_j)t) \\ &+r_{c_i}r_j\cos(\alpha_{c_i} - \beta_j - \omega_j t) + r_{c_j}r_i\cos(\alpha_{c_j} - \beta_i - \omega_i t)] \end{aligned} \tag{ix}$$

In equation ix, all parameters on the right hand side are known from the initial state of the system, and thus the distance $s_{ij}(t)$ between the nodes $i - j$ at any time t can be obtained. If the ANPs move at the same velocity, i.e., $\omega_i = \omega_j = \omega$ for all i, j and the radius of the circular orbits are identical, i.e., $r_i = r_j = r$ for all i, j, and the above expression simplifies to:

$$\begin{aligned} s_{ij}^2(t) &= r_{c_i}^2 + r^2 + 2r_{c_i}r\cos(\beta_i - \alpha_{c_i} + \omega t) + r_{c_j}^2 + r^2 + 2r_{c_j}r\cos(\beta_j - \alpha_{c_j} + \omega t) \\ &-2[\ r_{c_i}r_{c_j}\cos\ \alpha_{c_ic_j} + r^2\cos\beta_{ij} \\ &+r_{c_i}r\cos(\alpha_{c_i} - \beta_j - \omega t) + r_{c_j}r\cos(\alpha_{c_j} - \beta_i - \omega t)] \end{aligned} \tag{x}$$

An example of a plot of equation (ix) (generated using MATLAB) is shown in Fig. 2.3(a) with communication threshold distance $Tr = 18$. This implies that



(a) (b)

Figure 2.3: Effect of the distance between nodes on the existence of the communication link between them; (a)Distance between two points i and j as a function of time, (b)Active (Blue/Light gray)/Inactive (Red/ Dark gray) times of the link between i and j with transmission range $Tr = 18$

the link between the nodes i and j exists, when the distance between them is at most 18 and the link does not exist otherwise. This is shown in Fig. 2.3(b). The red(dark gray) part indicates the time interval when the link is *inactive*(or dead) and the blue(light gray) part indicates when it is *active* (or live).

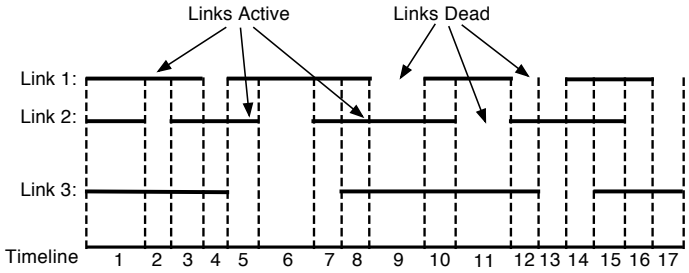


Figure 2.4: Active/Inactive time interval of each link and interval intersection projections on the time line

Thus using equation (ix) and comparing the distance between any two nodes with the communication threshold Tr, we can determine active/inactive times of all links. This can be represented as intervals on a time line as shown in Fig. 2.4. By drawing projections from the end-points of the active/inactive times of each link on the time line, we can find out all the links that are active during an interval on time line. As shown in Fig. 2.4, links 1, 2 and 3 are active in interval 1; links 1 and 3 are active in interval 2, links 1, 2 and 3 are active in interval 3 and so on. Once we know

all the links that are active during a time interval, we can determine if the graph is connected during that interval using any algorithm for computing graph connectivity [24]. By checking if the graph is connected at all intervals, we can determine if the graph is connected at all times, when the ANPs are moving at specified velocities.

We note that based on equation (ix), s_{ij} is periodic if every pair of velocities ω_i and ω_j are *commensurate*, i.e. ω_i/ω_j is a rational number [25]. Therefore, the network topologies will be repeated periodically and it is enough to check network connectivity in one period.

If the problem parameters (1) through (5) are specified, we can check if the dynamic graph is connected at all times following these two steps. In the first step, we determine the lifetime (active/inactive intervals) of every link between every pair of nodes i and j by comparing $s_{ij}(t)$ with Tr and finding the time points that the state of a link changes. Let $L(Tr) = \{e_1, e_2, \ldots, e_l\}$ denote the set of events e_is that state of a link changes when transmission range is Tr; $L(Tr)$ is sorted in increasing order of the time of the events. Hence, between two consecutive events e_i and e_{i+1} that happen at times t_i and t_{i+1} the set of active links is unchanged. Algorithm 1 shows the details of computing $L(Tr)$. In the second step we check the graph connectivity in each interval $[t_i, t_{i+1})$ for all $0 \leq i \leq l-1$ using connectivity checking algorithm in [26]. t_0 shows current time (starting point). Step 2 is described in detail in Algorithm 2.

Algorithm 1 Link Lifetime Computation

Input: (i) a set of points $\{c_1, c_2, \ldots, c_n\}$ representing the centers of circular flight paths, (ii) a set of radii $\{r_1, r_2, \ldots, r_n\}$ representing the radii of circular flight paths, (iii) a set of points $\{p_1, p_2, \ldots, p_n\}$ representing the initial locations of the platforms, (iv) a set of velocities $\{v_1, v_2, \ldots, v_n\}$ representing the speeds of the platforms.

Output: $L(Tr)$: an ordered set of events that the state of a link changes from active to inactive or inactive to active.

1: $L(Tr) \leftarrow \emptyset$
2: **for all** pairs i, j **do**
3: Compute l to be the set of time points t such that $s_{ij}(t) = Tr$ (equation ix) over a period of time, to find the instances of times t where the state of the link (i, j) changes. If $s_{ij}(t) = Tr$ and is $s_{ij}(t)$ increasing at t, it implies that the link dies at t, and if $s_{ij}(t)$ decreasing at t, it implies that the link becomes active at t.
4: **for all** $l_k \in l$ **do**
5: Find the position of l_k in $L(Tr)$ using binary search and Add the event into $L(Tr)$. ($L(Tr)$ is sorted in increasing order)
6: **end for**
7: **end for**

Algorithm 2 Checking Connectivity of Airborne Network

Input: $L(Tr)$

Output: *true* if graph is connected all the time; otherwise *false*.

1: **for all** $l_i \in L(Tr)$ **do**
2: Check if the AN graph is connected with the set of live links during interval $[l_i, l_{i+1})$. This can be done with the connectivity testing algorithm in [26]
3: **i**f AN graph is not connected, **r**eturn $false$
4: **end for**
5: **r**eturn $true$

Let n be the number of ANPs. The first loop of Algorithm 1 is executed for $O(n^2)$ times. The number of iterations of the inner loop depends on the number of the solutions of $s_{ij}(t) = Tr$. For the case that ANPs move at the same velocity, i.e., $\omega_i = \omega_j = \omega$ it is obvious that equation (x) is periodic and length of one periodic interval is $2\pi/\omega$. So, it is enough to execute Algorithm 1 for one period $[t_0, t_0 + 2\pi/\omega)$. In this case, equation (x) can be written as $A\cos(\omega t) + B\sin(\omega t) = \sqrt{A^2 + B^2}\sin(\varphi + \omega t)$ where A, B and φ are constants and can easily be obtained from equation (ix). In this case, the equation $s_{ij}(t) = Tr$ can have at most two solutions and the solutions can be found in constant time. Therefore, for every link, the timeline is divided into at most three segments in one period and the size of the set of intervals, $|L(Tr)|$ is $O(n^2)$; also, the time complexity of the binary search is $O(\log n^2)$. So, the total time complexity of Algorithm 1 is $O(n^2 \log n)$. Even when the velocities of the ANPs are different, s_{ij} remains periodic if every pair of velocities ω_i and ω_j are *commensurate*, i.e. ω_i/ω_j is a rational number [25]. In this case also we need to solve $s_{ij}(t) = Tr$ for one period only. Otherwise, equation (ix) is not periodic and we need to consider a period of time between t_0 and finish time t_f and find the solutions in that period. For the sake of simplicity, we assume that the ANPs move at the same speed. The running time of connectivity testing algorithm in [26] is $O(n^2)$. Also, as $|L(Tr)| = O(n^2)$ time complexity of Algorithm 2 is $O(n^4)$.

2.4 Computation of Critical Transmission Range in Fault Free Scenario

It is conceivable that even if the network topology changes due to movement of the nodes, some underlying structural properties of the network may still remain invariant. A structural property of prime interest in this context is the *connectivity* of the dynamic graph formed by the ANPs. We want the ANPs to fly in such a way, that even though the links between them are established and disestablished over time, the underlying graph remains connected at all times. We define *critical transmission*

range (CTR) to be the smallest transmission range necessary to ensure network graph $G(t)$ is always connected. We would like to determine CTR. In this case, the problem will be specified in the following way. Given controlling parameters 1, 2, 3 and 4, what is the minimum transmission range of the ANPs so that the resulting graph is *connected* at all times?

In the previous section we explained how we check network connectivity when all five parameters are given. The maximum transmission range of an ANP Tr_{max} is known in advance. In order to compute CTR we can conduct a binary search within the range $0 - Tr_{max}$ and we can determine the smallest transmission range that will ensure a connected AN during the entire operational time when all other problem parameters have already been determined. The binary search adds a factor of $\log Tr_{max}$ to the complexity of Algorithms 1 and 2.

2.5 Computation of Critical Transmission Range in Faulty Scenario

The CTR computed in previous section may not guarantee the connectivity of backbone network when some of ANPs fail. In this section, we consider the AN scenario where some of the network nodes are faulty and we compute *critical transmission range in faulty scenario(CTR_f)* which is defined to be the *smallest transmission range necessary to ensure network connectivity, irrespective of (a) the location of the fault region and (b) the time of the failure.* First we describe the fault model used in this research. Also, we identify the challenges that one has to confront, in order to find the CTR_f.

2.5.1 Fault Model

As we mentioned before, our focus is on *spatially correlated* (or *region-based*) faults such as Electromagnetic Pulse (EMP) attacks or jamming. Spatially correlated or region-based faults imply that the faulty nodes due to an attack are confined to a *geographic area.* In a two dimensional deployment area, a region can be viewed as a

circular area with radius R (in three dimensional space it can be viewed as a sphere with radius R). In our model, when a region is under attack and consequently fails at time t, some or all the ANPs within that region at time t also fail. In this version of the model, we also make an assumption that only one region can fail at any one time. Fig. 2.5 shows five ANPs moving on a two dimensional plane and a faulty region (red circle, centered at point P) at time t. Since ANPs 4 and 5 are within the fault region at time t, we assume that these nodes are damaged and no longer can be viewed as part of the backbone network. It may be noted that both the location of the center of the fault circle, P, as well as the time of attack, t, play a critical role in determining the impact of the attack on the backbone network.

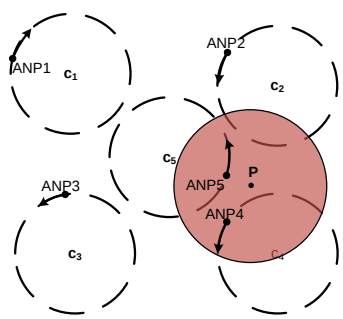


Figure 2.5: ANPs on a circular flight path on a 2D-plane with a fault region

2.5.2 Problem Formulation and Design Challenges

In faulty scenario the connectivity problem is defined in the following way: Given the controlling parameters 1, 2, 3 and 4 (defined in Section 2.2) as well as the radius of a region R, what is the *smallest transmission range necessary to ensure network connectivity, irrespective of (a) the location of the fault region and (b) the time of the failure*. In other words, the problem is how to compute CTR_f.

One can easily recognize the complexity of the problem by noting that potentially there could be an infinite number of locations for point P and infinite choices for attack time t. In our analysis we show that although there could be an infinite

number of choices of P and t, we need to consider only a small subset of them to correctly determine CTR_f. The tasks that need to be performed before a solution to the problem is found can be listed as follows:

- Computation and comprehension of the dynamic topology of the backbone network (in a fault-free scenario) as it changes with movements of the ANPs.
- How many regions (locations of point P) and instances of attack time t should be considered?
- How to determine the ANPs that are damaged when an attack takes place in location P at time t?

In section 2.3 we described the computation of the dynamic topology of the backbone network (in a fault-free scenario). In the following subsections we describe our techniques to deal with the second and third challenges and to compute CTR_f.

2.5.3 Regions to Examine

The authors in [11] introduced the notion of *region-based* faults and introduced a new metric, *region-based connectivity*, to measure the fault-tolerance capability of a network under the region-based fault model. Region-based connectivity of a network is defined to be the minimum number of nodes that has to fail in any region of the network before it is disconnected. In this study, a region is defined to be a circle of radius R. With this definition of a region, the number of potential regions could be infinite. The authors in [11] proved that in a static wireless network, only a limited number of distinct regions need to be examined to compute the region-based connectivity. They showed that it is enough to consider the regions centered at the intersection points of the circles centered at the nodes with radius R. Although the AN is dynamic, if we take a snapshot of the network at some instance of time t,

the AN can be viewed as a static network with a specific topology and nodes in specific locations on the plane. The vulnerability zone of a node i, $VZ_i(t)$, is defined to be a circular region centered at the location of node i at time t with radius R. The motivation for this definition of the vulnerability zone of node i is the following. If the center of the fault region is within the vulnerability zone of ANP_i (node i), then the ANP_i is likely to be damaged. The vulnerability zones of ANPs are shown in Fig. 2.6. Since there is no discernible difference between a static sensor network considered in [11] and a snapshot of an AN at a specific instance of time t, using the analysis presented in [11], we can conclude that it is enough to examine only the regions centered at the intersection points (I-points) of the vulnerability zones of the ANPs. The vulnerability zones of two ANPs and their intersection points are shown in Fig. 2.6. If a VZ_i does not have intersection with any other node's vulnerability zone, an I-point is considered at the location of the node i.

Since the ANPs are mobile, the location of the intersection points of their vulnerability zones also changes with time. Each pair of ANPs will have at most two intersection points. Since there are only $n(n-1)/2$ pairs of nodes, at most $n(n-1)$ intersections points can exist at any given time (it may be noted that depending on flight path of a pair of ANPs, their vulnerability zones may never intersect). We define a set of $n(n-1)+n$ I-points, $\mathcal{I} = \{I^1_{(1,2)}, I^2_{(1,2)}, I^1_{(1,3)}, I^2_{(1,3)}, \ldots, I^1_{(n-1,n)}, I^2_{(n-1,n)}, I_1, I_2, \ldots, I_n\}$, where the $I^1_{(i,j)}$ and $I^2_{(i,j)}$ are the intersection points of the vulnerability zones VZ_i and VZ_j. We will use the notation $I^1_{(i,j)}(t)$ and $I^2_{(i,j)}(t)$ to denote the locations of $I^1_{(i,j)}$ and $I^2_{(i,j)}$ at time t. Similarly, $I_i(t)$ will denote the location of node i at time t. Based on the results presented in [11], it is known that at any point of time t it is sufficient to examine only the regions centered at the I-points in $\mathcal{I}$. In the rest, we use $I_{(i,j)}$ to denote both $I^1_{(i,j)}$ or $I^2_{(i,j)}$.

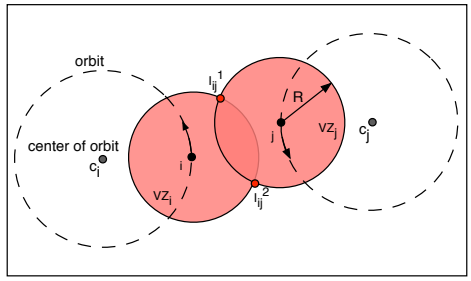


Figure 2.6: I_{ij}^1 and I_{ij}^2 are intersection points of VZ_i and VZ_j at time t.

For every two nodes i and j, $VZ_i(t)$ and $VZ_j(t)$ intersect iff $s_{ij}(t) \leq 2R$. In this case we say that the *region* centered at intersection point $I_{(i,j)}$ *exists* at time t; otherwise, it does not, i.e., there exists no region that can cover both nodes i and j at time t. It may be noted that due to the mobility of the ANPs, $I_{(i,j)}$ may exist at some point of time t and may not exist at some other point of time t'. By checking the condition $s_{ij}(t) \leq 2R$, we can determine the intervals on the timeline when $I_{(i,j)}$ exists for each pair of nodes i and j; i.e., we can compute *existence* intervals of each I-point on the timeline. Let $T(f) = \{(t_f^1, t_f^2), \ldots, (t_f^{k-1}, t_f^k)\}$ be the set of existence intervals of I-point $f \in \mathcal{I}$ where the first element in every pair (t_f^j, t_f^{j+1}) is the start time and the second one is the finish time of the j- th existence interval. If in a time interval $(t_{I_i}^j, t_{I_i}^{j+1})$, VZ_i does not have intersection with any other ANP's vulnerability zone then a region centered at I_i should be considered, i.e, $(t_{I_i}^j, t_{I_i}^{j+1}) \in T(I_i)$. Without loss of generality we can assume that the region centered at the point I_i exists all the time and it only covers node i. The computation of the intervals on the timeline when $I_{(i,j)}$ exists (or does not exist), for each pair of nodes i and j, can be carried out by an algorithm similar to Alg. 1 presented earlier. The only differences are (i) the value of t that satisfies the equation $s_{ij}(t) = 2R$ should be computed instead of the value of t that satisfies the equation $s_{ij}(t) = Tr$, (ii) since there is no need to combine existence interval information of one pair of nodes $(I_{(i,j)})$ with another pair, the binary search in step 5 of Alg. 1 is not needed.

2.5.4 Computation of the Damaged ANPs in a Fault Region

After finding the existence intervals of I-points we want to find the set of nodes that might be damaged by the failure of a region centered at an I-point when it exists. A node might be damaged by failure of a region if the Euclidean distance between the center of the region and the node is less than R. As explained in part 2.5.3 the regions centered at I-point $I_i \in \mathcal{I}$s only can destroy node i. Since, we know the locations of each pair of nodes i and j at time t, we can compute $VZ_i(t)$ and $VZ_j(t)$, and hence $I^1_{(i,j)}(t)$ and $I^2_{(i,j)}(t)$, the intersection points of $VZ_i(t)$ and $VZ_j(t)$ at time t.

Once the location of each intersection $I_{(i,j)}$ in its existence intervals $\in T(I_{(i,j)})$ are known, we can find the nodes that might be damaged if the region centered at $I_{(i,j)}$ fails. For ease of notation we denote the set of I-points $\in \mathcal{I}$ as $F = \{f_1, f_2, \ldots, f_l\}$. $D_{ik}(t)$ denotes the distance between I-point $f_i \in F$ and node k at time t. For every I-point $f_i \in F$ in its existence interval $\in T(f_i)$, we find $D_{ik}(t)$ for all $k \in V$. Since we know the position of the nodes and I-points at any point of time, $D_{ik}(t)$ can be computed easily. If $D_{ik}(t) \leq R$, then the node k may be damaged due to the region failure f_i. From this calculation, we can find out the *time interval* when node k is vulnerable to a failure f_i. In other words, we can find out the time intervals when a node k is *covered* by the region centered at f_i (i.e., $D_{ik}(t) \leq R$). It may be noted that this time interval will be subinterval of the intersection points existence time interval. Accordingly, every existence interval $(t^j_{f_i}, t^{j+1}_{f_i}) \in T(f_i)$ is divided into a set of smaller subintervals such that each of these intervals identify a specific set of nodes that may be damaged if the region centered at f_i fails. Suppose that t_m be the mth interval of $T(f_i)$. We define a set $NT(f_i, t_m) = \{(t_{m1}, N_{m1}), (t_{m2}, N_{m2}), \ldots, (t_{mj}, N_{mj})\}$as the set of subintervals into which t_m is divided, where t_{mj} denotes the start time of the jth subinterval of t_m where at least a node enters the region or leaves the region and

N_{mj} denotes the set of nodes within the region centered at f_i in its jth subinterval. We need to compute $NT(f_i, t_m)$ for every region $f_i \in F$ and for all of its existence intervals. Based on $NT(f_i, t_m)$ we can draw a timeline, *region-coverage timeline* for each region centered at an I-point $f_i \in F$. Fig. 2.7 shows an example in which $NT(f_1, t_1) = \{(t_{11}, \{1, 2\}), (t_{12}, \{1, 2, 3\}), (t_{13}, \{1, 2\})\}$.

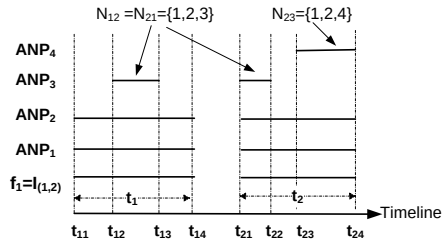


Figure 2.7: Region coverage timeline of the region centered at $f_1 = I_{(1,2)}$; The first timeline shows the availability intervals of f_1; i.e, $T(f_1) = \{t_1, t_2\}$.

*2.5.5 Computation of Critical Transmission Range in Faulty Scenario (*CTR$_f$*)*

In this section we propose an algorithm to find CTR_f.

The transmission range Tr is one of the parameters that determines the number of active links at any given time. Similarly, the location of the center of the fault region is one of the parameters that determines the number of ANPs that can potentially be damaged by the fault. For a specific region centered at an I-point f_i, and a transmission range Tr, we define an interval on the timeline as *static interval*, if the set of potentially damaged nodes due to a region fault at location f_i and the set of alive links with transmission range Tr remain unchanged. We can find static intervals using the timeline *region-coverage(f_i)* and the timeline *link-lifetime* $L(Tr)$. In order to find the static intervals for I-point f_i and transmission range Tr, we define four events during the time interval when f_i *exists*: (i) a dead link comes alive, (ii) a live link dies, (iii) an ANP node comes within coverage area of f_i and (iv) an ANP node moves out of the coverage area of f_i. The instance of time at which any of the

four events takes place is the instance of the start time of a new static interval. Let $SI(f_i, Tr)$ be a sorted list of events resulting from combining the sorted list $L(Tr)$ and $NT(f_i, t_m)$ for all $t_m \in T(f_i)$. Therefore, between any two consecutive elements in $SI(f_i, Tr)$ neither the topology nor the region coverage changes.

Once the nodes within a region (or nodes covered by a region) and the set of active links during a static interval are known, we can use Algorithm-2 in [11] in order to find the *region based connectivity* of the network with respect to I-point f_i. Region based connectivity with respect to an I-point f_i ($RBC(f_i)$) is defined to be the minimum number of nodes in the region centered at f_i whose failure disconnects the network. If the number of nodes that can be damaged due to a region based fault at f_i is n_i (i.e., the fault at f_i covers n_i nodes), we would like the ANPs to have enough transmission range, so that the region based connectivity of the graph is at least $n_i + 1$. This will ensure that the network will remain connected if any subset of the covered nodes fails. Using Algorithm-2 of [11], and applying binary search within the range $0 - Tr_{max}$ we can find the minimum transmission range necessary in each static interval, to ensure that the network remains connected when a region f_i fails (during the interval when it f_i is exists). We define err to be the maximum acceptable difference between the smallest transmission range necessary to maintain connectivity and the smallest transmission range computed by the algorithm to maintain connectivity. In our algorithm, we set the maximum possible transmission range to be equal to diameter of the deployment area. The algorithm computes the *minimum* transmission range necessary to maintain connectivity for each static interval. The *maximum* of these *minimum values* computed is the *critical transmission range in faulty scenario (CTR_f)* . Alg. 3 provides all the details.

In Alg. 3, line 1 takes $O(n^2)$. In order to compute $T(f_i)$ we need to solve $s_{ij} = 2R$. As described in Alg. 1, for the case that ANPs move at the same velocity,

ω, this equation can be solved easily in constant time and it has two solutions in one period. So, $|T(f_i)| \leq 2$. In line 5, we have to solve $D_{iv}(t) = R$ for all $v \in V$. For one node v, this equation also in one period can have a constant number of solutions since it can easily be converted to a single variable polynomial equation with degree 6. So, computation of $NT(f_i, t_m)$ takes $O(n)$ and $\sum_{t_m} |NT(f_i, t_m)| = O(n)$. Consequently, lines 2-7 have complexity of $O(n^3)$. The *while* loop is repeated for $\log Tr_{max}$ (binary search complexity). As it is discussed in Alg. 1, computation of $L(Tr)$ takes $O(n^2 \log n)$. Computation of $SI(f_i, Tr)$ need sorting the sorted lists $NT(f_i, t_m)$ and $L(Tr)$ which takes $O(n^2)$. Clearly, $|SI(f_i, Tr)| = O(n^2)$. Computing $RBC(f_i)$ takes $O(n^4)$ [11]. Therefore, the time complexity of Alg. 3 is $O(n^8 \log Tr_{max})$.

Algorithm 3 Computing CTR_f

1: Compute $s_{ij}(t)$ for all pair of ANPs i and j
2: **for all** I-points $f_i \in \mathcal{I}$
3: Compute $T(f_i) = \{(t_f^1, t_f^2), \ldots, (t_f^{k-1}, t_f^k)\}$
4: **for all** $t_m \in T(f_i)$
5: Compute $NT(f_i, t_m) = \{(t_{m_1}, N_{m_1}), \ldots, (t_{m_p}, N_{m_p})\}$
6: $error = Tr_{max}$, $tr_a = 0$, $tr_b = Tr_{min} = Tr_{max}$
7: **while** $error > err$
8: $error = error/2$, $Tr = (tr_a + tr_b)/2$
9: Find $L(Tr) = \{e_1, e_2, \ldots, e_t\}$ using Alg. 1
10: **for all** I-points $f_i \in \mathcal{I}$
11: $SI(f_i, Tr) \leftarrow$ Sort the lists $NT(f_i, t_m)$ and $L(Tr)$ based on time of the events (considering all $t_m \in T(f_i)$)
12: **for all** $event \in SI(f_i, Tr)$
13: Update the graph $G(t)$ (by adding or removing the links) or the region coverage of f_i
14: $RBC(f_i) \leftarrow$ Using Alg. 2 in [11] Compute the region-based connectivity considering only one region centered at I-point f_i
15: **if** $(RBC(f_i) \geq n_i + 1)$ NextSI $\leftarrow$ TRUE
16: **else** NextSI $\leftarrow$ FALSE; break;
17: **if** (NextSI = FALSE) $tr_a = Tr$; break;
18: **if** (NextSI = TRUE) $tr_b = Tr$; $Tr_{min} = Tr$
19: **return** Tr_{min}

2.6 Computation of Critical Transmission Range in Delay Tolerant Airborne Networks CTR_D

In previous sections we explained the computation of critical transmission range in fault free (CTR) and faulty scenarios (CTR_f). However, it may not be possible to equip the ANPs with radios that have coverage of radius CTR. Therefore, the backbone network cannot be connected all the times. On the other hand, based on the type of data that should be transmitted between ANPs, data transmissions may be tolerant to some amount of delay. Hence, ANPs may not be needed to have end-to-end paths all the times but they should be able to transmit data to each other in some limited time through intermediate nodes in different network topologies. In this section we investigate the problem of computation of minimum transmission range in delay tolerant airborne networks.

We consider that the trajectories and the distance function $s_{ij}(t)$ of the nodes are periodic over time. As a consequence, the network topologies are repeated periodically. However, periodicity is not an underlying assumption and our results can be utilized in non-periodic scenario as long as the node trajectories for the whole operational duration of a network are given. In Section 2.3 we explained how we can compute *link lifetime timeline* and accordingly the network topologies caused by ANPs mobility in a time period when all five controlling parameters are given. We represent the set of topologies in a periodic cycle starting from time t_0 (starting time of network operation) by the set $\mathcal{G} = \{G_1, G_2, \ldots, G_l\}$. Each network topology G_i exists for a time duration of T_i. As the focus of this section is study of the delay caused by network disconnection (which may be viewed as delay due to queuing at an intermediate node), we assume that other delays due to transmission and propagation are negligible.

In Fig. 2.8, an example of a dynamic graph with two topologies G_1 and G_2 in one periodic cycle is shown. G_1 and G_2 last for T_1 and T_2 time units respectively. It can be observed that there is no end-to-end path from A to C in either G_1 or G_2. However, A can transmit data to B in G_1, and B can forward it to C in G_2. In this case we say that A can reach C through a temporal path with delay equal to the lifetime of G_1, i.e. T_1; and the temporal path is completed in G_2. We define a *temporal path* from node s to d to be a set of tuples $\{(t_1, (v_1, v_2)), (t_2, (v_2, v_3)), \ldots, (t_k, (v_k, v_{k+1}))\}$ such that $v_1 = s, v_{k+1} = d$, $v_i \in V$ and for every tuple $(t_i, (v_i, v_{i+1}))$, edge (v_i, v_{i+1}) is active at time t_i, and $t_i \geq t_{i-1}$ for all $1 \leq i \leq k$. Moreover, without loss of generality, we assume that t_i corresponds to the starting time of a topology in $\mathcal{G}$. Then the path delay is defined to be $t_k - t_0$ where t_0 is the starting time of G_1 in the first periodic cycle. We note that all path delays are computed with respect to starting point t_0 but we later show that we can modify the starting point to any time.

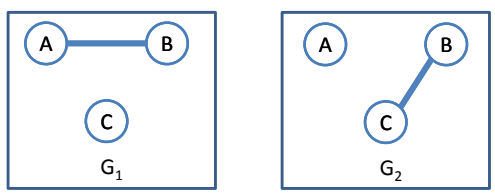


Figure 2.8: A dynamic graph with two topologies G_1 and G_2

We note that existence of a path from node i to j with some delay does not guarantee the existence of a path from j to i with the same delay. For example, in Fig. 2.8 the path from C to A has a delay of $T_1 + T_2$ while the path delay from A to C is equal to T_1. We say that a dynamic graph $G(t)$ is *connected with delay* D if there exists a temporal path from every node $i \in V$ to every node $j \in V - \{i\}$ with delay smaller than D. In a network, if the transmission range Tr is too small, ANPs may not be able to reach each other at all; i.e. there is no temporal path of finite delay between the ANPs. We define *critical transmission range in delay tolerant network (*CTR_D*)* to be the minimum transmission range necessary to ensure that the dynamic

graph is *connected with delay* D. We define *the connectivity problem in delay tolerant networks* as the problem of computation of CTR_D given the first four controlling parameters defined in Section 2.2, and the delay threshold D.

In order to find the value of CTR_D, first we explain an algorithm to check whether a transmission range Tr is adequate for having a connected dynamic network with delay D. Using Algorithm 1 in Section 2.3 we can compute the different network topologies and their lifetime in one periodic cycle. Before describing the rest of the algorithm, first we propose an observation.

Observation 1. *For a given transmission range Tr, there is a temporal path from every node u to every node v with finite delay iff the superimposed graph $G_c = \{V, \bigcup_{i=1}^{l} E_i\}$, where E_i is the set of edges in G_i, is connected.*

Although a transmission range Tr may be enough to result in a connected superimposed graph G_c, it may not be sufficient for the existence of a temporal path between every pair of nodes with delay smaller than a threshold D even if D is as large as $\sum_{i=1}^{l-1} T_i$. Fig. 2.9 depicts an AN with three topologies in one period. It can be observed that A cannot have a temporal path from A to D in the first period. Actually the fastest path includes edges (A, B) in G_3 in first period, (B, C) in G_2 in the second period and (C, D) in G_1 in the third period. Therefore, the path delay is $2(T_1 + T_2 + T_3)$. Generally, in the worst case in every period just a subpath (a set of consecutive edges) in one topology is used and therefore the maximum delay of a temporal path will be $D_{max} = (l-1)\sum_{i=1}^{l} T_i$. Hence, if $D \geq D_{max}$, examining the connectivity of G_c is enough to decide whether for a transmission range there exists a temporal path of delay smaller than D between every pair of nodes in the dynamic network.

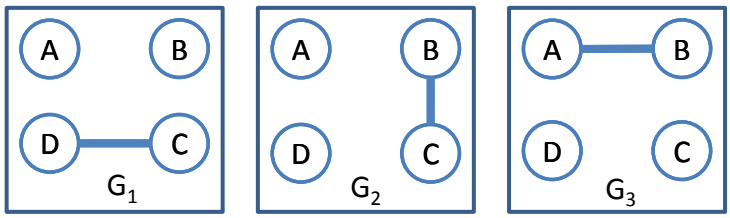


Figure 2.9: A dynamic graph with three topologies G_1, G_2 and G_3

Next, we explain the algorithm that checks for a given value of transmission range Tr whether a network is connected with delay D where $D < D_{max}$. Let $N(u)$ denotes the set of nodes that are reachable from $u \in V$ with delay smaller than D. Initially $N(u) = \{u\}$. The algorithm starts by computing the connected components in every topology G_i. Let $C_i = \{C_{i,1}, C_{i,2}, \ldots C_{i,q_i}\}$ represents the set of connected components in G_i where $C_{i,j}$ is the set of nodes in jth component of G_i and $q_i = |C_i|$. Let g and h be the quotient and remainder of $\frac{D}{\sum_{i=1}^{l} T_i}$ respectively, and $t_0 + h$ is the time where the network topology is G_p for a p, $1 \leq p \leq l$. Therefore, the topologies in time duration t_0 to $t_0 + D$ includes G_1 to G_l for g number of cycles and G_1 to G_p in last periodic cycle. Starting from first topology G_1 in first period, in each topology G_i, if a node v is in the same connected component with a node $w \in N(u)$, then v can be reachable from u through a temporal path which is completed in G_i; hence, $N(u)$ is updated to $N(u) \cup (\bigcup_{k:N(u)\cap C_{ik}\neq\emptyset} C_{ik})$. In this step the algorithm goes through all the topologies from t_0 to $t_0 + D$. In the end, if $N(u) = V$ for all $u \in V$ then the transmission range Tr is sufficient for having a connected network with delay D. In Algorithm 4 the steps of checking the connectivity of a dynamic graph with delay D is proposed.

Algorithm 4 Checking Connectivity of Airborne Network with delay D

Input: $\mathcal{G}(t) = \{G_1, G_2, \ldots, G_l\}$ and delay threshold D

Output: *true* if dynamic graph $\mathcal{G}(t)$ is connected with delay D; otherwise *false*.

1: Initialize $N(u) = \{u\}$ for every $u \in V$

2: **for all** topologies $G_i, 1 \leq i \leq l$

3: Compute $C_i = \{C_{i,1}, C_{i,2}, \ldots C_{i,q_i}\}$, the set of connected components of G_i

4: **for all** periods 1 to g

5: **for all** topologies $G_i, 1 \leq i \leq l$

6: **for all** node $u \in V$

7: $N(u) \leftarrow N(u) \cup (\bigcup_{k:N(u)\cap C_{i,k}\neq\emptyset} C_{i,k})$

8: **for all** topologies $G_i, 1 \leq i \leq p$ (the topologies in the last period)

9: **for all** node $u \in V$

10: $N(u) \leftarrow N(u) \cup (\bigcup_{k:N(u)\cap C_{i,k}\neq\emptyset} C_{i,k})$

11: **for all** node $u \in V$

12: **if** $N(u) \neq V$, **r**eturn $false$

13: **r**eturn $true$

As we explained in section 2.3, number of topologies, l in one period is $O(n^2)$. The computation of the connected components of a graph $G_i = (V, E_i)$ needs using either breadth-first search or depth-first search with time complexity of $O(|V| + |E_i|) = O(n^2)$. Hence, step 2-4 takes $O(n^4)$. This algorithm is used for the case that $D < (l-1)\sum_{i=1}^{l} T_i$. Therefore, number of periods $g < l - 1$ and $g = O(n^2)$. Computation of Step 7 also needs $O(n^2)$ since $|N(u)|$ and total size of all components in G_i is $O(n)$. Finally, we can conclude that total time complexity of the algorithm is $O(n^7)$.

As we mentioned before, in Algorithm 4 the delays are computed with respect to t_0. We can easily extend it to any time in the network operation duration, by

repeating Algorithm 4 for every $t_i, 1 \leq i \leq l$ where t_i is the starting time of topology G_i. The complexity increases by a factor of $l = O(n^2)$. We note that in this case even if a node starts communication at some time instances t where $t_i \leq t \leq t_{i+1}$, the delay will be smaller than the case it starts at t_i. Hence it is enough to just consider the time points in which a topology change happens.

Similar to the computation of CTR and CTR_f, in order to compute CTR_D we can conduct a binary search within the range $0 - Tr_{max}$ and we can determine the smallest transmission range that will ensure the AN is connected with delay D during the entire operational time. The binary search adds a factor of $\log Tr_{max}$ to the complexity of Algorithm 4.

2.7 Simulations

The goal of our simulation is to compare critical transmission range in different scenarios of non faulty, faulty and delay tolerant and investigate the impact of various parameters, such as the number of ANPs, the region radius and delay on critical transmission range. In our simulation environment, the deployment area is a 1000×1000 square mile area. The centers of the orbits of the ANPs are chosen randomly in such a way that the orbits do not intersect with each other. In our simulation, we assume that all the ANPs move at the same angular speed of $\omega = 20$ radian/hour. Hence a period length is 0.1π hour. One interesting point to note is that, in this environment where all the ANPs are moving at the same angular speed on circular paths, the value of CTR is independent of the speed of movement of the ANPs. This is true because changing the angular speed ω effects just the time at which the events, such as a link becomes active or a link dies, take place. If we view the dynamic topology of the backbone network over one time period as a collection of topologies $\mathcal{G} = \{G_1, G_2, \ldots, G_l\}$, where G_i morphs into $G_{i+1}, 1 \leq i \leq l$ at some time, by increasing or decreasing the angular speed of all the ANPs, we just make the transitions

from G_i to G_{i+1} faster or slower, without changing the topology set $\mathcal{G}$. Similarly, the set of ANPs that are damaged to failure of a region at a certain time, remains unchanged.

In our first set of experiments we compute CTR, CTR_f when $R = 20, 60$, and CTR_D when $D = 0.5period, 2period$ for different values of number of nodes, n. Fig. 2.10 depicts the result of these experiments. In these experiments, for each value of n we conducted 30 experiments and the results are averaged over the 30 different random initial setups. We set *orbit radius* = 10. We observe that expectedly an increase in the number of nodes results in a decrease in CTR, CTR_f and CTR_D. Moreover, $CTR_D \leq CTR \leq CTR_f$ for all instances. In all of the experiments, we compute CTR_D with respect to all times (corresponding to beginning of a new topology) not only t_0.

In the second set of experiments, we examined the impact of change of the region radius R on the transmission range. We conducted these experiments for two values of *orbit radii*, 10 and 30, and $n = 35$ in both the cases. For each value of R, we conducted 100 experiments and the results are averaged over them. Fig. 2.11(a) shows the results. It may be observed that increase in the value of R leads to increase in CTR. This observation is quite expected as larger regions can destroy more nodes at a time. Moreover, it may be noted that for larger values of orbit radii the transmission range also increases. The reason is that for a specific number of nodes in a bounded deployment area, larger orbit radii result in larger distance between the nodes. Accordingly, larger transmission range is necessary, particularly in the case of larger Rs.

Finally, we conducted experiments to investigate the impact of delay D on the value of CTR_D. Our experiment setup is the same as our first set of experiments. Fig. 2.11(b) depicts the results. We observe that when value of delay D is zero the

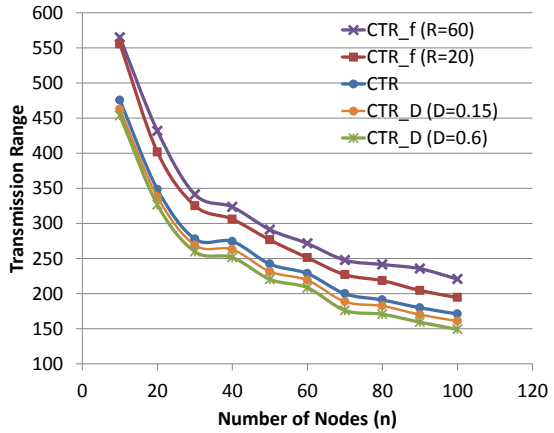
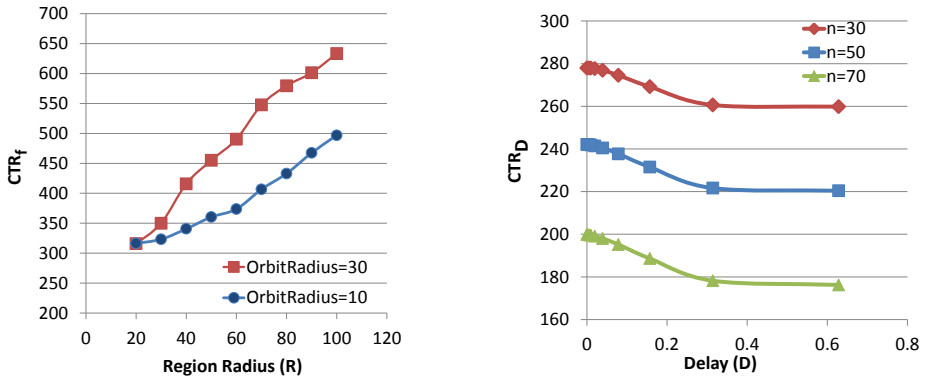


Figure 2.10: Transmission Range vs. Number of Nodes



(a) (b)

Figure 2.11: (a) Transmission Range (CTR_f) vs. Region Radius, $n = 35$; (b) Transmission Range (CTR_D) vs. Delay

value of CTR_D is equal to CTR and by increasing delay, CTR_D decreases and the interesting observation is that when delay becomes greater than $2period$ the decrease in the value of CTR_D is unnoticeable or even zero.

2.8 Conclusion

Existence of sufficient control over the movement pattern of the mobile platforms in Airborne Networks opens the avenue for designing topologically stable airborne networks. In this chapter, we discussed the system model and architecture for Airborne Networks (AN). We studied the problem of maintaining the connectivity in the un-

derlying dynamic graphs of airborne networks when trajectories of nodes are given. We developed techniques to compute the *dynamic topology* of the AN at any instance of time and proposed an algorithm to compute critical transmission range when all nodes are operational. Motivated by the importance of robustness and fault tolerance capability of ANs, we have also investigated the region-based connectivity of the ANs and proposed an algorithm to find the minimum transmission range necessary to ensure that the surviving nodes of the network remain connected, even when all or some nodes of region fail due to an enemy attack. In the process of computing the minimum transmission range in faulty scenario, we developed techniques to (i) compute all the fault regions that need to be considered to *ensure overall connectivity at all times* and (ii) compute the set of nodes that might be damaged by the *failure of a specific region at a specific time*. Moreover, we defined and formulated the critical transmission range in delay tolerant airborne networks CTR_D and proposed an algorithm to compute CTR_D. Through simulations, we have illustrated the impact of the number of nodes, the region radius in faulty scenario and delay in delay tolerant networks on the minimum transmission range.

Chapter 3

FAULT TOLERANT DESIGN OF WIRELESS SENSOR NETWORKS

3.1 Introduction

The primary goal of a wireless sensor network is to deliver the collected sensor data to the sink node, either in the raw form or after in-network aggregation. Generally, each sensor comprises of two components: the sensing component and the communication component. The sensing component gathers information from the surrounding area and the communication component is responsible for communicating the gathered information to the appropriate node for processing. Most often, the data collection operation from all the sensor nodes is carried out by creating a *tree* topology that *spans* all the sensor nodes, with the sink node as the *root* [27, 28].

Directional antennas offer substantial advantages over their omni-directional counterparts, as they can focus their transmission energy in a specific direction, using a narrow beam of width α. A directional antenna can be mounted on a *swivel* and can be oriented towards a target or alternately each sensor can be equipped with multiple antennas, each occupying a sector with beam width α. The transmitted signal disperses in any unguided wireless media and as a consequence, the signal strength diminishes with distance. Although attenuation is in general a complex function of the distance and the makeup of the environment through which the signal propagates, a significant cause of signal degradation is *free space loss*. Free space loss for an ideal isotropic antenna is measured as the ratio of the transmitted power to the received power and is given by $\frac{(4\pi d^2)}{\lambda^2}$. where λ is the carrier wavelength, and d is the propagation distance between transmission and reception antennas. In particular, the energy required by an antenna to reach all nodes within its transmission radius is proportional to the area covered. Thus, an omni-directional antenna with a transmission radius r will consume power proportional to πr^2 (the area of a circle with

radius r) while a directional antenna with beam width α radians will consume power proportional to $\frac{\alpha}{2}r^2$. The expression is valid under the assumption that the signal is transmitted over the primary lobe and the power consumed by the remaining lobes is negligible [29]. The expressions show that with a directional antenna with beam width α, power consumption can be reduced by a factor of $\frac{\alpha}{2\pi}$. Moreover, in comparison with omnidirectional antennas, the directional antennas have significantly less interference and fading [30, 31]. For additional information on antenna theory, we refer the reader to [32].

Due to such attractive features, sensors with directional antennas are being increasingly used for wireless sensor networks. Some examples include camera networks for vision-based sensing, radar networks for weather monitoring and sonar network for underwater object detection [31]. With rapid advances in the miniaturization of directional antenna technology, it is likely that the use of directional antennas in sensor platforms will proliferate. This trend is demonstrated by the increasing interest in the use of directional antennas for performance improvement in wireless networks in general and wireless sensor networks in particular [30].

Just as the directional antennas offer a number of advantages, it also introduces a few problems. When sensor nodes use omni-directional antennas, the network topology typically is a *mesh* and not a *tree*. A tree that spans over this mesh topology is utilized for the purpose of data gathering. Although it may appear that the use of omni-directional antenna for the purpose of data gathering is wasteful, as many of the network links created by such antennas are never utilized, this is not completely true. The unused links essentially introduce a certain level of *redundancy* that can be utilized when one or more of the sensor nodes fail and the data gathering spanning tree becomes disconnected. However, the negative aspect of this lack of redundancy is that it can no longer deal with a fault scenario, where one or more sensor nodes

fail. Without any built-in redundancy, when some nodes fail, the data gathering tree is disconnected and the sink node fails to receive any data from some of the sensors.

The primary motivation of our work is to retain the advantage of both types of antennas by combining the *efficiency* of directional antennas with the *redundancy* of omni-directional ones. In this chapter we study the problem of enhancing the fault tolerance capability of a data gathering tree by adding a few additional links so that the failure of any one sensor or a pair of adjacent sensors would not disconnect the tree. We consider that the sensor nodes are equipped with directional antennas and nodes p and q can communicate with each other if antennas of p and q direct their beams to each other. In this case a bidirectional link is used between the nodes p and q. Fig. 3.1 shows a data collection tree with two sensor nodes u and v and a sink r. The sectors show the communication ranges and the lines show the wireless links. The addition of an edge between two nodes p and q in the sensor network topology corresponds to the deployment of two new directional antennas at the nodes p and q directed towards each other.

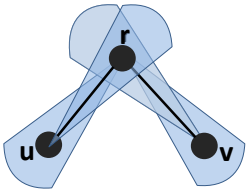


Figure 3.1: A data collection tree constructed by directional antennas

Most of the previous studies on the fault tolerant design of wireless sensor networks [33, 34] consider that the sensor nodes are equipped with omni-directional antennas and the objective is to assign transmission power to the nodes such that the network is k-connected while power consumption is minimized. In a k-connected (k-fault-tolerant) network there are k disjoint paths between every two nodes. However, for data collection in sensor networks it is necessary and sufficient that every sensor node has k disjoint paths to the sink node and k disjoint paths between every two

nodes may not be needed. In [35] the authors studied the problems of all-to-one and one-to-all k-fault-tolerant topology control problems. In these problems also the sensor nodes are considered to have omni-directional antennas and the network graph is directed. In [11], authors introduced a new fault tolerance metric called region-based connectivity. In [11], it is assumed that the nodes that may fail are confined to a region. A region may be defined in several ways based either on the geometric layout of the sensor nodes in the deployment area or on the network topology. In this research, we are confining our attention to the fault models based on the network topology of the sensor network and will study two failure models i) single node failure and ii) the simultaneous failure of two adjacent nodes. Even for these two specific models the problems turn out to be computationally hard. Assuming that addition of each link to the tree involves a cost, we study the problem of *least cost augmentation of the tree* so that even after failure of a single node or a pair of adjacent nodes, all the surviving nodes will remain connected to the sink node. We prove that the least cost tree augmentation problem is NP-complete under both types of fault scenarios. Moreover, we provide two approximation algorithms, one for single node failure and the other for a pair of adjacent node failure, with performance bounds of two and four respectively. The experimental evaluations of the algorithms demonstrate that they perform even better in practice and almost always produce near optimal solution.

In the theoretical computer science community, problems of this vein are known as the *graph augmentation problems*. Two important problems in this class are the bi-connectivity augmentation ($BICA$) and the bridge-connectivity augmentation ($BRCA$) (a bridge is defined to be an edge whose removal disconnects the graph) [36]. Although at a first glance, it may appear that *tree connectivity augmentation* under single node fault model, TCA_1, is the same as $BICA$ or $BRCA$, we demonstrate through the examples shown in Fig. 3.2(a) and 3.2(b) that TCA_1 is distinctly

different from both $BICA$ and $BRCA$. The solid lines in Fig. 3.2(a) and 3.2(b) are the existing edges in the input graph. A few of the edges that may be added to the graph are shown in dashed lines and cost of each of these edges is 1. The cost of the edges that can be added to the graph, but not shown in dashed lines, is 10. In Fig. 3.2(a), the solution of the TCA_1 is the addition of edges $\{(a, b), (c, d)\}$ with a total cost of 2. However, for $BICA$, one more edge such as (b, c) is also needed and as such the total cost will be at least 12. The Fig. 3.2(b) shows an example in which TCA_1 has a solution with cost 3, (addition of edges $\{(g, h), (h, i), (i, j)\}$) but $BRCA$ requires the addition of edges $\{(a, e), (g, h), (i, j), (d, f)\}$ with a total cost of 4.

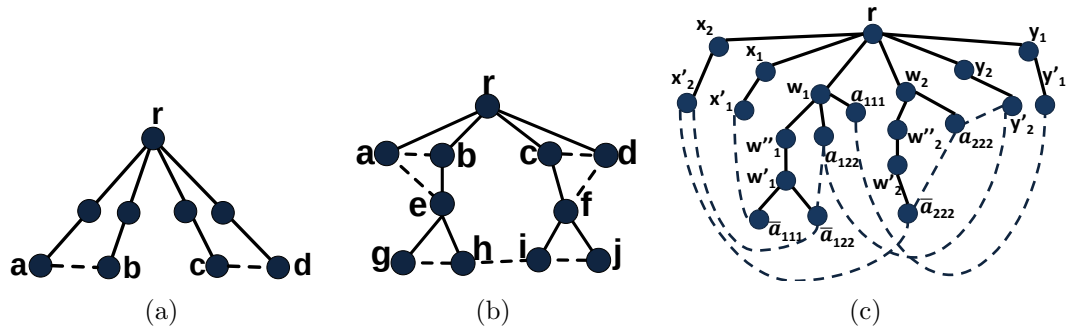


Figure 3.2: (a) Comparison of TCA_1 with BICA, (b) Comparison of TCA_1 with BRCA, (c) An instance of TCA_1 correponding to a 3DM instance

In [37], the authors have studied the problem of least cost augmentation of a graph such that there exist at least $k + 1$ disjoint paths from a root node r to each node $v \neq r$. They propose an approximation algorithm with approximation ratio 2 where its time complexity is $O(n^4)$. Even though the TCA_1 problem is a special case of the problem studied in [37], we prove that TCA_1 is still NP-complete. We propose an approximation algorithm for TCA_1 with the same approximation ratio of 2 but $O(n^2)$ time complexity. To the best of our knowledge the graph augmentation problems under (topological) region based failures has not been studied before. Hence, toward this direction we study the problem of tree augmentation problem under two adjacent node failure.

The rest of the chapter is organized as follows. In Section 3.2 we formally define the problems we have studied. In Section 3.3, we show that both problems are NP-complete. In Section 3.4 and 3.5 we outline approximation algorithms for the single-node and adjacent node failure problems, respectively. In Section 3.6 we report the experimental results and conclude in Section 3.7.

3.2 Problem Formulation

Definition: $u \odot 1$ fault tolerant graph - A graph $G = (V, E)$ with a specified vertex $u \in V$ is said to be $u \odot 1$ *fault tolerant* if after the failure of any one node $v \in V - \{u\}$, any residual node $w \in V - \{v\}$ remains connected to the node u.

Definition: $u \odot 2$ fault tolerant graph - A graph $G = (V, E)$ with a specified vertex $u \in V$ is said to be $u \odot 2$ *fault tolerant* if it has both the following properties:

(1) G is $u \odot 1$ fault tolerant.

(2) After the failure of any pair of *adjacent nodes* $v, w \in V - \{u\}$, any residual node $x \in V - \{v, w\}$ remains connected to the node u.

Tree Connectivity Augmentation Problem - Single Fault (TCA_1)

Instance: Complete undirected graph $G = (V, E)$, weight function $c(e) \in \mathbb{Z}^+$, $\forall e \in E$, a spanning tree $T_1 = (V, E_1)$ of G rooted at some node $r \in V$, and a cost budget B.
Question: Is there a set $E^{aug} \subseteq E - E_1$, such that the graph $(V, E_1 \cup E^{aug})$ is $r \odot 1$ fault tolerant and $\sum_{e \in E^{aug}} c(e) \leq B$?

Tree Connectivity Augmentation Problem - Adjacent Double Fault (TCA_2)

Instance: The same as the instance in TCA_1 problem.
Question: Is there a set $E^{aug} \subseteq E - E_1$, such that the graph $(V, E_1 \cup E^{aug})$ is $r \odot 2$ fault tolerant and $\sum_{e \in E^{aug}} c(e) \leq B$?

3.3 Computational Complexity

In this section we show that both the tree connectivity augmentation problems, TCA_1 and TCA_2 are NP-complete, by a transformation from the *3-dimensional matching*, which is known to be NP-complete [38]. Obviously, a candidate solution of TCA_1 and TCA_2 can be verified in polynomial time, and they are in NP. We also show hardness of approximation result for TCA_1 problem using similar transformation.

3-Dimensional Matching (3DM)

Instance: A set $M \subseteq W \times X \times Y$, where W, X and Y are disjoint sets having the same number q of elements.

Question: Does M contain a matching, that is, a subset $M' \subseteq M$ such that $|M'| = q$ and no two elements of M' agree in any coordinate?

Theorem 3.3.1. *TCA_1 is NP-complete.*

Proof. Let $M \subseteq W \times X \times Y$ be an instance of 3DM, with $|M| = p$ and $W = \{w_i | i = 1, 2, \ldots, q\}$, $X = \{x_i | i = 1, 2, \ldots, q\}$ and $Y = \{y_i | i = 1, 2, \ldots, q\}$. We start by creating a set of nodes having labels as follows:

- r, where r will be the root of the spanning tree T_1,
- w_i (and similarly x_i, y_i) for all $w_i \in W$ (respectively $x_i \in X$, and $y_i \in Y$),
- for each $w_i \in W$ (and $x_i \in X, y_i \in X$), one additional node with label w_i' (respectively x_i' and y_i'),
- for each $w_i \in W$, one additional node with label w_i''
- for each triple $(w_i, x_j, y_k) \in M$, two nodes with labels $a_{ijk}, \bar{a}_{ijk}$.

We now create an instance of TCA_1 as follows:

$$
\begin{aligned}
V &= \{r\} \cup \{w_i, w_i', w_i'', x_i, x_i', y_i, y_i' | i = 1, 2, \ldots, q\} \cup \\
&\quad \{a_{ijk}, \bar{a}_{ijk} | (w_i, x_j, y_k) \in M\} \\
E &= \{(u, v) | u, v \in V \ and \ u \neq v\} \\
E_1 &= \{(r, w_i), (w_i, w_i''), (w_i'', w_i') | i = 1, 2, \ldots, q\} \cup \\
&\quad \{(r, x_i), (x_i, x_i'), | i = 1, 2, \ldots, q\} \cup \\
&\quad \{(r, y_i), (y_i, y_i') | i = 1, 2, \ldots, q\} \cup \\
&\quad \{(w_i, a_{ijk}) | (w_i, x_j, y_k) \in M\} \ \cup \\
&\quad \{(w_i', \bar{a}_{ijk}) | (w_i, x_j, y_k) \in M\} \\
B &= p + q
\end{aligned}
$$

$c(x_j', \bar{a}_{ijk}) = c(y_k', a_{ijk}) = c(\bar{a}_{ijk}, a_{ijk}) = 1$, for all $(w_i, x_j, y_k) \in M$.

All other edges in E have weight 2.

Fig. 3.2(c) depicts an instance of TCA_1 corresponding to the 3DM instance where $q = 2$ and $M = \{(w_1, x_1, y_1), (w_1, x_2, y_2), (w_2, x_2, y_2)\}$. The solid lines show edges in E_1 and the dashed lines show the edges in $E - E_1$ with cost 1.

We claim that M contains a matching M' iff there is a set E^{aug} of cost no more than B, such that the graph $(V, E_1 \cup E^{aug})$ is $r \odot 1$ fault tolerant.

To prove the only if part, let M contain a matching M'. We form E^{aug} by following the procedure given below:

Step i) For each triple $(w_i, x_j, y_k) \in M'$, we add edges $(x_j', \bar{a}_{ijk})$ and (y_k', a_{ijk}),

Step ii) For each triple $(w_i, x_j, y_k) \in M - M'$, we add edge $(a_{ijk}, \bar{a}_{ijk})$.

Since $|M'| = q$ and $|M - M'| = p - q$, and the cost of edges added in Step i and Step ii is 2 and 1 respectively, the total cost of the added edges in Step i and

Step ii is $2q + p - q$. Thus, the total cost of the edges in E^{aug} is $p + q$. $E_1 \cup E^{aug}$ includes the following cycles that pass through r:

- $r, w_i, w_i'', w_i', \bar{a}_{ijk}, x_j', x_j, \forall i, j, k : (w_i, x_j, y_k) \in M'$,
- $r, w_i, a_{ijk}, y_k', y_k, \forall i, j, k : (w_i, x_j, y_k) \in M'$,
- Considering that each w_i is in a triple $(w_i, x_{j'}, y_{k'}) \in M'$ and is in previous cycles, there is a cycle $r, w_i, a_{ijk}, \bar{a}_{ijk}, w_i', \bar{a}_{ij'k'}, x_{j'}', x_{j'}, \forall i, j, k : (w_i, x_j, y_k) \in M - M'$.

It can be readily verified that all the nodes in $V - \{r\}$ appear in at least one of the above cycles. Therefore, there are two disjoint paths from r to each vertex.

To prove the if part, let there be a set of edges $E^{aug} \subseteq (E - E_1)$, with cost at most $p + q$, so that in the graph $(V, E_1 \cup E^{aug})$ every non-adjacent vertex of root r has two node disjoint paths to r. There are exactly $2p + 2q$ leaf nodes in $T_1 = (V, E_1)$ and they are not adjacent to r. Among these leaf nodes, there exists p nodes having labels of the form a_{ijk} and $\bar{a}_{ijk}$ and q nodes having labels of the form y_k' and x_j'. To ensure that $2p + 2q$ leaf nodes have two node disjoint paths to r, at least $p + q$ edges must be added to $T_1 = (V, E_1)$. Since the cost of the edges in E^{aug} is at most $p + q$ and $|E^{aug}| \geq p + q$, it implies that $|E^{aug}| = p + q$ and the cost of each edge in E^{aug} is 1. It may be noted that there are only three types of edges i) $(x_j', \bar{a}_{ijk})$, ii) (y_k', a_{ijk}) and iii) $(\bar{a}_{ijk}, a_{ijk})$ that have cost 1. In order to have two node disjoint paths from $2p + 2q$ leaf nodes to r, each node of the from y_k' and x_j' must be connected to a node of the form a_{ijk} and $\bar{a}_{ijk}$ respectively. The total cost of these set of edges will be $2q$. Since the total cost of E^{aug} is $p + q$, the cost of the edges to connect the remaining leaves of the type a_{ijk} or $\bar{a}_{ijk}$, (i.e., the ones that were not connected to either y_k' and x_j'), must be $p - q$ and the number of such leaves must be $2(p - q)$. This will only be possible, if $2(p - q)$ leaves can be grouped into $p - q$ pairs of nodes

$(a_{ijk}, \bar{a}_{i'j'k'})$, such that $i = i', j = j', k = k'$. This implies that exactly q nodes, each of the form a_{ijk} and $\bar{a}_{ijk}$, must be connected to q nodes, each of the form y'_k and x'_j, respectively, and these $2q$ nodes (a_{ijk} and $\bar{a}_{ijk}$ together) can be grouped into q pairs of nodes $(a_{ijk}, \bar{a}_{i'j'k'})$, such that $i = i', j = j', k = k'$. Moreover, since there are two disjoint paths from every node to r, nodes w_i, w'_i and w''_i are in a cycle with just one x'_j, x_j and r. Otherwise, there would be a $w_{i'}$ which is not in any cycle with r and its failure disconnects the graph. Since these q pairs of ijk indices connects to all the y'_k and x'_j nodes and w_i can be in exactly one cycle with x'_j, the corresponding subset $M' \subseteq M$ must be a matching for the instance of the 3DM problem.

Theorem 3.3.2. *TCA_2 is NP-complete.*

Proof. The proof is identical to that of Theorem 3.3.1 and hence omitted.

Theorem 3.3.3. *For some fixed $\epsilon > 0$, it is NP-hard to approximate TCA_1 within a factor of $1 + \epsilon$.*

Proof. We use similar reduction as in the NP-completeness proof of Theorem 3.3.1 from problem 3DM-5 to TCA_1. 3DM-5 is a bounded version of the 3DM problem in which every element of $W \cup X \cup Y$ can appear at most five times in a triple of M. It is shown in [39] that 3DM-5 is Max SNP-hard. In particular, it is proved that for some fixed $\epsilon_0 > 0$, it is NP-hard to distinguish whether an instance of 3DM-5 with $|W| = |X| = |Y| = q$ has a perfect matching (of size q) or every matching has size at most $(1 - \epsilon_0)q$.

We use exactly the same reduction as in the proof of Theorem 3.3.1. More explicitly, we create an instance of TCA_1 from an instance of 3DM-5, exactly in the same way we create an instance of TCA_1 from 3DM in Theorem 3.3.1. In the rest

of the proof, we show that the reduction from 3DM-5 to TCA_1 is gap-preserving. Specifically, we have to show that

1. If an instance, I, of 3DM-5 has a solution of size q then the corresponding instance of TCA_1, J has a solution of size $p+q$.

2. If the solution of 3DM-5 for the instance I is smaller than $q(1-\epsilon_0)$, then the solution (cost) of TCA_1 for the instance J is greater than $(p+q)(1+\epsilon)$.

The proof of part (1) is the same as proof of *only if* part in Theorem 3.3.1.

In order to prove the second part, we will prove the following lemma first:

Lemma 1: If the solution cost of TCA_1 for the instance J, $TCA_1(J)$, is at most $(p+q)(1+\epsilon)$, the solution of 3DM-5 for the instance I, $3DM(I)$, is at least $q-(2+10\Delta)(p+q)\epsilon$ where Δ is the maximum number of times that an element in $W \cup X \cup Y$ can appear in the triples of M.

Proof: Let $E^{aug} \subseteq E - E_1$ be a set of augmenting edges of cost at most $(p+q)(1+\epsilon)$ such that $G' = (V, E_1 \cup E^{aug})$ is $r \odot 1$ fault tolerant. The tree $T_1 = (V, E_1)$ has $2(p+q)$ leaves and they are not adjacent to r. Each leaf of T_1 must be adjacent to at least one edge of E^{aug} to make G' $r \odot 1$ fault tolerant. We call a leaf *proper* if it is adjacent to exactly one edge of E^{aug} and that edge has cost 1. We call a leaf *improper* otherwise (i.e., it is incident upon at least one edge of E^{aug} of cost 2 or upon more than one edge of E^{aug}).

We first prove that at most $2(p+q)\epsilon$ leaves are improper in the following way: For every improper leaf, the total cost of the edges of E^{aug} that are incident at this leaf is at least 2. Similarly, for every proper leaf this cost is exactly 1. The sum of these costs over all leaves is at most $2 \cdot cost(E^{aug})$, since the cost of every edge of E^{aug} is counted at most twice (once from every end). Thus,

number of proper leaves + 2· (number of improper leaves) $\leq 2 \cdot cost(E^{aug}) \leq 2(p+q)(1+\epsilon)$.

Since total number of leaves is $2(p+q)$, we will have $2(p+q)$+(number of improper leaves) $\leq 2 \cdot cost(E^{aug}) \leq 2(p+q)(1+\epsilon)$. Hence, the number of improper leaves is at most $2(p+q)\epsilon$.

We now construct a set M' which is almost a matching. Initially, let $M' = \emptyset$. Then iteratively for $j = 1, 2, ..., q$ we try to find a triple (in M) that contains x_j and add it to M', as follows. If x'_j is proper, then it is adjacent to a cost 1 edge of E^{aug}; hence it is adjacent to some leaf $\bar{a}_{ijk}$. If both $\bar{a}_{ijk}$ and a_{ijk} are proper, then the latter is adjacent (via a cost 1 edge) to some leaf y'_k. If this leaf y'_k is proper, then add the triple (w_i, x_j, y_k) to M'. Notice that $M' \subseteq M$.

We next claim that $|M'| \geq q - 2\Delta(p+q)\epsilon$. Indeed, an improper x'_j , a_{ijk}, or $\bar{a}_{ijk}$ can cause only one iteration (namely, the one with the corresponding value of j) to fail. An improper y'_k can cause at most Δ iterations to fail, since it can be connected by edges of cost 1 to at most Δ leaves a_{ijk}. Denoting the number of improper y'_k by n_y, we have that the number of iterations that fail is at most $2(p+q)\epsilon - n_y + n_y\Delta$. Since we showed before that $n_y \leq 2(p+q)\epsilon$, this is at most $2\Delta(p+q)\epsilon$.

By our construction, the triples in M' definitely have distinct elements from X and from Y, but its elements from W might be repeated. For every element w_i that belongs to more than one triple in M', we remove from M' all but one of the triples that contain w_i. The resulting set of triples, denoted M'', is thus a matching. Let $\mu = q - |M''|$ be the number of vertices w_i that do not appear in any triple of M' (or equivalently, of M''). Notice that $|M'| - |M''| \leq q - |M''| = \mu$, so an upper bound on μ yields a lower bound on the size of the matching M''. We will show that $\mu \leq (2 + 8\Delta)(p+q)\epsilon$. Let $E^{aug'}$ be the edges of E^{aug} that correspond to triples in

M', namely, those edges $(x'_j, \bar{a}_{ijk})$ and (y'_k, a_{ijk}) for $(w_i, x_j, y_k) \in M'$. We have that $cost(E^{aug'}) \geq 2|M'| \geq 2q - 4\Delta(p+q)\epsilon$; hence

$$cost(E^{aug} - E^{aug'}) \leq (p+q)(1+\epsilon) - 2q + 4\Delta(p+q)\epsilon = p - q + (1+4\Delta)(p+q)(1+\epsilon) \qquad (1)$$

We showed that each leaf (of T_1) a_{ijk} or $\bar{a}_{ijk}$ must be incident to an edge of E^{aug}. The edges of $E^{aug'}$ are incident, by their definition, to at most $2|M'| \leq 2q$ distinct such leaves; thus, the edges of $E^{aug} - E^{aug'}$ must be incident to the (at least) $2p - 2q$ remaining leaves a_{ijk} and $\bar{a}_{ijk}$. If we split the cost of every edge in $E^{aug} - E^{aug'}$ (evenly) between its two endpoints, then we get that at least $2p - 2q$ leaves are each charged a cost of at least $1/2$. It follows that

$$cost(E^{aug} - E^{aug'}) \geq (2p - 2q) \cdot (1/2) \qquad (2)$$

We shall now improve the lower bound (2) by considering the μ vertices w_i which do not make an appearance in M'. Each such w_i is a cut-vertex of $(V, E_1 \cup E^{aug'})$ (by definition of $E^{aug'}$), since its removal disconnects $W_i = \{w'_i, w''_i\} \cup \{a_{ijk}, \bar{a}_{ijk} : (w_i, x_j, y_k) \in M\}$ from the rest of the graph. But w_i cannot be a cut-vertex of G', and thus $E^{aug} - E^{aug'}$ must contain an edge that connects W_i to the rest of the graph. We have three cases for this edge: (i) if it is incident (in W_i) to w'_i or w''_i, then the edges cost is at least 2 and w'_i or w''_i is charged at least 1; (ii) if the edge is incident (in W_i) to some $\bar{a}_{ijk}$ or a_{ijk} and (in the rest of the graph) to some $a_{i'j'k'}$ or $\bar{a}_{i'j'k'}$ (with $i \neq i'$), then the edges cost is 2, and the endpoint in W_i is actually charged $1/2$ more than in the lower bound (2); or (iii) if this edge is incident (in W_i) to some $\bar{a}_{ijk}$ or $a_{i'j'k'}$ and (in the rest of the graph) to a vertex that is not $a_{i'j'k'}$ or $\bar{a}_{i'j'k'}$, then the edges cost is at least 1, so the endpoint not in W_i is charged at least $1/2$. In all three cases, the fact that w_i is a cut-vertex in $(V, E_1 \cup E^{aug'})$ implies that the lower bound (2) can be increased by $1/2$. It is easy to see that the increases corresponding

to different w_is are distinct, and thus,

$$cost(E^{aug} - E^{aug'}) \geq (2p - 2q) \cdot (1/2) + \mu \cdot (1/2) \qquad (3)$$

Combining equations (1) and (3) we indeed get that $\mu \leq (2+8\Delta)(p+q)\epsilon$. Therefore, instance I will contain a matching M'' where

$$|M''| \geq |M'| - \mu \geq q - 2\Delta(p+q)\epsilon - (2+8\Delta)(p+q)\epsilon = q - (2+10\Delta)(p+q)\epsilon,$$

which completes the proof of Lemma 1.

Now using lemma 1 we can prove the second part of our theorem. In any instance of 3DM-5, $\Delta = 5$ and $|M| = p \leq 5q$. Hence, from Lemma 1 we have $3DM(I) \geq q(1 - 312\epsilon)$, if $TCA_1(J) \leq (p+q)(1+\epsilon)$. Considering the contrapositive of Lemma 1 we will have if $3DM(I) < q(1-\epsilon_0)$ then $TCA_1(J) > (p+q)(1+\epsilon_0/312)$. Hence our reduction is also gap-preserving.

3.4 Tree Connectivity Augmentation - Single Fault Scenario

In this section we propose an approximation algorithm with a guaranteed performance bound for TCA_1. The input to the algorithm is a complete undirected graph $G_1 = (V, E)$ with cost function $c : E \to \mathbb{Z}^+$ defined on the edges, and $T_1 = (V, E_1)$, a spanning tree of G_1 with a specified vertex $r \in V$ as the root. The output is a set of edges $E^{aug} \subseteq E - E_1$ with minimum cost, such that, in the graph $(V, E_1 \cup E^{aug})$, there are two node disjoint paths from every node v ($v \in V - r$) to the node r. Since it is considered that the edges of $T_1 = (V, E_1)$ have already been deployed, we assume that the cost of the edges in E_1 is zero, i.e., $c(e) = 0, \forall e \in E_1$. We compute the edge set E^{aug} using a sequence of steps where, in each step, we construct a new graph/tree (undirected/directed). The sequence of construction of graphs is as follows: $[T_1 = (V, E_1)] \Rightarrow [T_2 = (V_2, E_2)] \Rightarrow [G_2 = (V_2, E_2')] \Rightarrow [T_2^d = (V_2, A_2)] \Rightarrow [G_2^d = (V_2, A_2')]$, where T_2 is a tree constructed from T_1, G_2 is a complete graph defined with the vertex set of T_2; T_2^d is a directed tree defined on T_2, and G_2^d is a completely connected directed

graph defined with the vertex set of T_2^d. From G_2^d we identify a set of *arcs* (directed edges) A_2^{aug}, so that the directed graph $(V_2, A_2 \cup A_2^{aug})$ is *strongly connected* [40]. Finally, we construct E^{aug} from A_2^{aug}. We now describe, in detail, the construction rules for these graphs/trees.

[A] Construction of T_2: Let $V_p \subset V$ be the set of leaves in T_1 and let $V_q = V - (V_p \cup \{r\})$ be the set of all internal (non-leaf) nodes except the root. We define a new tree $T_2 = (V_2, E_2)$ using the following rules:

- $V_2 = V \cup \{v_{ij} | i, j \in V - \{r\}$ and $(i, j) \in E_1\}$.
- For each edge $(i, j) \in E_1$, we include in E_2,
  - edge (i, j), if $i = r$ or $j = r$,
  - edges (i, v_{ij}) and (v_{ij}, j), otherwise.

[B] Construction of G_2: Let $G_2 = (V_2, E_2')$ be the complete graph defined on V_2. We define the cost function $c' : E_2' \rightarrow \mathbb{Z}^+ \cup \{\infty\}$ as follows. For every edge $(x, y) \in E_2'$, if $x, y \in V$, $c'(x, y) = c(x, y)$; otherwise, $c'(x, y) = \infty$ (i.e., we set the initial cost of the edges between a node $u \in V_2 - V$ and every other node in V_2 to infinity).

Next we define two functions $d(u, v)$ (distance function) and $p(u, v)$ (pointer function) for every pair of nodes u and v in G_2. We define both the functions in terms of c' and T_2.

Definition: $d(u, v) = min\{c'(x, y) | u$ and v are on the path from x to y in $T_2\}$.

Definition: $p(u, v)$ is a pointer to an edge (s, t), such that $d(u, v) = c'(s, t)$, where u and v are on the path from s to t in T_2. If there are more than one such edge, any one of them can be selected as the value of $p(u, v)$.

Example: The Fig. 3.3(a) shows a spanning tree $T_1 = (V, E_1)$ of a complete graph $G_1 = (V, E)$ (for the sake of clarity, only three edges from the set $E - E_1$, $(a, b), (c, d)$ and (d, e) are shown in Fig. 3.3(a)). The solid lines indicate the edges in E_1 and the dashed lines show a subset of the edges in $E - E_1$. The cost of each edge in $E_1 = 0$. The weights associated with the dashed lines indicate the cost of these edges. All other edges in $E - E_1$, (not shown in Fig. 3.3(a)), have a cost of 10. Fig. 3.3(b) shows the tree $T_2 = (V_2, E_2)$ constructed from T_1 in Fig. 3.3(a). From T_2, we can construct the complete graph $G_2 = (V_2, E_2')$ following the construction rules described earlier. The solid lines in Fig. 3.3(b) indicate the edges in E_2 and the dashed lines show a subset of the edges in $E_2' - E_2$ (only five edges with associated weights are shown). In this example, $d(v_{ac}, v_{ad}) = c'(c, d) = c(c, d) = 1$, $d(a, b) = c'(d, e) = c(d, e) = 1$ and $p(v_{ac}, v_{ad}) = (c, d)$, $p(a, b) = (d, e)$.

We now discuss the rationale for definitions of the $d(u, v)$ and $p(u, v)$ given above. In order to have another path from a to b in Fig. 3.3(a), (different from the one in the tree T_2, $a - r - b$), the edge (d, e) with cost 1 or the edge (a, b) (in G_2) with cost 4 can be added to the tree. The addition of the link (d, e) will result in a cheaper path from a to b with cost 1. The goal of the distance function $d(u, v)$ is to identify this edge. The function $p(u, v)$ is defined to be a pointer to the edge selected by the function $d(u, v)$. We note that since the cost of the edges

Computation of $d(u, v)$: It has been shown in [36] that $d(u, v)$ for all pairs of nodes can be computed in $O(|V|^2)$. For ease of reference, we summarize the algorithm for computing the function $d(u, v)$ presented in [36]. Initially, for every pair of nodes $u, v \in V_2, d(u, v) = c'(u, v)$ and $p(u, v) = (u, v)$. Let $l(u, v)$ be the number of edges on the path from u to v in T_2 and $s(u, v)$ be the node adjacent to v on this path. The edges $(u, v) \in E_2' - E_2$ are sorted in non-decreasing order, based on $l(u, v)$. For each edge (u, v) in the sorted order, we compute the distance function $d(u, v)$ as follows.

If $d(u,v) < d(u,s(u,v))$ then $d(u,s(u,v)) = d(u,v)$ and $p(u,s(u,v)) = (u,v)$. If $d(u,v) < d(s(v,u),v)$, then $d(s(v,u),v) = d(u,v)$ and $p(s(v,u),v) = (u,v)$.

[C] Construction of $T_2^d = (V_2, A_2)$: We construct T_2^d from $T_2 = (V_2, E_2)$ by directing all edges of T_2 towards the root node r. We will use A_2 to represent the set of *arcs* (directed edges) corresponding to the undirected edges in E_2.

[D] Construction of $G_2^d = (V_2, A_2')$: G_2^d is a completely connected directed graph, with associated cost $c''(u,v)$ with each arc $u \to v \in A_2'$ as follows:

$$c''(u,v) = \begin{cases} \infty, & \text{if } v = r, \\ \infty, & \text{if } u \in V_q \text{ and } v \in subtree(u) \\ \text{d(u,v)}, & \text{otherwise.} \end{cases}$$

Here we define $subtree(u)$ to be the set of nodes in the subtree rooted at node u in tree T_2.

We note that each arc $u \to v \in A_2$ where $v \neq r$ has a zero cost. The rationale for assigning the arc costs in this specific way is as follows:

(a) By assigning a cost of ∞ to the edges where $v = r$, we ensure that the minimum cost arborescence on G_2^d is rooted at r,

(b) By assigning a cost of ∞ to the edges (u,v) where $u \in V_q$ and $v \in subtree(u)$ in T_2, we ensure that, in G_2^d, no node $u \in V_q$ will have a directed path from u to the nodes in $subtree(u)$, unless it first goes through some nodes not in $subtree(u)$.

When constructing $G_2^d = (V_2, A_2')$, our goal is to identify a set of arcs A_2^{aug}, $(A_2^{aug} \subseteq A_2')$, so that the graph $(V_2, A_2 \cup A_2^{aug})$ is *strongly connected*, i.e., there exists a directed path between every pair of nodes in V_2. We obtain the arc set A_2^{aug} by computing the *least-cost arborescence* [41] in $G_2^d = (V_2, A_2')$, which we denote by $T_2^{arb} = (V_2, A_2^{arb})$. We obtain the set of arcs A_2^{aug} from A_2^{arb} by excluding those arcs

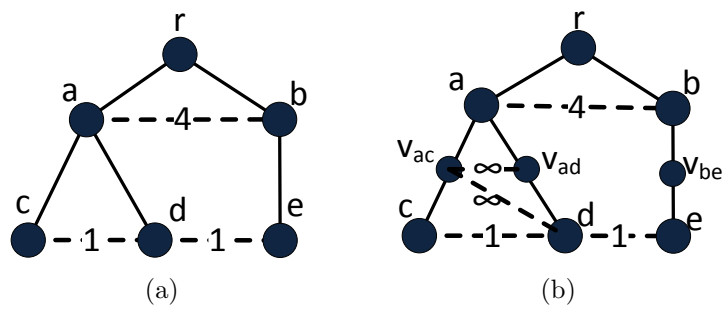


Figure 3.3: (a) An example of T_1; E_1 includes the edges shown with solid lines. (b) The tree with solid lines is T_2 corresponding to T_1 in (a) and dashed lines are some of the edges in $E_2' - E_2$.

with cost zero, i.e., $A_2^{aug} = A_2^{arb} - \{a \in A_2^{arb} | c''(a) = 0\}$. Finally, we construct the set of edges E^{aug} that we have to add to the input tree $T_1 = (V, E_1)$, to obtain the $r \odot 1$ fault tolerant graph $(V, E_1 \cup E^{aug})$ as follows: $E^{aug} = \{p(u,v) | u \to v \in A_2^{aug}\}$. We note that if A_2^{aug} has any arc whose endpoint is in $V_2 - V$, using the function $p(u,v)$ it will be replaced by an edge in E. The reason is that $c'(u,v) = \infty$ if u or v is in $V_2 - V$.

Algorithm 5 TCA_1 Algorithm

Input: $G_1 = (V, E)$, a complete graph with cost $c(e)$ for every edge $e \in E$; $T_1 = (V, E_1)$, a spanning tree of G_1 with root r.

Output: A set of edges $E^{aug} \subseteq E - E_1$, such that $(V, E_1 \cup E^{aug})$ is $r \odot 1$ fault tolerant.

1: Construct $T_2 = (V_2, E_2)$, a complete graph $G_2 = (V_2, E_2')$ and the cost function $c'(.)$ from T_1 and G_1 using the technique described in [A].
2: Compute $d(u,v)$ and $p(u,v)$ for each pair of nodes $u, v \in V_2$ using the technique described in [B].
3: Compute a directed tree $T_2^d = (V_2, A_2)$ by directing all edges in E_2 toward root r using technique described in [C].
4: Compute a completely connected directed graph $G_2^d = (V_2, A_2')$ with cost c'' defined on the arcs set A_2' using technique described in [D].
5: Compute a minimum cost arborescence $T_2^{arb} = (V_2, A_2^{arb})$ of the graph $G_2^d = (V_2, A_2')$.
6: Set $A_2^{aug} = A_2^{arb} - \{a \in A_2^{arb} | c''(a) = 0\}$.
7: Set $E^{aug} = \{p(u,v) | u \to v \in A_2^{aug}\}$.
8: Return E^{aug}.

We note that the time complexity of the TCA_1 algorithm (Algorithm 1) is $O(|V|^2)$. Line 4 has $O(|V|^2)$ time complexity. Finding minimum cost arborescence also needs $O(|V|^2)$ time [41].

Theorem 3.4.1. *Algorithm TCA_1 finds a set of edges E^{aug} such that $(V, E_1 \cup E^{aug})$ is $r \odot 1$ fault tolerant.*

Proof. In order to prove that $(V, E_1 \cup E^{aug})$ is $r \odot 1$ fault tolerant, we need to show that there is no node in V_q whose removal disconnects the graph (V_q is the set of all internal nodes in the tree $T_1 = (V, E_1)$ except the root r). Since the graph $(V_2, A_2 \cup A_2^{aug})$ is constructed by augmenting $T_2^d = (V_2, A_2)$ with the arcs of the T_2^{arb} (excluding the arcs that are already in A_2), it must be strongly connected. Accordingly, there must be a directed path from any node $v \in V_q$ to the nodes in $subtree(v)$. Let $w \neq v$ be a node in $subtree(v)$. Since there is no directed edge $\in A_2'$ going out of v to the nodes in $subtree(v)$ (because the cost of these edges is infinity), the first arc in a path from v to any other node in $subtree(v)$ should go through a node s where s is not in $subtree(v)$. Since the graph $(V_2, A_2 \cup A_2^{aug})$ is strongly connected, there must be a path from s to w in $(V_2, A_2 \cup A_2^{aug})$ not including v. Suppose that $c \rightarrow d \in A_2^{aug}$ is on the path from s to w which does not include v. If (c, d) is replaced by $p(c, d) = (e, f)$, (there is a path going through e, c, d, f in order in T_2), the new path also will not include v, because v cannot be on the path from c to e or f to d in T_2. Hence, even if a node $v \in V_q$ is removed from the graph $(V, E_1 \cup E^{aug})$, the graph remains connected. Therefore, $(V, E_1 \cup E^{aug})$ is $r \odot 1$ fault tolerant.

Theorem 3.4.2. *Algorithm TCA_1 finds a set of edges E^{aug} with a total cost C^{aug}, such that $C^{aug} \leq 2C^{opt}$, where C^{opt} is the cost of the optimal solution.*

Proof: Our proof strategy is as follows. Let C^{opt} be the optimal cost of edges E^{opt} necessary to add to the input tree $T_1 = (V, E_1)$, so that the resulting graph

becomes $r \odot 1$ fault tolerant and C^{aug} is the cost of edges E^{aug} selected by the TCA_1 Algorithm. We show that there exists a subset of arcs A'' in the graph $G_2^d = (V_2, A_2')$ with three useful properties. If C'' is the cost of the arcs in A'', $A'' \subseteq A_2' - A_2$, (i.e., $C'' = \sum_{u \to v \in A''} c''(u,v)$), then (i) $C^{opt} \geq C''/2$, (ii) $C^{aug} \leq C''$, and (iii) the graph $(V_2, A_2 \cup A'')$ is strongly connected. From (i) and (ii) it follows that $C^{aug} \leq 2C^{opt}$.

We can compute the set of arcs $A'' \subseteq A_2' - A_2$ from the optimal solution $E^{opt} \subseteq E - E_1$ following the procedure described below.

Let Q be the set of nodes that are strongly connected in $(V_2, A_2 \cup A'')$. Initially, we set $A'' = \emptyset$, $Q = \{r\}$ and mark all the edges in E^{opt} as *unused*. We update Q, A'' and the marking of the edges in E^{opt} using the following procedure:

While $Q \neq V_2$ repeat the following steps:

- Select an unused edge (u, v) from E^{opt}, such that there is a node $t \in Q - V_q$ on the weakly directed path from u to v in $T_2^d = (V_2, A_2)$. (The weakly directed path from u to v in T_2^d is the path from u to v in T_2^d in which the direction of the arcs is ignored.)
- If $t \neq u$, add $t \to u$ to A'' and if $t \neq v$ add $t \to v$ to A''.
- Add all the nodes on the weakly directed path from u to v in T_2^d to Q. Since t has been selected from set Q, it is already accessible from root r in $(V_2, A_2 \cup A'')$. Therefore, by adding the new arcs to A'' in step (ii) all the nodes on the weakly directed path from u to v in T_2^d are now accessible from root r in current augmented directed graph $(V_2, A_2 \cup A'')$.
- Change the marking of the edge (u, v) from *unused* to *used*.

We need to show that, during the execution of the iterative process, an *unused* edge $(u, v) \in E^{opt}$ and a suitable vertex $t \in Q - V_q$ exist. While there are some edges

in E^{opt} which have not been used previously, there is still some node w in V_q whose deletion disconnects some node $a \in subtree(w)$ from the root in $(V_2, A_2 \cup A'')$. So, there is no directed path from r to a in $(V_2, A_2 \cup A'')$ during that iteration. Therefore, there should be an edge (u, v) in E^{opt} which creates a path from r to $a \in subtree(w)$ such that the path does not include w in $(V, E_1 \cup E^{opt})$. Also, the path from u to v in T_1 will contain more nodes from Q than just one vertex from V_q; otherwise, the removal of that vertex would disconnect the graph $(V, E_1 \cup E^{opt})$ which contradicts the fact that $(V, E_1 \cup E^{opt})$ is $r \odot 1$ fault tolerant. Since there is no directed edge, in T_2^d, between the nodes in V_q, the weakly directed path from u to v in T_2^d should include a node $t \in Q - V_q$.

Let C'' be the cost of the arcs in A''; $C'' = \sum_{u \to v \in A''_t} c''(u, v)$. For every edge $(u, v) \in E^{opt}$, we have to add at most two arcs $t \to u$ and $t \to v$ to A''. Because $c''(t, u) \leq d(u, v)$ and $c''(t, v) \leq d(u, v)$, $C'' \leq 2C^{opt}$. Also we know that the graph $(V_2, A_2 \cup A'')$ is strongly connected. We can, therefore, construct an arborescence on $(V_2, A_2 \cup A'')$ rooted at r using c'' as the cost of the edges. Since the TCA_1 algorithm gives us the minimum cost arborescence on G_2^d and $A_2 \cup A'' \subseteq A'_2$, $C^{aug} \leq C'' \leq 2C^{opt}$.

3.5 Tree Connectivity Augmentation - Adjacent Double Fault Scenario

In this section, we propose an approximation algorithm for the TCA_2 problem. The input to the algorithm is a complete undirected graph, $G_1 = (V, E)$, with cost function $c : E \to \mathbb{Z}^+$ and a spanning tree $T_1 = (V, E_1)$ rooted at a specific node $r \in V$. The output is a set of edges $E^{aug} \subseteq E - E_1$ with minimum cost, such that the graph $(V, E_1 \cup E^{aug})$ is $r \odot 2$ fault tolerant. The cost of the edges in E_1 is considered to be zero. Since in our model we assume that r does not fail, we exclude the possibility of two adjacent node failures when one of them is r. We define $st(u, T)$ to denote the set of nodes in the subtree rooted at a node u in a tree T. If T is directed we just ignore the direction of edges. With respect to the tree T_1, we define $par(u)$ and

$s(u)$ to denote u's parent and the set of u's siblings, respectively. Also, we use $[u, v]$ to denote two adjacent nodes u and v in T_1, where $u = par(v)$ and $u \neq r$. When $[u, v]$ fails, the surviving nodes in $st(u, T_1)$ get disconnected from r. Hence, we need to augment the tree T_1 such that in the augmented graph there is a path from every node $w \in st(u, T_1), w \neq u, v$ to r and the path includes neither u nor v. In the tree augmentation algorithm TCA_2, the tree T_1 is augmented in two phases. In phase I, we find a set of edges $E_1^{aug} \subseteq E - E_1$ such that the graph $(V, E_1 \cup E_1^{aug})$ has the following properties; (1) after the failure of any two adjacent nodes $[u, v]$ the remaining nodes in $st(v, T_1)$ will have a path to r or at least to one of the nodes in $s(v)$, (2) all the nodes not directly connected to the root have two node disjoint paths to r. We note that completing phase I may not ensure the connectivity of nodes in $s(v)$ to r after the failure of $[u, v]$ and hence $(V, E_1 \cup E_1^{aug})$ may not be $r \odot 2$ fault tolerant. In phase II, we add another set of augmenting edges $E_2^{aug} \subseteq E - (E_1 \cup E_1^{aug})$ to the graph $(V, E_1 \cup E_1^{aug})$, such that the failure of any two adjacent nodes $[u, v]$ does not disconnect the nodes in $s(v)$ from r. Thus, in graph $(V, E_1 \cup E_1^{aug} \cup E_2^{aug})$, after the failure of a $[u, v]$ the remaining nodes will be connected to the root r.

In phases I and II, we only consider the failures of nodes that are adjacent in T_1. However, the edges that we added during phase I and phase II of TCA_2 introduce the possibility of additional failures, as two nodes i and j, that were previously not adjacent, will become adjacent after the augmentation process. We will explain later how we deal with this possibility and ensure that the augmented graph is $r \odot 2$ fault tolerant.

3.5.1 TCA$_2$ Phase I

In this phase we propose an algorithm to find a set of edges $E_1^{aug} \subseteq E - E_1$ such that in graph $(V, E_1 \cup E_1^{aug})$ for every $[u, v]$ and node $w \in st(v, T_1)$, w has a path to the root or to a node in $s(v)$ that does not include either u or v. We use an approach

similar to that in Section 3.4 for the TCA_1 algorithm where we created a sequence of graphs to find the augmenting edges. In this case, since we need to deal with the failures of two adjacent nodes, the algorithm and the proofs are more complicated.

First, we compute $d(u, v)$ and $p(u, v)$ functions defined in Section 3.4 [B] for every pair of nodes u and $v \in V$ using T_1 and c as inputs, so that $d(u, v) = min\{c(x, y)|u$ and v are in the path from x to y in $T_1\}$ and $p(u, v)$ is the pointer to the edge selected by the function $d(u, v)$. Next, we construct a new directed tree T_2^d and directed graph G_2^d as follows.

[A] Construction of $T_2^d = (V_2, A_2)$: We construct T_2^d from T_1 by directing all edges of T_1 toward the root node r. Then for every three consecutive nodes $k \to j \to i$ where $i \neq r$ we add a new node v_{ijk} between j and k and replace the directed edge $k \to j$ with $k \to v_{ijk}$ and $v_{ijk} \to j$; i.e, $V_2 = V \cup \{(v_{ijk})|i, j, k \in V$ and $j = par(k)$ and $i = par(j)$ and $i \neq r\}$ and $A_2 = \{j \to i|i = par(j)$ and $(i = r$ or $par(i) = r)\} \cup \{k \to v_{ijk}, v_{ijk} \to j|j = par(k)$ and $i = par(j)$ and $i \neq r\}$. In the Fig. 3.4(a) the tree formed by solid lines is an example of T_1 and the tree with solid arrows in Fig. 3.4(b) is the corresponding T_2^d.

[B] Construction of $G_2^d = (V_2, A_2')$: G_2^d is a directed graph on V_2. We use another pointer function $p' : (A_2' - A_2) \to E$, where $p'(i, j)$ is an undirected edge in E corresponding to the arc $i \to j \in (A_2' - A_2)$. We compute A_2', p' and the cost of the arcs in A_2', c' in the following way:

(i) For every edge $i \to j \in A_2$:

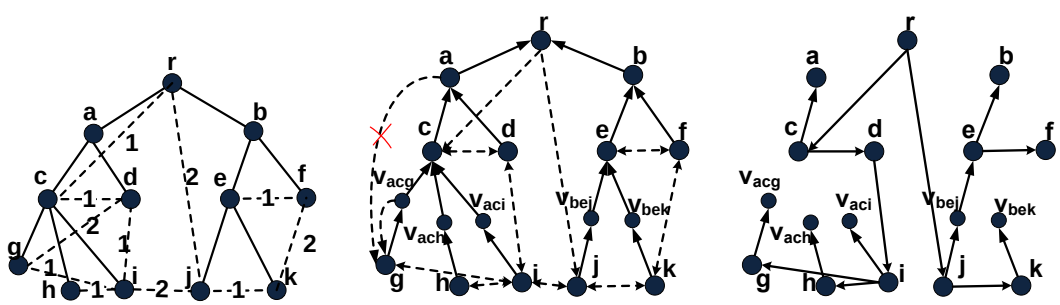
- If $j \neq r$, add $i \to j$ to A_2', $c'(i, j) = 0$.
- Else add $j \to i$ to A_2', $c'(j, i) = 0$, $p'(j, i) = (i, j)$.

(ii) For every edge $(i, j) \in E - E_1$:

- If $j \in st(i, T_1)$ and $i \neq r$, add $v_{ikl} \to j$ to A'_2 where there is a directed path from j to v_{ikl} in T_2^d. In this case, $p'(v_{ikl}, j) = (i, j)$, $c'(v_{ikl}, j) = d(i, j)$.
- Else if $j \neq r$, add $i \to j$ to A'_2, $p'(i, j) = (i, j)$, $c'(i, j) = d(i, j)$.

(iii) Step (ii) is repeated by interchanging i and j.

In Fig. 3.4(a) dashed lines show a subset of edges in $E - E_1$ and the value on every edge (i, j) shows its cost $c(i, j)$. Other edges in $E - E_1$ that are not shown in the figure are supposed to have cost 10. In Fig. 3.4(b) a subset of arcs in $A'_2 - A_2$ are shown in dashed arrows. We note that the dashed arrow with cross on it means that the arc $a \to g$ is not in A'_2 and instead $v_{acg} \to g$ is in A'_2 and $p'(v_{acg}, g) = (a, g)$.



(a) (b) (c)

Figure 3.4: (a) An example of T_1; solid lines show the edges in E_1 and dashed ones show a subset of edges in $E - E_1$. (b)T_2^d corresponding to T_1 in part (a); solid arrows show the arcs in T_2^d and dashed ones show a subset of arcs in $A'_2 - A_2$. (c) T_2^{arb}, minimum cost arborescence computed on G_2^d corresponding to part (b).

Similar to the algorithm TCA_1 we augment the directed tree T_2^d with some additional arcs $A_2^{aug} \subseteq A'_2$ such that the augmented graph $(V_2, A_2 \cup A_2^{aug})$ is *strongly connected*. In order to compute A_2^{aug}, we compute the *minimum cost arborescence* rooted at r on graph $G_2^d = (V_2, A'_2)$ using c' and denote it by $T_2^{arb} = (V_2, A_2^{arb})$. Fig. 3.4(c) depicts T_2^{arb} corresponding to the example depicted in 3.4(a). The

set of arcs A_2^{aug} is obtained by excluding from A_2^{arb} those arcs with cost zero, i.e., $A_2^{aug} = A_2^{arb} - \{a \in A_2^{arb} | c'(a) = 0\}$. Finally, the set of edges E_1^{aug} is obtained by including in the set E_1^{aug} the edge $p(p'(u,v))$ for each $u \rightarrow v \in A_2^{aug}$, i.e., $E_1^{aug} = \{p(p'(u,v)) | u \rightarrow v \in A_2^{aug}\}$. In the example shown in Fig. 3.4, we get $E_1^{aug} = \{(r,c),(c,d)\ ,(d,i),(g,i),(h,i),\ (r,j),(e,f),(j,k)\}$. The augmented graph $(V, E_1 \cup E_1^{aug})$ is shown in Fig. 3.5(a). The steps of TCA_2 *Phase I* are summarized in Algorithm 6 Phase I.

Theorem 3.5.1. *Algorithm TCA_2 Phase I finds a set of edges E_1^{aug} such that in the graph $(V, E_1 \cup E_1^{aug})$ if any two adjacent nodes $[i,j] \in T_1$ fail, the remaining nodes in $st(j, T_1)$ have a path to r or to a node in $s(j)$. The graph is also $r \odot 1$ fault tolerant.*

Proof. Let $G_{tmp} = (V, E_1 \cup E_{tmp})$ be an undirected graph on V where $E_{tmp} = \{p'(u,v) | u \rightarrow v \in A_2^{aug}\}$. Consider the two adjacent nodes $[i,j]$ in T_1 such that their removal disconnects $st(k, T_1)$ from r where k is one of the j's children. We know that $(V_2, A_2 \cup A_2^{aug})$ is *strongly connected*. So, there should be a directed path from j to k in $(V_2, A_2 \cup A_2^{aug})$. Since there is no arc in A_2' from j to the nodes in $st(j, T_2^d)$, there should be a directed path, P, from j to k such that the node following j on P, called f is not in $st(j, T_2^d)$ and the subpath from f to k does not include j. There are three possibilities for f: (i) f can be v_{hij}. The node after v_{hij} on P can be either i or a node $a \in st(j, T_2^d)$. If it is i, as there is no directed edge from i to its subtree in A_2' the next node on P is some node $\notin st(i, T_2^d)$. So, in this case there is a directed path from some node $\notin st(i, T_2^d)$ to k that includes neither i nor j. In second case there will be a directed path from v_{hij} to k which does not include i and j. Because of the way p' is computed the edge from v_{hij} to a is replaced by h (i's parent) to a in G_{tmp} . Hence, in both cases, in G_{tmp} there will be a path from r to k that includes neither i nor j. (ii) f can be a node $\notin st(i, T_2^d)$. Then there is a path from f to k that does not include either i or j. (iii) f can be a node in a subtree of a j's sibling $\in s(j)$. We

denote this j's sibling by b. Then there will be a subpath from f to k that the path includes neither i nor j. In this case in G_{tmp} after removal of $[i, j]$, nodes $\in st(k, T_1)$ have a path to b. So, in all cases, when $[i, j]$ fails, $st(j, T_1)$ is connected to r or to a node $\in s(j)$ in G_{tmp}.

Now it remains to show that after replacing (w, x) in E_{tmp} with $p(w, x)$, k still remains connected to r or a node $\in s(j)$. Let $p(w, x) = (y, z)$; If (w, x) is on a path from k to r or k to j's sibling that eliminates i and j as two adjacent node failure in G_{tmp}, then definitely i and j cannot be on the path from w to y and z to x in T_1. Therefore, after replacing (w, x) in E_{tmp} with $p(w, x)$, k has a path to root or a j's sibling not including i and j.

Theorem 3.5.2. *Let C_1^{aug} be total cost of the edges in E_1^{aug}. We claim that $C_1^{aug} \leq 2C^{opt}$ where C^{opt} is the cost of the optimal solution for TCA_2 problem.*

Proof. In order to prove the theorem, we find a set of arcs $A_1'' \subseteq A_2'$ using the edges in the optimal solution E^{opt} such that graph $(V_2, A_2 \cup A_1'')$ is strongly connected and total cost of the arcs in A_1'', $C_1'' \leq 2C^{opt}$ ($C_1'' = \sum_{u \to v \in A_1''} c'(u, v)$).

Let $E_1^{opt} = E^{opt} - \{(i, l) | i = par(par(l)) \text{ and } i \neq r\}$ and Q_1 be the set of nodes which are strongly connected in $(V_2, A_2 \cup A_1'')$. Initially, $Q_1 = \{r\}$, $A_1'' = \emptyset$ and all the edges in E_1^{opt} are marked *unused*. We use the following procedure to compute A_1'' and Q_1.

While $Q_1 \neq V_2$ repeat the following steps:

- Select an unused edge (i, j) from E_1^{opt} such that one of the following conditions holds and update A_1'' and Q_1 accordingly:
  - The *lowest common ancestor* of nodes i and j, $LCA(i, j)$, in T_1 is r ($LCA(i, j)$ in a tree is the shared ancestor of i and j that is located far-

thest from the root. A node is allowed to be an ancestor of itself). If $r \neq i$ add the directed edge $r \to i$ and if $r \neq j$ add $r \to j$ to A_1''. Also, add all the nodes on the weakly directed path from i to j in T_2^d to Q_1 since these nodes are strongly connected with root r in current $(V_2, A_2 \cup A_1'')$. At least one of the edges in E_1^{opt} has this condition. Otherwise, in $(V, E_1 \cup E_{opt})$ none of the nodes could have two disjoint paths to r which is a contradiction. We note that because of the way we compute cost of the arcs in A', $c'(r,i) \leq c(i,j)$ and $c'(r,j) \leq c(i,j)$.

- $i \in Q_1$, $j \notin st(i, T_1)$ and $j \neq r$. Add $i \to j$ to A_1'' and add all the nodes on the weakly directed path from j to i in T_2^d to Q_1. (This condition should be checked for the case that i and j are interchanged.)
- $LCA(i,j) = k$, $k \in Q_1$ and $k \notin \{r, i, j\}$. There is a node $v \in Q_1 \cap V$ such that $par(v) = k$ and i or j is in $st(v, T_1)$. Let $i \in st(v, T_1)$. Add arc $v \to j$ and if $v \neq i$, add $j \to i$ to A_1''. Since v is on the path from i to j in T_1, $c'(v,j) \leq c(i,j)$. Then, add the nodes on the weakly directed path from i to j in T_2^d to Q_1.
- $LCA(i,j) = k$, $k \in Q_1$, $k \neq r$ and $k \in \{i, j\}$. There is a node $v_{klm} \in V_2 - V$ such that i or j is in $st(v_{klm}, T_2^d)$ and $v_{klm} \in Q_1$. Let $i \in st(v_{klm}, T_2^d)$. Add arc $v_{klm} \to i$ to A_1''. Obviously $c'(v_{klm}, i) = d(k,i) \leq c(i,j)$. Then, add the nodes on the weakly directed path from i to j in T_2^d to Q_1.

- Change the marking of the edge (i,j) to *used*.

Now we need to show that while $Q_1 \neq V_2$ there is some *unused* edge (i,j) such that one of the conditions in the procedure holds for it. Let $E_{used} \subseteq E_1^{opt}$ be the set of edges marked *used* during the procedure. Assume that $Q_1 \neq V_2$; so, there exists some node $x \in V_2 \cap V$ that is not reachable from r. Definitely, the nodes $\in st(x, T_2^d)$

are not reachable from r either. If x has some sibling y in T_1, that $y \in Q_1$ then there is no directed path from y to x in $(V_2, A_2 \cup A_1'')$; otherwise, x would be reachable from r and it would be $\in Q_1$. Also, the directed path from x to y in $(V_2, A_2 \cup A_1'')$ can only go through $par(x)$, w, since during the procedure only directed edges from nodes $\in Q_1$ to other nodes can be added. In this case if $[par(w), w]$ fails, in the graph $(V, E_1 \cup E_{used})$, x gets disconnected both from r and y. So, there should be some *unused* edge $(i, j) \in E_1^{opt}$ that x is on the path from i to j in T_1 and one of the above conditions holds for it. Otherwise, it contradicts with E^{opt} to be the optimal solution. If $y \notin Q_1$ then similarly it can be concluded that in the graph $(V, E_1 \cup E_{used})$ failure of $[par(w), w]$ disconnects both x and y from r and from each other. So, there should be some *unused* edges $\in E_1^{opt}$ that one of the above conditions holds and make x and y connected to the root or one of their siblings.

Based on this procedure, for every edge (i, j) in E_1^{opt} at most two arcs from A_2' are added to A_1'' such that each one has cost $\leq c(i, j)$. Therefore, total cost of the arcs in A_1'', $C_1'' \leq 2C^{opt}$. We know that $(V_2, A_2 \cup A_1'')$ is strongly connected. So, on $(V_2, A_2 \cup A_1'')$ we can construct an arborescence tree using c' as the cost of the edges. Since $(A_2 \cup A_1'' - \{i \to r | i \in V\}) \subseteq A_2'$ and in TCA_2 *Phase I* we compute minimum arborescence tree on (V_2, A_2'), $C_1^{aug} \leq C_1'' \leq 2C^{opt}$.

3.5.2 TCA$_2$ Phase II

After adding the edges E_1^{aug} obtained in TCA_2 *Phase I* to E_1 it may still happen that the failure of two adjacent nodes $[i, j]$ disconnects some nodes in $s(j)$ from root r. It may also disconnect the nodes in $st(j, T_1)$ from r, but based on Theorem 3.5.1 these nodes are connected to at least a node in $s(j)$ in graph $(V, E_1 \cup E_1^{aug})$. Hence, we need to find a new set of edges $E_2^{aug} \subseteq E - (E_1 \cup E_1^{aug})$ such that in graph $(V, E_1 \cup E_1^{aug} \cup E_2^{aug})$ after the failure of any two adjacent nodes $[i, j]$, nodes in $s(j)$ remain connected to r.

We define a cost function c'' for $(i,j) \in E$ in the following way. If $(i,j) \in E_1^{aug}$ then $c''(i,j) = 0$, else $c''(i,j) = c(i,j)$. Again we need to compute functions $d(i,j)$ and $p(i,j)$ defined in section 3.4 [B] for every pair of nodes u and $v \in V$ but using T_1 and c'' as input; i.e., $d(u,v) = min\{c''(x,y)|u$ and v are in the path from x to y in $T_1\}$ and $p(u,v)$ is the pointer to the edge selected by the function $d(u,v)$.

Let $S = \{[i,j]|i \neq r,\ i = par(j)$ in T_1 and failure of $[i,j]$ disconnects the graph $(V, E_1 \cup E_1^{aug})\}$. If S is empty it means that there is no $[i,j]$ whose removal disconnects $(V, E_1 \cup E_1^{aug})$ and E_2^{aug} will be $\emptyset$. Otherwise, for every $[i,j] \in S$ we define a set D_j to be the set of connected components that do not include r in graph $(V, E_1 \cup E_1^{aug})$ caused by failure of $[i,j]$. Let $\mathcal{C}_{jk}$ be the kth component in D_j. We note that each component $\mathcal{C}_{jk} \in D_j$ includes at least a node from $s(j)$. Now, we construct a new directed tree T_3^d and directed graph G_3^d as follows.

[A] Construction of $T_3^d = (V_3, A_3)$: We construct T_3^d from T_1 by directing all edges of T_1 toward the root node r. Then for every two consecutive nodes $j \to i$ where $i \neq r$ we add a new node u_{ij} and replace the directed edge $j \to i$ with $j \to u_{ij}$ and $u_{ij} \to i$; For every $[i,j] \in S$ and every component $\mathcal{C}_{jk} \in D_j$ we add two new nodes x_{jk} and y_{jk} to V_3 and add the arcs $x_{jk} \to y_{jk}$ and $y_{jk} \to j$ to A_3.

The solid arrows in Fig. 3.5(b) shows the directed tree T_3^d corresponding to the example in Fig. 3.4. It can be seen that there are two nodes x_{i1} and x_{i2} associated with node i. The reason is that after failure of $[c,i]$ in $(V, E_1 \cup E_1^{aug})$ nodes g and h get disconnected from r and they are in two separate components.

[B] Construction of $G_3^d = (V_3, A_3')$: Let $p'' : (A_3' - A_3) \to E$ be a pointer function where $p''(i,j)$ is an undirected edge in E pointing to an arc $i \to j \in (A_3' - A_3)$ and $c'''(i,j)$ be the cost of the arcs in A_3'. We compute A_3' and c''' through the following procedure:

(i) For every edge $i \to j \in A_3$:

- If $j \neq r$, add $i \to j$ to A'_3, $c'''(i,j) = 0$.
- Else add $j \to i$ to A'_3, $c'''(j,i) = 0$, $p''(j,i) = (i,j)$.

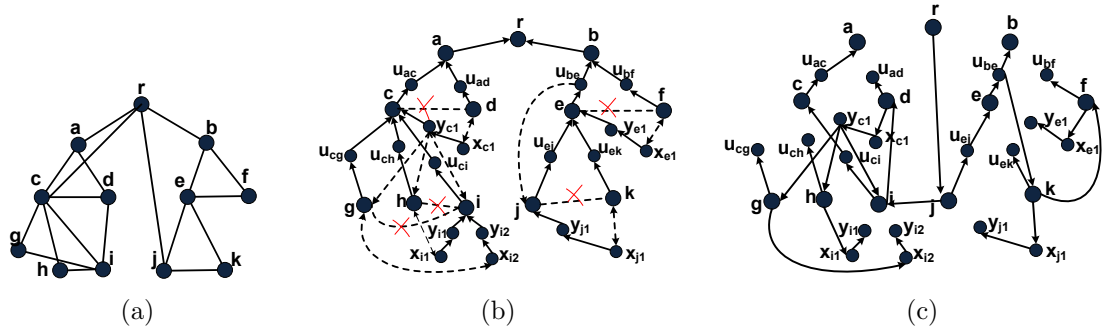
(ii) For every edge $(i,j) \in E - E_1$:

- If $j \in st(i, T_1)$ and $i \neq r$ add one of the following arcs to A'_3: Let k be the i's child where $j \in st(k, T_1)$ in T_1.
  - If failure of $[i,k]$ disconnects j from r in $(V, E_1 \cup E_1^{aug})$, i.e., j is in a component $\mathcal{C}_{kl}$ in D_k, add $y_{kl} \to j$ to A'_3; $p''(y_{kl}, j) = (i,j)$; $c'''(y_{kl}, j) = d(i,j)$;
  - else, add $u_{ik} \to j$ to A'_3; $p''(u_{ik}, j) = (i,j)$ and $c'''(u_{ik}, j) = d(i,j)$.
- Else if $par(j) \neq r, j \neq r$ and failure of $[par(j), j]$ disconnects i from r in $(V, E_1 \cup E_1^{aug})$, add $i \to x_{jz}$ to A'_3 where $i \in \mathcal{C}_{jz}$ for a component number z. Also, $p''(i, x_{jz}) = (i,j)$ and $c'''(i, x_{jz}) = d(i,j)$.
- Else if $par(i), i \neq r$ and removal of $[par(i), i]$ from $(V, E_1 \cup E_1^{aug})$ disconnects j from r, add $x_{iw} \to j$ to A'_3 where $j \in \mathcal{C}_{iw}$ for a component number w; Also, $p''(x_{iw}, j) = (i,j)$ and $c'''(x_{iw}, j) = d(i,j)$;
- Else if $j \neq r$ add $i \to j$ to A'_3 and $p''(i,j) = (i,j)$ and $c'''(i,j) = d(i,j)$.

(iii) Repeat step (ii) by interchanging i and j.

In Fig. 3.5(b) the dashed lines without a cross show a subset of the arcs in $A'_3 - A_3$. In graph $(V, E_1 \cup E_1^{aug})$ the failure of $[a,c]$ disconnects c's children from r but they remain connected to d. Therefore, corresponding to the edge $(a,g) \in E - E_1$ the arc $y_{c1} \to g$ is added to A'_3 and there is no arc in A'_3 from the node u_{ac} to its

subtree. However, after the failure of $[b, e]$, j and k remain connected to r. So, the edge $u_{be} \to j$ and $u_{be} \to k$ are in A'_3 corresponding to the edges (b, j) and (b, k) in $E - E_1$ respectively.



(a) (b) (c)

Figure 3.5: (a) Solution of TCA$_2$ Phase I, i.e., $(V, E_1 \cup E_1^{aug})$. (b) Solid arrows form the tree T_3^d corresponding to the example shown in Fig. 3.4. (c) T_3^{arb}

We augment T_3^d with a set of additional edges A_3^{aug} such that $(V_3, A_3 \cup A_3^{aug})$ is strongly connected. A_3^{aug} is obtained by computing minimum cost arborescence tree $T_3^{arb} = (V_3, A_3^{arb})$ on G_3^d (Fig. 3.5(c)). T_3^{arb} is rooted at r as in A'_3 there is no edge from the nodes in V_3 to r. The set of arcs A_3^{aug} is obtained by excluding from A_3^{arb} the arcs with cost zero; i.e., $A_3^{aug} = A_3^{arb} - \{a \in A_3^{arb} | c'''(a) = 0\}$. Finally, the set of edges E_2^{aug} is obtained by including in the set E_2^{aug} the value of $p(p''(u, v))$ for each $u \to v \in A_3^{aug}$, i.e., $E_2^{aug} = \{p(p''(u, v)) | u \to v \in A_3^{aug}\}$. In the example depicted in Fig. 3.5, $A_3^{aug} = \{j \to i, u_{be} \to k, k \to f\}$ and using functions p'' and p, we have $E_2^{aug} = \{(i, j), (k, f)\}$.

Theorem 3.5.3. *Algorithm TCA$_2$ Phase II finds a set of edges E_2^{aug} such that in the graph $(V, E_1 \cup E_1^{aug} \cup E_2^{aug})$ after removal of any two adjacent nodes $[i, j]$ the remaining nodes still have path to r.*

Proof. Based on Theorem 3.5.1, in graph $(V, E_1 \cup E_1^{aug})$ if $[i, j]$ fails, the nodes in $st(j, T_1)$ remain connected to r or at least to one of the nodes in $s(j)$. Now, we prove that in $(V, E_1 \cup E_1^{aug} \cup E_2^{aug})$ after failure of $[i, j]$, nodes in $s(j)$ still

have a path to r. Let $G_{tmp} = (V, E_1 \cup E_1^{aug} \cup E_{tmp})$ be an undirected graph where $E_{tmp} = \{p''(u,v) | u \to v \in A_3^{aug}\}$. Now, consider $[i,j]$ whose failure disconnects j's sibling, k, from r. We know that $(V_3, A_3 \cup A_3^{aug})$ is strongly connected. Hence, there should be a directed path from j to k in $(V_3, A_3 \cup A_3^{aug})$. Because of the way we defined the arcs in A_3' we know that there is no arc in A_3' from j to any node in $st(k, T_3^d)$ and no edge from j to any other node that in $(V, E_1 \cup E_1^{aug})$ gets disconnected by failure of $[i,j]$. Also, we know there is no edge in A_3' from j to the nodes in $st(j, T_3^d)$. So the node following j, f on the directed path from j to k is either some node out of $st(i, T_3^d)$ or f is u_{ij}. In first case, in G_{tmp} there will be a path from some node w out of $st(i, T_1)$ to k that the path includes neither i nor j. Since $w \notin st(i, T_1)$ after removal of $[i,j]$, it remains connected to r in G_{tmp}. Therefore, k remains connected to r in G_{tmp}. In second case, the node after u_{ij} is some node $t \in st(j, T_1)$ such that the subpath from t to k includes neither i nor j. Because of the way we defined A_3', there is an edge in A_3' from u_{ij} to t if t remains connected to the root in $(V, E_1 \cup E_1^{aug})$ when $[i,j]$ fails. So, there should be a path from t to r in G_{tmp} where it includes neither i nor j. So, in G_{tmp} there is a path from k to r which eliminates i and j. Similar to the proof of Theorem 3.5.1, it can be shown if edge $(a,b) \in E_{tmp}$ is on a path from k to r in G_{tmp} such that the path eliminates $[i,j]$ ($k \in st(i, T_1)$), after replacing it by $p(a,b) = (w,z)$, there will be a new path that includes neither i nor j. The reason is that i and j cannot be on the path from a to w and z to b in T_1.

Theorem 3.5.4. *Let C_2^{aug} be the total cost of the edges in E_2^{aug}. We claim that $C_2^{aug} \leq 2C^{opt}$ where C^{opt} is the cost of the optimal solution.*

Proof. We know that in $(V, E_1 \cup E_1^{aug})$ the nodes $i \in \mathcal{C}_{jk}$, kth component in D_j, get disconnected from r after failure of $[par(j), j]$. Moreover, in $(V, E_1 \cup E_1^{aug})$ there are two disjoint paths from i to r. Hence, one of the paths from i to r goes through j that does not include $par(j)$. This subpath from i to j includes only nodes

Algorithm 6 TCA_2 Algorithm

Input: The same as input in Algorithm 5.

Output: A set of edges $E^{aug} \subseteq E - E_1$ such that $(V, E_1 \cup E^{aug})$ is $r \odot 2$ fault tolerant.

Phase I

1: Compute $d(u, v)$ and $p(u, v)$ for each pair of nodes $u, v \in V$ using T_1 and c as the inputs.
2: Compute a directed tree $T_2^d = (V_2, A_2)$ using the technique described in [A] in section 3.5.1
3: Compute a directed graph $G_2^d = (V_2, A_2')$, functions $c'(.)$ and $p'(.)$ using the technique described in [B] in section 3.5.1.
4: Compute a minimum cost arborescence $T_2^{arb} = (V_2, A_2^{arb})$ of the graph $G_2^d = (V_2, A_2')$.
5: Set $A_2^{aug} = A_2^{arb} - \{a \in A_2^{arb} | c'(a) = 0\}$.
6: Set $E_1^{aug} = \{p(p'(u, v) | u \to v \in A_2^{aug}\}$.

Phase II

1: Compute the cost function c'': if $((u, v) \in E_1^{aug})$ $c''(u, v) = 0$; else, $c''(u, v) = c(u, v)$
2: Compute $d(u, v)$ and $p(u, v)$ for each pair of nodes $u, v \in V$ using T_1 and cost function c''.
3: Compute the set $S = \{[i, j] | (i, j) \in E_1, i \neq r$ and failure of $[i, j]$ disconnects the graph $(V, E_1 \cup E_1^{aug})\}$ and D_j $\forall [i, j] \in E_1^{aug}$
4: Compute the directed tree $T_3^d = (V_3, A_3)$ using the technique described in [A] in section 3.5.2
5: Compute the directed graph $G_3^d = (V_3, A_3')$, functions $c'''(.)$ and $p''(.)$ using the technique described in [B] in section 3.5.2.
6: Compute the minimum cost arborescence $T_3^{arb} = (V_3, A_3^{arb})$ of the graph $G_3^d = (V_3, A_3')$.
7: Set $A_3^{aug} = A_3^{arb} - \{A_3^{arb} | c'''(a) = 0\}\}$.
8: Set $E_2^{aug} = \{p(p'(u, v) | u \to v \in A_3^{aug}\}$.
9: Remove edges (u, v) from $E_1^{aug} \cup E_2^{aug}$ if failure of u and v disconnects $(V, E_1 \cup E_1^{aug} \cup E_2^{aug})$
10: Return $E_1^{aug} \cup E_2^{aug}$

in $\mathcal{C}_{jk} \cup \{j\}$. Also, we know that in TCA_2 *Phase II* cost of the edges in $E_1 \cup E_1^{aug}$ is zero. Corresponding to this undirected subpath from i to j there is a directed path in G_3^d from i to x_{jk} with cost zero. We denote this zero cost directed path from i to j in G_3^d by $P0_{ij}$.

In order to prove the theorem we find a set of directed edges $A_3'' \subseteq A_3'$ using the edges in the optimal solution E^{opt} such that $(V_3, A_3 \cup A_3'')$ is strongly connected and $\sum_{u \to v \in A_3''} c'''(u,v) \leq 2C^{opt}$.

Let Q_2 show the set of nodes which are strongly connected in $(V_3, A_3 \cup A_3'')$. Initially, $Q_2 = \{r\}$, $A_3'' = \emptyset$ and all the edges in E^{opt} are marked *unused*. We use the following procedure to compute A_3'' and Q_2.

While $Q_2 \neq V_3$ repeat the following steps:

1. Select an unused edge (i,j) from E^{opt} such that one of the following conditions holds:

(i) $LCA(i,j) = r$. In this case if $r \neq i$ add the arc $r \to i$ and if $r \neq j$ add $r \to j$ to A_3''. It should be noted that $c'''(r,i) \leq c(i,j)$ and $c'''(r,j) \leq c(i,j)$. Then go to step 2.

(ii) $LCA(i,j) = k$, $k \in Q_2$, $k \neq r$ and $k \in \{i,j\}$. Let $q \in \{i,j\}$ where $q \neq k$. Also, one of the following conditions holds.

- There is a node $u_{kl} \in V_3 \cap Q_2$ such that $q \in st(u_{kl}, T_3^d)$. Also, $u_{kl} \to q \in A_3'$. Add $u_{kl} \to q$ to A_3''. Obviously $c'''(u_{kl}, q) = d(k,q) \leq c(i,j)$.
- There is a node $y_{lm} \in Q_2$ where $l \in V, par(l) = k, q \in st(l, T_1)$ and $q \in \mathcal{C}_{lm}$. Add $y_{lm} \to q$ to A_3''.

(iii) $LCA(i,j) = k$, $k \in Q_2$, $k \notin \{r,i,j\}$. There is a node $v \in Q_2 \cap V$ such that $par(v) = k$ and i or j is in $st(v, T_1)$. Let $i \in st(v, T_1)$. If j is in a component $\mathcal{C}_{vm} \in D_v$

and $x_{vm} \in Q_2$, add $x_{vm} \to j$. Else if v is in a component $\mathcal{C}_{jm'} \in D_j$ add $v \to x_{jm'}$. Else, if v is not in any component in D_j add $v \to j$ to A''_3. Similarly, if $v \neq i$ add $j \to i$ or $x_{jp} \to i$ or $j \to x_{ip'}$. More precisely, if j is not in any components of D_i and i is not in any component of D_j, add $j \to i$. Else if i is in a component $\mathcal{C}_{jp} \in D_j$ and $x_{jp} \in Q_2$ add $x_{jp} \to i$. Else if j is in a component $\mathcal{C}_{ip'} \in D_i$ add $j \to x_{ip'}$.

In all cases at most two arcs with cost $\leq c(i,j)$ are added to A''_3. Then go to step 2.

2. For every new arc $a \to b$ added to A''_3 in previous step add every node k on the weakly directed path from a to b in T^d_3 to Q_2. Also, if $k \in V$ and k is in some component $\mathcal{C}_{lm} \in D_l$ for every node $l \in s(k)$, add the arcs on the path $P0_{kl}$ to A''_3; and add every node t on $P0_{kl}$ and every node on the path from t to r in T^d_3 to Q_2. We note that all the arcs on path $P0_{kl}$ are have cost zero.

3. Change the marking of the edge (i,j) to *used*.

We need to show that while $Q_2 \neq V_3$ there is some *unused* edge $(i,j) \in E^{opt}$ such that one of the conditions in the procedure holds for it. In each iteration Q_2 shows the nodes from V_3 that are strongly connected with r in $(V_3, A_3 \cup A''_3)$. Assume that $Q_2 \neq V_3$; so, there should be some node $k \in V_3 \cap V$ where it is not accessible from r (Note that if all the nodes in $V_3 \cap V$ are in Q_2 then all the other nodes $\in V_3 - V$ can definitely get reached through zero cost arcs and can be added to Q_2). Hence, the nodes in $st(k, T^d_3)$ are not accessible from r and there is no directed path from the nodes in Q_2 to the nodes in $st(k, T^d_3)$ in $(V_3, A_3 \cup A''_3)$. Also, the directed path from k to the nodes in Q_2 can only go through k's parent, t. In this case if t fails then k gets disconnected from r. Hence $(V, E_1 \cap E_{used})$ will get disconnected after failure of t. So, $E_{used} \neq E^{opt}$ and there should be some unused edge from a node in $st(k, T_1)$ to some node in $st(j, T_1)$ where j is a node in Q_2. Therefore, there should

be some unused edge $\in E^{opt}$ that one of the above conditions holds for it. Otherwise, it contradicts with E^{opt} to be the optimal solution.

Based on this procedure, for every edge (i, j) in E^{opt} at most two nonzero arcs from A'_3 are added to A''_3 such that each one has cost $\leq c(i, j)$. Therefore, total cost of the arcs in A''_3, $C''_2 \leq 2C^{opt}$. We know that $(V_3, A_3 \cup A''_3)$ is strongly connected. So, on $(V_3, A_3 \cup A''_3)$ we can construct an arborescence tree rooted at r using c''' as the cost of the edges. Since $A_3 \cup A''_3 - \{i \to r | i \in V\} \subseteq A'_3$ and in TCA_2 *Phase II* we compute minimum arborescence tree on (V_3, A'_3), $C_2^{aug} \leq C''_2 \leq 2C^{opt}$.

As we mentioned before, the edges that are added during TCA_2 *Phase I* and *Phase II* to T_1 introduces possibility of additional two adjacent node failures since two nodes i and j that were not adjacent before augmentation will be adjacent in graph $(V, E_1 \cup E_1^{aug} \cup E_2^{aug})$ if $(i, j) \in E_1^{aug} \cup E_2^{aug}$. In the following theorem we show that if any new two adjacent node failure $\{i, j\}$ results in disconnection in $(V, E_1 \cup E_1^{aug} \cup E_2^{aug})$ the edge (i, j) can be removed from $E_1^{aug} \cup E_2^{aug}$ and the remaining graph will be $r \odot 2$ fault tolerant. Therefore, the final set of augmenting edges, E^{aug} is constructed in the following way: For every edge $(i, j) \in E_1^{aug} \cup E_2^{aug}$ where $i \neq r$ and $j \neq r$, if removal of both nodes i and j does not disconnect the graph $(V, E_1 \cup E_1^{aug} \cup E_2^{aug})$, edge (i, j) is added to E^{aug}. Algorithm 6 shows the steps in algorithm TCA_2. Similar to Algorithm TCA_1, the time complexity of TCA_2 *Phase I* is $O(|V^2|)$. In the third step in *Phase II*, $S = O(|V|)$ and since the edges in E_1^{aug} is computed based on T_2^{arb}, its cardinality is $O(|V|)$. If we use depth first search to find the components for a two adjacent node failure $[i, j]$ it takes $O(|V| + |E_1 \cup E_1^{aug}|) = O(|V|)$. So, time complexity of this step is $O(|V|^2)$. Similarly, step 9 takes $O(|V|^2)$. Therefore, time complexity of Algorithm 6 is $O(|V|^2)$.

Theorem 3.5.5. *The augmented graph* $(V, E_1 \cup E^{aug})$ *is* $r \odot 2$ *fault tolerant.*

Proof. Consider that $(i, j) \in E_1^{aug} \cup E_2^{aug}$ and failure of $i \neq r$ and $j \neq r$ disconnects the graph $(V, E_1 \cup E_1^{aug} \cup E_2^{aug})$. Let $u \neq i, j$ be one of the nodes that gets disconnected from root when i and j fail. u should be in $st(i, T_1)$ or $st(j, T_1)$. We know that in $(V, E_1 \cup E_1^{aug} \cup E_2^{aug})$ there are two node disjoint paths from node u to r. So one path should go through i and not include j and the other path should go through j and not include i. So there will be a cycle containing u, j, r, i, u in order. Edge (i, j) is a chord in this cycle which divides the cycle into two sub-paths, one from i to j which includes r and the other is from i to j and it includes u. There cannot exist an edge from the nodes on first sub-path to the nodes on the second sub-path of the cycle; otherwise, u would not get disconnected in graph $(V, E_1 \cup E_1^{aug} \cup E_2^{aug})$ after removal of i and j. So, theses two subpaths can replace the edge (i, j) and after removal of edge (i, j), in graph $(V, E_1 \cup E_1^{aug} \cup E_2^{aug} - \{(i, j)\})$ if two adjacent node $[u, v]$ fails where $(u, v) \in E_1$ the graph still remains connected. Consequently, the graph $(V, E_1 \cup E^{aug})$ is $r \odot 2$ fault tolerant.

Theorem 3.5.6. *Algorithm* TCA_2 *finds a set of edges* E^{aug} *with total cost* $C^{aug} \leq 4C^{opt}$ *where* C^{opt} *is the cost of the optimal solution.*

Proof. Since $E^{aug} \subseteq E_1^{aug} \cup E_2^{aug}$ based on Theorems 3.5.2 and 3.5.4, $C^{aug} \leq 4C^{opt}$.

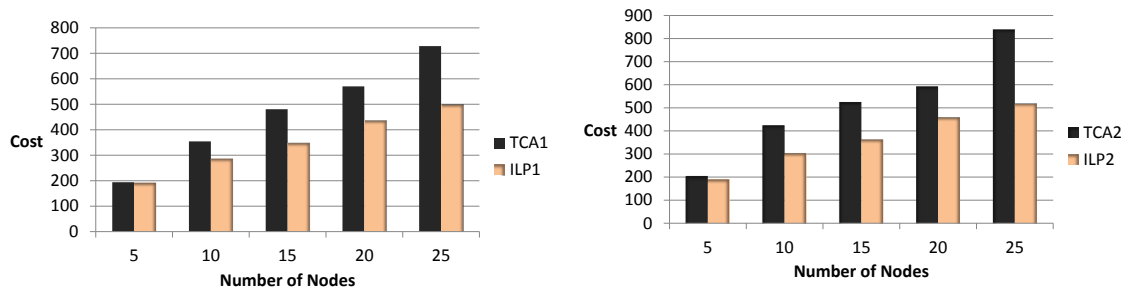
3.6 Experimental Results

In this section we present the experimental results of the approximation algorithms proposed for TCA_1 and TCA_2 problems. In the experiments we compare the results of the approximation algorithms against the optimal solution. Moreover, we examine energy consumption by directional and omni-directional antennas deployed in a sensor network.

For every instance of our experiment, we generate the locations of the sensor nodes randomly, using a uniform distribution on a square deployment area of size 100×100 units. We take the cost of edge between the nodes u and v in the sensor network, as an indicator of the transmit power needed by the nodes to reach each other. Accordingly, we construct a complete graph $G = (V, E)$ by setting the cost of each edge $c(i, j)$ proportional to $d^2(i, j)$ where $d(i, j)$ is the Euclidean distance between nodes i and j. We assume that an omni-directional antenna will consume power proportional to r^2 where r is the radius of the coverage circle. A directional antenna with same transmit range r but with transmit beam width α degrees will consume power proportional to $\frac{\alpha}{2\pi}r^2$ [29]. In our model, an edge represents two directional antennas transmitting signals to each other. Therefore, if the beam width is α we assume that the cost of the edge (i, j) is $\frac{\alpha}{\pi}d^2(i, j)$. For each problem instance we compute the minimum spanning tree to be the initial tree T_1 and select a node randomly as the root of the tree.

In our first set of experiments our objective is to compare the results of the approximation algorithms with the optimal solution, obtained by solving an integer linear programming (ILP). We denote the ILP used to find the optimal solution of TCA_1 and TCA_2 by ILP_1 and ILP_2, respectively. We used the CPLEX package to solve the ILP formulations. Since ILPs takes considerable amount of time in these experiments, we vary the number of nodes, n, from 5 to 25 in steps of 5. For each value of n, we generate 10 random instances of network layouts in a 2-dimensional plane. We set the beam width to 30 degrees. We present in Figures 3.6(a) and 3.6(b) the comparisons between the optimal augmentation costs and the augmentation costs computed by the TCA_1 and TCA_2 algorithms, respectively. For each n, we compute the average cost over 10 instances. We note that in these simulations, the ratio of the average cost of augmentation computed by the TCA_1 algorithm to the average cost

of optimal solution is smaller than 1.46. Also, the ratio of the average augmentation cost obtained by TCA_2 algorithm to the average optimal cost for each value of n is smaller than 1.62. These ratios are significantly better than the approximation factors for TCA_1 and TCA_2 algorithms that are 2 and 4, respectively. In all of these experiments TCA_1 and TCA_2 algorithms execute in a small fraction of the time required to achieve the optimal solution. For larger values of n, ILPs take so much time that we were unable to get the optimal solution in a reasonable amount of time while TCA_1 and TCA_2 can find an augmentation solutions with costs close to those of optimal solutions within a reasonable time.



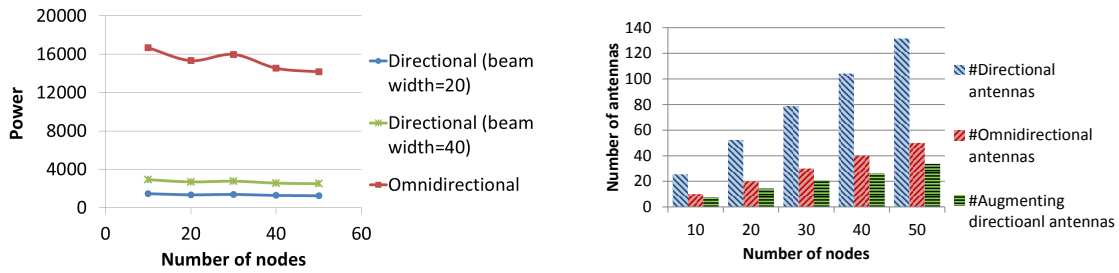
(a) (b)

Figure 3.6: (a) Comparison of augmentation cost of TCA_1 algorithm and ILP_1; (b) Comparison of augmentation cost of TCA_2 algorithm and ILP_2.

In our second set of experiments, we compare the power consumption of directional antennas versus omni-directional antennas. We vary the number of nodes, n, from 10 to 50 in steps of 10. For each value of n, we generate 50 random instances of network layouts in a 2-dimensional plane. We compute the average cost of augmentation by taking the average of costs incurred in these 50 instances. We perform the experiments for two values of beam width, 20 and 40 degrees. For each instance we execute the TCA_1 algorithm. We assume that the transmit range of the omni-directional antenna is large enough to cover furthest neighbor in the augmented graph. Therefore, the total power consumption in network with omni-directional antennas is proportional to $\sum_{1 \leq i \leq n} \max_{\{j|(i,j) \in E_1 \cup E^{aug}\}} d^2(i,j)$. In the case of directional

antennas, every edge in the augmented graph corresponds to two antennas, where each antenna needs $\frac{\alpha}{2\pi}d^2(i,j)$ amount of power to reach the other. Hence, the total power consumption is proportional to $\sum_{(i,j)\in E_1\cup E^{aug}} \frac{\alpha}{\pi}d^2(i,j)$. The Fig. 3.7(a) illustrates the comparison of power consumption in these two cases. We observe that the total power consumption in the network is significantly smaller when directional antennas are used instead of omni-directional ones. More specifically, when the beam width is 20 degrees, the power consumption with directional antennas is less than 9 percent of omni-directional antennas and when the beam width is 40 degrees it is less than 18 percent. We note that in order to make the network $r \odot 1$ fault tolerant, we need to install additional directional antennas, which has a fixed cost associated with it. When nodes have omni-directional antenna, each node requires only one antenna. When we use directional antennas, the total number of antennas deployed in every sensor node is equal to the node degree (including both the initial set of antennas in the tree and the antennas installed to augment the tree). In Fig. 3.7(b) we illustrate the average number of antennas that is needed in the network for each value of n. The first diagram shows the total number of directional antennas needed in the network. We observe that the ratio of the total number of directional antennas to the number of omni-directional antennas in these experiments is less than 2.7. However, in the TCA_1 problem we consider that the initial set of edges in the tree has cost zero (it is not part of augmentation cost). Therefore, only the cost of the new (augmenting) edges (i.e., the cost of the corresponding additional antennas) that are added during augmentation phase needs to be considered. The third diagram in Fig. 3.7(b) depicts the average number of directional antennas that are added to the network during the augmentation process. This number is smaller than 75 percent of the number of omni-directional antennas. More accurately, augmenting directional antennas that are needed to make the network $r \odot 1$ fault tolerant are fewer than 75 percent of number of omni-directional antennas. Therefore, the savings in power

consumption with directional antennas outweighs the cost of additional directional antennas needed, particularly when the width of the antenna beam is narrow.



(a) (b)

Figure 3.7: Comparison between (a) power consumption of directional antennas and omni-directional antennas, (b) number of directional antennas and omni-directional antennas

3.7 Conclusion

Motivated by the importance of both data collection and fault tolerance in wireless sensor networks, we studied the problem of enhancing the fault tolerance capability of a data gathering tree by adding a few additional links. We considered two fault models: 1) single node failure and 2) two adjacent node failure. We proved that the least cost tree augmentation problem is NP-complete under both types of fault scenarios. Moreover, we proposed two approximation algorithms, one for single node failure and the other for a pair of adjacent node failure, with performance bounds of two and four respectively. Experimental evaluation of the approximation algorithms shows that they perform even better in practice. In future we plan to study the tree augmentation problem under more general topological fault models like when a fault is defined as a subgraph with diameter d.

Chapter 4

ROUTING AND SPECTRUM ALLOCATION IN SPECTRUM SLICED OPTICAL NETWORKS

The phenomenal growth of the Internet traffic in the last few years and its anticipated growth in the next few years, necessitates introduction of innovative and efficient technology solutions in optical networks of the future. A recent Cisco study on growth of global IP traffic makes the following observations and forecasts [42].

- Global IP traffic has increased eightfold over the last five years, and will increase threefold over the next five years.
- In 2016, global IP traffic will reach 1.3 zettabytes per year or 109.5 exabytes per month.
- IP traffic will grow at a compound annual growth rate of 29 percent from 2011 to 2016.

In order to meet the challenges posed by the explosive size of the network traffic, optical networks must be operated in the most efficient manner. The traditional WDM network operates at the granularity of a wavelength, which may lead to inefficient use of resources as some connection requests may not have enough traffic to utilize the full capacity of a wavelength. However, such wastage of networking resources can be avoided if the optical network can be made to operate at a finer grain (i.e., sub-wavelength level) instead of the current practice of course grain operation (i.e., wavelength level). Recent introduction of Orthogonal Frequency Division Multiplexing (OFDM) technology in optical networks [43] offers an opportunity for operating optical networks at a much finer grain than what is currently possible. The advan-

tages offered by the OFDM in terms of flexibility and scalability originate from the unique multicarrier nature of this technology [43].

Utilizing the OFDM technology, a spectrum efficient and scalable optical transport network called *spectrum-sliced elastic optical path network (SLICE)* was proposed recently [44]. Just as the ability to operate at a granularity finer than a wavelength (i.e., a sub-wavelength) will enable the network operator to manage resources more efficiently, the same is true if the operator is provided with capability to operate at super-wavelength granularity. Such a capability will be useful for the network operator to meet large traffic demand. The goal of SLICE architecture is to allocate variable sized optical bandwidths that matches with the user traffic demands. It achieves that goal by *slicing off* spectral resources of a route and allocating only the requested amount to establish an end-to-end optical path.

Although the sub-wavelength (sub-carrier) level allocation capability of SLICE leads to more effective resource utilization, it also leads to additional complexities in network control and management. First if a call requests for d sub-carriers, the network controller must allocate d *consecutive* sub-carriers to this request. Second, if the paths corresponding to two requests R_1 and R_2 share a fiber link, not only the set of carriers allocated to R_1 and R_2 must be disjoint, in order to avoid interference, they must be *separated* from each other in the spectrum domain by a few carriers, known as *guard carriers* or *guard bands*. The first and the second constraints are known as the *sub-carrier consecutiveness constraint* and the *guard-carrier constraint* respectively [45]. The introduction of the *sub-carrier consecutiveness constraint* significantly increases the complexity of the Routing and Spectrum Assignment (RSA) problem that needs to be solved in SLICE. The RSA problem may be informally defined as follows: Given a network topology and a set of call requests with varying demands (in terms of the number of sub-carriers) find a route for each request and allocate a number of

sub-carriers to each request (equal to their requested demand), so that *the utilized part of the spectrum is minimized.* It may be noted that if the demand of each request is one sub-carrier, then the RSA problem reduces to the Routing and Wavelength Assignment (RWA) problem, which has been studied extensively. The RSA problem is significantly more complex than the RWA problem, as it is NP-complete even for a simple network topology, such as a chain. It may be noted that the RWA problem for the same topology can be solved with a low order polynomial time algorithm.

One can conceive of two different versions of the RSA problem - *off-line* and *on-line.* In the off-line version all the requests are known ahead of time before path and spectrum allocation for any request is carried out. In the on-line version, the requests come in a sequence and path and spectrum allocation for a request has to be carried out at the time of arrival of that request. Because the off-line version has the luxury of knowing all the requests, it can carry out better optimization of utilized spectrum span than its on-line counterpart. The performance of an on-line algorithm is measured in terms of the metric *competitive ratio.* In this metric, the performance of an on-line algorithm is compared with the performance of an optimal off-line algorithm that knows the sequence of requests in advance. The maximum ratio between their respective performances, taken over all sequences, is known as the *competitive ratio* of the algorithm [46].

In this chapter we study both off-line and on-line versions of RSA problem for networks with arbitrary structure as well as the networks with specific structure, such as a *tree* and a *ring.* Moreover, we introduce the *Spectrum Constrained Routing and Spectrum Assignment* (SCRSA) problem. The goal of the RSA problem is to satisfy all the requests in a way that maximizes the unutilized part of the spectrum [47]. In other words, the goal of the RSA problem is to minimize the *spectrum span.* However, the smallest spectrum span for a certain set of requests may be greater

than the *available spectrum span*. In this case, not all the requests can be satisfied. The goal of the SCRSA problem is to satisfy the largest number of requests without exceeding the available spectrum span. To the best of our knowledge, the SCRSA problem has not been studied earlier. The contributions in this research are as follow,

- We prove that the RSA problem is NP-complete when the network topology is a *chain* or a *ring*.
- We provide approximation algorithms for the off-line RSA problem with a performance bound of $O(log\ k)$ in a binary tree where k is the number of requests, and $4+2\epsilon$ in a ring.
- We provide an algorithm for the on-line version of the RSA problem for the ring network with a competitive ratio of $\min\{O(\log(d_{max})), O(\log(k))\}$ where k is the total number of requests, $d_{max} = \max_{1\leq i\leq k} d_i$, and d_i is the demand in terms of the number of sub-carriers associated with request R_i.
- We provide heuristics for off-line and on-line RSA in networks with arbitrary topology and measure the effectiveness of the heuristics with extensive simulation.
- We introduce the SCRSA problem and provide approximation algorithms for the problem with a performance bound of $O(log\ S)$ for a binary tree, where S is the available spectrum and $(4 + 2/(e-1))^{-1} - \epsilon$ for a ring.

The rest of this chapter is organized as follows. We discuss related works in Section 4.1. In Sections 4.2 and 4.4 we introduce definitions and notations and present a few preliminary observations respectively. We present formal problem statements for both the RSA and the SCRSA problem in Section 4.3. Analytical results for chains, trees and rings for the RSA and the SCRSA are presented in Sections 4.5 and

4.6 respectively. Experimental results for the RSA problem with arbitrary network topology is presented in Section 4.7. The section 4.8 concludes the chapter.

4.1 Related Work

Utilizing the optical OFDM technology, the SLICE architecture proposes a novel scheme for slicing off the spectral resources of a route, resulting in more efficient utilization [44]. The fact that the sub-carriers in the SLICE architecture has to be assigned in a contiguous manner, led to the formulation of the RSA problem. To the best of our knowledge, the RSA problem was originally introduced in [48, 49, 47]. Since then a few other papers, [45, 50] have also studied the RSA problem and proposed solution techniques. In most of these studies [45, 47, 50, 51], the authors propose an integer linear program based solution and a heuristic solution. Based on the experimental results, the authors claim effectiveness of their heuristics.

The on-line version of RSA problem has been studied in [49, 52, 53, 54, 55, 56]. In all of these papers, the objective of the on-line RSA problem is to maximize the number of requests that can be satisfied and minimize the blocking probability. In this version of on-line RSA problem, the number of available spectrum sub-carriers is limited. The authors of these papers proposed heuristic solutions mainly by modifying the Dijkstra shortest path algorithm or using K-shortest path algorithm accompanying with the First-Fit algorithm. To the best of our knowledge none of these papers consider the objective of minimizing the utilized spectrum while satisfying all the requests. It may be the case that all the requests should be satisfied while the utilized spectrum is minimized. In this research we propose a new heuristic for arbitrary network graphs. We also modify the K-shortest path approach for this version of the on-line RSA and through simulations we evaluate their performance.

Most of the studies both on on-line and off-line RSA do not present any analytical results for the RSA problem, even for the simplest optical network topologies

such as trees and rings. The ring topology is of particular importance in the optical domain because of its application in metro networks and in some long haul networks. A major thrust of our effort is to present analytical results for the RSA and SCRSA for optical networks with special structures, such as binary trees and rings.

It is well known that the RWA problem is equivalent to the computation of the chromatic number of the *path intersection graph* (all definitions are provided in section III) of the best routes for the requests. The RSA problem is a generalized version of the RWA problem, where each request is associated with a weight representing the demand of that request in terms of the number of sub-carriers. An implication of this generalization is that the RSA problem is no longer equivalent to computation of the *maximum chromatic number* of the path intersection graph of the best routes for the requests. In section IV, we show that the RSA problem is equivalent to computation of the *interval chromatic number* (ICN) of the path intersection graph of the best routes for the requests. The computation of ICN of an interval graph (a subclass of the chordal graphs) is proven to be NP-complete [57]. In [58], the authors have proposed an approximation algorithm with $2 + \epsilon$ performance bound for computation of ICN of interval graphs. In [59], the authors have proposed approximation algorithms with $O(\log n)$ performance bound for computation of ICN of chordal graphs.

In this dissertation we introduce the SCRSA problem. A special case of this problem, Wavelength Constrained RWA (WCRWA) problem for ring networks was studied in [60, 61]. The objective of the WCRWA problem is to satisfy as many call requests as possible, subject to the constraint that the number of wavelengths necessary to satisfy the requests do not exceed the number of available wavelengths. To the best of our knowledge, SCRSA problem has not been studied before even for the networks with ring or tree structures.

4.2 Definitions and Notations

Spectrum Slice/Interval: A number of consecutive sub-carriers from a_i to b_i denoted by $[a_i, b_i]$, that is allocated to a specific request R_i to establish a connection between (s_i, t_i) with d_i sub-carriers. The length of this spectrum slice/interval is $b_i - a_i + 1 = d_i$.

Spectrum Span/Spread: The total amount of spectrum used for allocating a slice to all the requests; If $R_i, 1 \leq i \leq k$ is allocated the spectrum interval $[a_i, b_i]$ then the spectrum span is $[\min_{1 \leq i \leq k} a_i, \max_{1 \leq i \leq k} b_i]$.

Stable Set: A stable set (also known as independent set) of a graph $G = (V, E)$ is a subset $V' \subseteq V$, such that no two nodes in V' have an edge between them in $G = (V, E)$.

Clique Number/Weighted Clique Number: The Clique Number $\omega(G)$ of a graph $G = (V, E)$ is the size of the largest complete subgraph of $G = (V, E)$ [62]. A weighted graph G^* is a triple $G^* = (V, E, w)$, where w is a positive integer valued function on the vertex set V. The weight $w(V')$ of a subset of vertices $V' \subseteq V$ is the sum of the weights of the vertices in V'. The *weighted clique number* of G^*, denoted by $\omega^*(G^*)$ is the maximum weight of a clique in G^*.

Maximum Chromatic Number: Consider a weighted graph $G^* = (V, E, w)$ with a strictly positive integer weight $w(v)$ associated with each node $v \in V$. The weight of a stable set S of G^* is defined as $w(S) = \max_{v \in S} w(v)$. The maximum chromatic number, $\chi_{max}(G^*)$, of G^* is the smallest value of $\sum_{i=1}^{t} w(S_i)$, where each S_i is a stable set of G^* such that $\cup_{i=1}^{t} S_i = V$ and for every pair of stable sets S_p and S_q, $S_p \cap S_q = \emptyset$ [59]. When $w(v) = 1$ for all vertices $v \in V$, then maximum chromatic number is called *chromatic number*.

Interval Chromatic Number (ICN): An *interval t-coloring* of a weighted graph $G^* = (V, E, w)$ is a function c from V to $\{1,2, \ldots, t\}$ such that $c(x) + w(x) - 1 \leq t$ and if both $c(x) \leq c(y)$ and $(x, y) \in E$ then $c(x) + w(x) - 1 < c(y)$. We can view an interval coloring c of G^* as assigning an interval $[c(v), \ldots, c(v) + w(v) - 1]$ of $w(v)$ colors to each vertex v so that the intervals of colors assigned to two adjacent vertices (i.e., the pair of nodes that has an edge between them) do not overlap. If interval t-coloring is feasible for a graph G^* then G^* is said to be *interval t-colorable*. The interval chromatic number of G^*, denoted by $\chi_{int}(G^*)$ is the *least* t such that G^* has a interval t-coloring [63].

Interval Graph: Let $\mathcal{F}$ be a family of non-empty sets. The *intersection graph* of $\mathcal{F}$ is obtained by representing each set in $\mathcal{F}$ by a node and connecting the two nods with an edge, if and only if the corresponding sets intersect. The intersection graph of a family of intervals on a linearly ordered set (such as the real line) is called *Interval Graph* [62].

Chordal Graph: An undirected graph $G = (V, E)$ is called a Chordal Graph if every cycle of length strictly greater than 3 has a *chord*, that is, an edge joining two nonconsecutive nodes in the cycle [62]

Path Intersection Graph: Consider a graph $G = (V, E)$ and a set of paths $\mathcal{P} = \{P_1, \ldots, P_k\}$, where each P_i is a path between a node pair (s_i, t_i), $\forall i, 1 \leq i \leq k$. A graph $G' = (V', E')$ is a Path Intersection Graph corresponding to $\mathcal{P}$, if each vertex $p_i \in V'$ corresponds to a path $P_i \in \mathcal{P}$ and two nodes p_i and p_j in V' have an edge between them, if the corresponding paths P_i and P_j in $\mathcal{P}$ have at least one common edge in E.

4.3 Problem Statement

In this section we provide a formal statement of the RSA and SCRSA problems. We formulate the RSA problem as it is defined in [47, 50].

Routing and Spectrum Assignment (RSA) Problem: Given a graph $G = (V, E)$ representing the network topology, and a set of request triples $R_i = (s_i, t_i, d_i), 1 \leq i \leq k$, where s_i represents a source node, t_i represents a destination node, and d_i represents the demand between s_i and t_i in terms of sub-carriers, find (i) a set of paths $\mathcal{P} = \{P_1, \ldots, P_k\}$, such that each $P_i, 1 \leq i \leq k$ is a path from s_i to t_i and (ii) assign an interval $I_i = [a_i, b_i]$ (*spectrum slice*) of length d_i to each P_i, such that all the intervals $I_i, 1 \leq i \leq k$ can be fitted within a smallest interval $\mathcal{I} = [\min_{1 \leq i \leq k} a_i, \max_{1 \leq i \leq k} b_i]$ (*spectrum span*) such that the intervals I_i and I_j do not overlap if the corresponding paths P_i and P_j share an edge between them in $G = (V, E)$. Moreover, if the paths P_i and P_j overlap, not only the corresponding intervals I_i and I_j must be non-overlapping, these two intervals must be separated by a fixed number of sub-carriers, known as the *guard band*. Without loss of generality, we number the first available sub-carrier one and the rest are numbered sequentially.

In *On-line RSA Problem* the connection requests arrive in a sequence one by one where k is the total number of requests. Once a request R_i arrives without knowledge of the future requests, a path P_i from s_i to t_i should be assigned to R_i and a spectrum interval $I_i = [a_i, b_i]$ of length d_i should be assigned to P_i such that at all times the constraints in RSA problem is satisfied and the objective is to minimize the spectrum span.

Hereafter by RSA we mean off-line RSA unless we mention otherwise.

Spectrum Constrained RSA (SCRSA) Problem: The RSA problem computes the smallest spectrum span needed to satisfy all k call requests. The input to the SCRSA

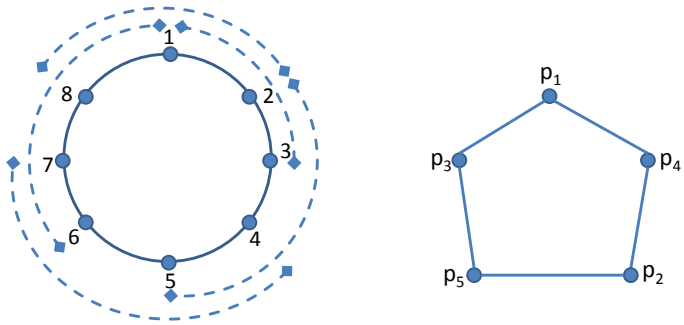
problem is comprised of all the parameters of the RSA problem and one additional one. The additional parameter S indicates the spectrum span available to satisfy the call requests. Because of the constraint on the available spectrum span S, not all k call requests may be satisfied. The goal of the SCRSA problem is to find the largest subset of call requests that can be satisfied with available spectrum span S.

4.4 Preliminary Observations

The RSA problem has two distinct components - the routing component and the spectrum allocation component. The solution to the RSA problem will be sub-optimal if (i) sub-optimal routes are chosen, and/or (ii) sub-optimal spectrum allocation is carried out. It is tempting to think that the complexity of the RSA problem arises due to the fact that it has to deal with two interdependent subproblems at the same time. However, this is not necessarily true. In certain types of network topology (e.g., a tree), there exists only one path that connects the source node s_i to the destination node t_i for all $i, 1 \leq i \leq k$. This implies that routing is a trivial problem in these networks. However, even in these networks, spectrum allocation (SA) is not only non-trivial, it is computationally hard (i.e., NP-complete). In the following, we make a few observations. In the first three observations, we assume there is no guard carrier constraint, i.e. size of the guard band is zero. Then, in the last observation we explain how we deal with guard carrier constraint.

Observation 1: Weighted Clique Number establishes a *lower bound* for the SA problem.

In the SA problem a set of paths $\mathcal{P}$ on graph G is associated to the set of requests $\{R_i, 1 \leq i \leq k\}$. Let $G' = (V', E', w)$ be the weighted path intersection graph of paths in $\mathcal{P}$ where $V' = \{p_1, p_2, \ldots, p_k\}$ where each node p_i corresponds to a path $P_i \in \mathcal{P}$ and the weight of p_i is d_i; i.e., $w(p_i) = d_i$. For any clique Q in G' every vertex p_i in Q needs to have $w(p_i)$ distinct colors and therefore $\sum_{p_i \in Q} w(p_i)$ is a lower bound



(a) (b)

Figure 4.1: (a) An example of SA instance where the network graph G is a ring (b) Path intersection graph G' of paths in SA instance in (a)

on the optimal spectrum span. Fig. 4.1 shows an example of SA instance where the network graph is a ring with 8 nodes and requests are $\{R_1 = (1, 3, 15), R_2 = (1, 6, 6), R_3 = (2, 5, 6), R_4 = (2, 8, 6), R_5 = (4, 7, 12)\}$. Dashed lines show the paths for the requests. Fig. 4.1(b) depicts G', the path intersection graph of these paths where $w(p_1) = 15, w(p_2) = 6, w(p_3) = 6, w(p_4) = 6$ and $w(p_5) = 12$. It can be observed that the largest clique in G' is $\{p_1, p_3\}$ and weighted clique number of G' is 21.

Observation 2: Maximum Chromatic Number establishes an *upper bound* for the SA problem.

In the computation of $\chi_{max}(G')$, the vertices in V' are partitioned into stable sets. Since the paths corresponding to the vertices in a stable set S are edge disjoint, they can share the spectrum. Therefore, $w(S) = \max_{v \in S} w(v)$ consecutive sub-carriers is enough for allocation to the paths in S. Accordingly, the span with size of $\chi_{max}(G')$ is sufficient for the spectrum allocation of the requests in G. In Fig. 4.1, it can be observed that $\chi_{max}(G')$ is 27 and the stable sets are $\{p_1, p_5\}, \{p_2, p_3\}, \{p_4\}$. The requests corresponding to the paths in first, second and third stable sets are allocated the spectrum interval of $[1, 15]$, $[16, 21]$ and $[22, 27]$ respectively. In the next observation we show that the span of size $\chi_{max}(G')$ may not be necessary.

Observation 3: Interval Chromatic Number (ICN) finds the solution to the SA problem (i.e., it establishes both the upper and the lower bound for the SA problem). Let $\chi_{int}(G')$ be the ICN of graph G'. In computation of $\chi_{int}(G')$, each node $p_i \in V'$ is assigned an interval $[a_i, b_i]$ of colors with length $w(p_i) = d_i$ where the intervals of two adjacent vertices do not intersect and total number of distinct colors used is minimum. Therefore, interval $[a_i, b_i]$ can be allocated to the path P_i in G and no two paths with common edge intersect in their spectrum intervals. Hence, the spectrum span of $\chi_{int}(G')$ is sufficient for the spectrum allocation of requests in G with predefined set of paths $\mathcal{P}$. Moreover, $\chi_{int}(G')$ is the minimum spectrum span needed in the SA problem; otherwise, it contradicts with $\chi_{int}(G')$ being the minimum interval chromatic number of G'.

In the example depicted in Fig. 4.1, $\chi_{int}(G')$ is 24 where the requests R_1 to R_5 are assigned intervals $[1, 15]$, $[13, 18]$, $[16, 21]$, $[19, 24]$ and $[1, 12]$ respectively.

Observation 4: An algorithm that finds the optimal solution for the RSA problem when the guard band $g = 0$, can be utilized to find the optimal solution for the RSA problem when $g > 0$.

Proof: Let RSA_1 denote an instance of RSA problem where the size of guard band, g is $g > 0$ and OPT_1 denote the minimum spectrum span needed in RSA_1. We define another RSA instance, RSA_2, with the same graph G as in RSA_1. For every request $R_i = (s_i, t_i, d_i)$ in RSA_1 we add a request $R'_i = (s_i, t_i, d_i + g)$ to RSA_2 and size of guard band in RSA_2 is zero. Let OPT_2 be the minimum spectrum span of RSA_2. Clearly, any feasible solution of RSA_2 is a feasible solution of RSA_1 and $OPT_2 \geq OPT_1$. We show that $OPT_2 \leq OPT_1 + g$. Of this concern, we construct a feasible solution for RSA_2 from the optimal solution of RSA_1.

Let, $\mathcal{P}_1 = \{P_1, P_2, \ldots, P_k\}$ and $I = \{I_1 = [a_1, b_1], I_2 = [a_2, b_2], \ldots, I_k = [a_k, b_k]\}$ be the set of paths and the allocated intervals in the optimal solution of RSA_1 respectively, after sorting them based on the start points of intervals; i.e. $a_1 \leq a_2 \leq \ldots \leq a_k$. We construct a solution of RSA_2 as follows. We consider the same set of paths as $\mathcal{P}_1$. In the spectrum allocation part, we assign interval $I'_i = [a_i, b_i + g]$ to every request $R'_i, 1 \leq i \leq k$. We need to show that $\{I'_1, I'_2, \ldots, I'_k\}$ is a feasible solution of SA part. Using induction on the set of requests we can prove the feasibility of this solution as follows. Clearly, $I'_1 = [a_1, b_1 + g]$ is a feasible solution when the set of requests includes just R'_1 (induction base). We assume that for an integer $q, 1 \leq q \leq k$, $\{I'_1, I'_2, \ldots, I'_q\}$ is a feasible solution of SA problem on the set of requests $\{R'_1, R'_2, \ldots, R'_q\}$. We prove that $\{I'_1, I'_2, \ldots, I'_{q+1}\}$ where $I'_{q+1} = [a_{q+1}, b_{q+1} + g]$ is a feasible solution of SA problem on the set of requests $\{R'_1, R'_2, \ldots, R'_{q+1}\}$. In order to prove this, it is enough to verify that there is no request $R'_i, 1 \leq i \leq q$ where P_i intersects with P_{q+1} and I'_i intersects with I'_{q+1}. Let R'_j be a request where P_j intersects with P_{q+1}. Based on the optimal solution of RSA_1 we know that $a_j \leq a_{q+1}$, $I_j \cap I_{q+1} = \emptyset$, and I_j and I_{q+1} are separated with guard band g. Hence, $a_{q+1} > b_j + g$. Consequently, $I'_{q+1} = [a_{q+1}, b_{q+1} + g]$ does not have intersection with $I'_j = [a_j, b_j + g]$. Therefore, $I' = \{I'_1, I'_2, \ldots, I'_k\}$ is a feasible solution of SA part in RSA_2 instance. Clearly, spectrum span of intervals in I' uses at most g sub-carriers more than intervals in I. Therefore, the optimal solution of RSA_2, can be used to construct the solution of RSA_1 by replacing each interval $I'_i = [a_i, b_i + g]$ by $I_i = [a_i, b_i]$.

Similarly, it can be shown that if an algorithm finds the optimal solution for the SCRSA problem when the guard band $g = 0$, it can be utilized to find the optimal solution for the SCRSA problem when $g > 0$.

Accordingly, for the rest of our analysis we assume $g = 0$.

Theorem 4.5.1. *RSA problem is NP-Complete when the optical network topology is a chain.*

Proof: Stockmeyer showed that the computation of the ICN of an Interval graph (ICNIG) is an NP-complete problem (problem SR2 in [57]). We prove that the RSA problem is NP-Complete when the optical network topology is a chain by giving a transformation form the ICNIG problem.

ICNIG Problem:

Instance: Given the interval representation of an interval graph $G = (V, E)$ (i.e., a set of intervals $\{I_1, \ldots, I_n\}$), and a set of positive integer weights w_i associated with $I_i, 1 \leq i \leq n$, and an integer B.

Question: Is the ICN of $G = (V, E)$ at most B?

From an instance of the ICNIG, we will create an instance of the RSA-P, we will prove that the ICN of the instance of RSA-P will be at most B iff the ICN of the instance of ICNIG is at most B. Suppose that the left and the right end point of the interval $I_i, 1 \leq i \leq n$ is denoted by a_i and b_i. Then project the end points a_i and $b_i, 1 \leq i \leq n$ on a real line $\mathcal{R}$. We will refer to the points where projection of a_i meets $\mathcal{R}$ as a'_i. Similarly, the projection points corresponding to b_i will be referred to as b'_i. If some endpoints are the same, we consider just one point on $\mathcal{R}$ that represents all of them. It may be noted that on $\mathcal{R}$ each point a'_i and b'_i has two neighboring points, one to the left and one to the right (except the leftmost and the rightmost points). We will now construct a graph with nodes corresponding to the distinct points a'_i and b'_i, $1 \leq i \leq n$ denoted by $v(a'_i)$, $v(b'_i)$ respectively. Each node $v(a'_i)$ (or $v(b'_i)$) will have

an edge to the node corresponding to the left neighbor of the point a'_i (b'_i) and an edge to the node corresponding to the right neighbor of the point a'_i (b'_i). It can be easily verified that the constructed graph is a chain. Next we will create the requests $R_i = (v(a'_i), v(b'_i), w_i), 1 \leq i \leq n$. Construction of the instance of the RSA-P from an instance of the ICN is now complete. It can easily be verified that the ICN of the instance of RSA-P will be at most B iff the ICN of the instance of ICNIG is at most B.

Theorem 4.5.2. *There exists an approximation algorithm with a performance bound of $2 + \epsilon$ for RSA problem when $G = (V, E)$ is a chain.*

Proof: When the graph $G = (V, E)$ is a chain, then there exists only one path for each request and routing is trivial. Let $\mathcal{P} = \{P_1, \ldots, P_k\}$ be the set of paths of requests in G. Clearly, intersection graph of paths in $\mathcal{P}$, $G' = (V', E')$ is an interval graph (each path P_i can be represented as an interval from s_i to t_i). Therefore, RSA problem in a chain will be equivalent to the computation of ICN of interval graph G'. In [58], an approximation algorithm with performance bound of $2+\epsilon$ for computation of ICN of interval graphs is proposed. Therefore, the same algorithm will have an approximation ratio of $2 + \epsilon$ for RSA problem when graph G is a chain.

4.5.2 RSA Problem in Trees

Theorem 4.5.3. *RSA problem is NP-Complete when the optical network topology is a Binary Tree.*

Proof: Since chain is a special case of binary tree, the proof follows proof of Theorem 4.5.1

Theorem 4.5.4. *There exists an approximation algorithm with a performance bound of $O(\log k)$ for RSA problem when $G = (V, E)$ is a Binary Tree.*

Proof: When network topology is a tree, then there exists only one path for each request and routing is trivial. Let $\mathcal{P}$ be the set of paths of requests. It has been shown in [64] that the intersection graph of paths in a binary tree is a *chordal graph*. Hence, RSA problem when network is a binary tree is equivalent to the computation of ICN of path intersection graph which is a chordal graph. In [59] an approximation algorithm with performance bound of $O(\log k)$ is proposed for computation of ICN in chordal graphs when k is number of nodes. Therefore, the same algorithm will find a solution for RSA with approximation ratio of $\log k$ where k is the number of requests (number of nodes in path intersection graph).

4.5.3 RSA Problem in Rings

4.5.3.1 Off-line

Theorem 4.5.5. *RSA problem is NP-Complete when the optical network topology $G = (V, E)$ is a Ring.*

Proof: If the demands of the requests in the RSA instance are all equal to one, then RSA problem becomes RWA problem. In [65], it is proven that the RWA problem for optical networks with a ring topology is NP-complete. Since RWA problem is a special case of the RSA problem, it follows that the RSA problem for optical networks with a ring topology is also NP-complete.

Next, we propose an approximation algorithm called RSA-R, for RSA problem when network topology is a ring. In RSA-R, we use cut-one-link approach and take advantage of the approximation algorithm for computation of ICN in interval graphs proposed in [58]. The steps of RSA-R are explained in Algorithm 7.

Theorem 4.5.6. *Algorithm 7 has performance bound of $4 + 2\epsilon$.*

Proof: After removing one edge randomly from $G = (V, E)$, the induced graph

Algorithm 7 RSA-R

1: Remove an edge $e \in E$ randomly; Let G_p be the induced chain;
2: Compute the set of paths $\mathcal{P}'$ for the requests in graph G_p;
3: Compute and return the approximated ICN of intersection graph of paths in $\mathcal{P}'$ using algorithm in [58];

G_p is a chain. Let OPT and OPT_p be the optimal spectrum span in RSA problem when network graph is G and G_p, respectively, and $\mathcal{I}$ be the size of the spectrum computed by Algorithm 7. Based on Theorem 4.5.2 we have (1) $\mathcal{I} \leq (2+\epsilon)OPT_p$. We denote the set of paths in the optimal solution of RSA when network graph is G by $\mathcal{P}_{OPT}$. The paths in $\mathcal{P}_{OPT}$ can be partitioned into two subsets, $\mathcal{P}_e^1$ and $\mathcal{P}_e^2$ such that $\mathcal{P}_e^1$ is the set of paths that include edge e and the paths in $\mathcal{P}_e^2$ do not include edge e. Let OPT_e^1 and OPT_e^2 be the ICN of the intersection graph of paths in $\mathcal{P}_e^1$ and $\mathcal{P}_e^2$ respectively. Then we have (2) $OPT \geq \max(OPT_e^1, OPT_e^2)$. Since all the paths in $\mathcal{P}_e^1$ have intersection in edge e, their intervals do not intersect. Clearly, (3) $OPT_p \leq OPT_e^1 + OPT_e^2$. The reason is that in the worst case, all requests that were routed through edge e in $\mathcal{P}_{OPT}$ are routed the other way in G_p and now they at most need OPT_e^1 spectrum span not intersecting the spectrum allocated to the paths in $\mathcal{P}_e^2$. Therefore, using relations in (2) and (3) we have $OPT_p \leq 2OPT$. Also, based on relation (1) we can conclude $\mathcal{I} \leq (4+2\epsilon)OPT$.

4.5.3.2 On-line

In this part, we propose an on-line algorithm for RSA problem when network topology is a ring. In this algorithm, Similar to off-line scenario first we use cut-one-link approach and after removing one link the induced graph is a chain. In the chain for every request there exists just one path. Therefore routing is trivial. For the spectrum assignment, we use First-Fit technique that finds the first free spectrum interval fit the demand of the current request. The steps of the algorithms are explained in Algorithm 8.

Algorithm 8 On-line RSA in Ring

1: Remove an edge $e \in E$ randomly; Let G_p be the induced chain;
2: **while** A new request arrives **do**
3: Find the path for the request in graph G_p;
4: Compute the first free spectrum interval fit the demand of the current request
5: **end while**

Theorem 4.5.7. *Algorithm 8 has competitive ratio of* $\min\{O(\log(d_{max})), O(\log(k))\}$ *where k is total number of requests and* $d_{max} = \max_{1 \leq i \leq k} d_i$.

Proof: In order to compute the competitive ratio we need to compare the spectrum span of Algorithm 8 with the optimal spectrum span of the off-line RSA. Let OPT and OPT_p be the optimal spectrum span in RSA problem when network graph is G and G_p, respectively, and $\mathcal{I}_O$ be the size of the spectrum computed by Algorithm 8. Let G'_p be the path intersection graph. Clearly, minimum spectrum needed to satisfy requests in G_p is equivalent to the $\chi_{int}(G'_p)$. Based on the paper [46], First-Fit algorithm will have competitive ratio of $\min\{O(\log(d_{max})), O(\log(\chi_{G'_p}))\}$ for on-line interval coloring in G'_p. Also it is obvious $\chi_{G'_p} \leq k$ (i.e., chromatic number of G'_p is at most as large as the number of nodes in G'_p that is number of requests). Hence, we have (1) $\mathcal{I}_O \leq \min\{O(\log(d_{max})), O(\log(k))\} \cdot OPT_p$. Based on the proof of Theorem 4.5.6, we have $OPT_p \leq 2OPT$. Also, based on relation (1) we can conclude $\mathcal{I}_O \leq \min\{O(\log(d_{max})), O(\log(k))\} \cdot OPT$.

4.5.4 RSA Problem in General Graphs

Since RSA problem is NP-Complete, computation of the optimal solution in a reasonable amount of time may not be possible unless P=NP. As such we need to find heuristic solutions for both versions off-line and on-line RSA problem. In this section we propose two heuristics called DPH and $MSCP$ For off-line and on-line RSA problem in general networks, respectively. We evaluate the performance of the heuristics in section 4.7 using simulations on some real optical networks. The main idea in our

heuristics is that they try to find disjoint paths for routing the requests to increase the reuse of sub-carriers in spectrum allocation. In next parts we explain the heuristics in detail.

4.5.4.1 Off-line

In DPH first the requests are sorted in decreasing order of their demands. In each iteration, a set of disjoint paths is found. Then, spectrum allocation of these requests is computed. Since the paths found in one iteration are disjoint their spectrum interval can intersect. Also, DPH checks if any of the paths can reuse sub-carriers used in previous iterations trying to minimize the spectrum span. The details of DPH is explained in Algorithm 9. It can easily been verified that time complexity of DPH is $O(k^2|V|^2)$.

4.5.4.2 On-line

We develop a heuristic for on-line RSA called *Minimum Sub-Carrier Path Heuristic (MSCP)*. We define a new weight function on the edges (fibers) of the network where weight of an edge $e \in E$, $w(e)$ will be largest sub-carrier number that is used in that edge. We also define the weight of a path, P from node s to node t to be $\max_{e \in P}\{w(e)\}$. For each new request, $MSCP$ selects the path with minimum weight. The minimum weight path can be computed by modifying the distance function in Dijkstra algorithm so that it considers the new weight function as the distance. After finding the path, $MSCP$ uses First-Fit algorithm to find the first available spectrum slice with the length of the request demand in all the edges of the path. Then, $MSCP$ updates the weight of every edge in the path to the largest sub-carrier so far used in that edge.

Algorithm 9 $DPH-$ Heuristic for RSA in general graphs

```
Sort the requests in decreasing order of the demands; Let R = {R_1, R_2, ... R_k} be the sorted set.
Define I to be a set of intervals, I_i = [a_i, b_i], 1 ≤ i ≤ k allocated to R_i s where the intervals in I are sorted in decreasing order of b_i s. Initially, I includes an interval I_0 = [1, 1] corresponding to an empty path P_0.
while there is a request that is not allocated a path and spectrum interval do
    D = ∅; D is a set of disjoint paths sorted in decreasing order of demands.
    P_f = shortest path between s_f and t_f in G where R_f is the first request in R not allocated spectrum interval;
    Add P_f to D;
    G_T = (V, E_T) where E_T = E − {e|e ∈ P_f};
    for all R_i, f ≤ i ≤ k that is not allocated a path do
        if There exists a path between s_i and t_i in G_T then
            P_i = the shortest path between s_i to t_i in G_T;
            Add P_i to D; and G_T = (V, E_T − {e|e ∈ P_i});
        end if
    end for
    for all Paths P_i ∈ D do
        for all Intervals I_j ∈ I do
            if paths P_i and P_j do not intersect in any edge and I_j is the last interval in I then
                a_i = a_j;
            else if paths P_i and P_j do not intersect in any edge and I_l is the interval right after I_j in I then
                a_i = b_l + 1;
            else
                a_i = b_j + 1; break;
            end if
        end for
        I_i = [a_i, a_i + d_i − 1]; Add I_i to I such that I remains sorted in decreasing order of b_i.
    end for
end while
```

Algorithm 10 $MSCP-$ Heuristic for On-line RSA in general graphs

```
Initially, w(e) = 0, ∀e ∈ E
while A new request R_i comes do
    Compute P_i = minimum-weight path between s_i and t_i in G;
    Find the first available spectrum slice [a_i, b_i] with the length of d_i using First-Fit
    algorithm
    for all Edges e ∈ P_i do
        w(e) = max(w(e), b_i)
    end for
end while
```

For each request, time complexity of minimum-weight path computation is $O(|V|^2)$ and First-Fit algorithm takes $O(k|V|^2)$ where k is the number of requests. Hence, time complexity of $MSCP$ is $O(k^2|V|^2)$.

4.6 Spectrum Constrained RSA Problem

In this section we study SCRSA problem when optical network is a binary tree or a ring.

Theorem 4.6.1. *SCRSA problem is NP-Complete when the optical network topology is a chain, binary tree or a ring.*

Proof: We prove that SCRSA problem is NP-Complete by giving a transformation form the RSA problem. For a given graph $G = (V, E)$, a set of requests and a value of S, the decision version of RSA problem is that whether all requests can be routed in G with at most S spectrum span. In the instance of SCRSA problem we consider the same graph G and same set of requests and spectrum span of S and a number q and the decision version of SCRSA problem is that whether at least q number of requests can be routed in G using at most spectrum span of size S. In the instance of SCRSA, if we set q to be the number of all requests, obviously, the RSA problem is reduced to the SCRSA problem. Therefore, SCRSA is also NP-complete in chains, binary trees and rings.

4.6.1 SCRSA in Chains

Theorem 4.6.2. *There exists an approximation algorithm with a performance bound of* $(2 + 1/(e-1))^{-1} - \epsilon \approx 1/2.58$ *for SCRSA problem when* $G = (V, E)$ *is a chain.*

Proof: When the graph $G = (V, E)$ is a chain, then there exists only one path for each request and routing is trivial. Let $\mathcal{P} = \{P_1, \ldots, P_k\}$ be the set of paths of requests in G. Clearly, intersection graph of paths in $\mathcal{P}$, $G' = (V', E')$ is an interval graph (each path P_i can be represented as an interval from s_i to t_i). Therefore, SCRSA problem in a chain will be equivalent to the computation of largest subgraph (in terms of number of vertices) of interval graph G' where the subgraph is *interval S-colorable*. The authors of [66] have studied the problem of SAP (Storage Allocation Problem). In this problem a set of requests for memory is given where each request includes start time, finish time, demand (contiguous memory blocks) and profit. Total memory is given as $\mathcal{S}$. The objective is to select a subset of requests where total demands do not exceed $\mathcal{S}$, and no two requests can use the same block at the same time, and total profit is maximized. The authors propose an approximation algorithm with a performance bound of $(2 + 1/(e-1))^{-1} - \epsilon \approx 1/2.58$. It can be seen that when the profits are the same, SAP problem is actually equivalent to the problem of SCRSA in chains, i.e. finding the largest subgraph of an interval graph (intervals are given) such that the subgraph is interval $\mathcal{S}$-colorable. Hence the same approximation algorithm for SAP problem can be used for SCRSA problem in chains with a performance bound of $(2 + 1/(e-1))^{-1} - \epsilon \approx 1/2.58$.

4.6.2 SCRSA in Binary Tree

Now, we propose an algorithm denoted by SCRSA-B for SCRSA problem when network is a binary tree and the available spectrum S is fixed. Since the network topology is a binary tree the routing is trivial and $\mathcal{P}$ is determined. Therefore, we need to

compute the largest subgraph of the intersection graph of paths in $\mathcal{P}$ that is interval S-colorable. It has been shown in [64] that the intersection graph of paths in a binary tree is a chordal graph. When the demand of all requests are equal to one, SCRSA problem in a binary tree is equivalent to the problem of finding maximum S-colorable subgraph in the path intersection graph which is chordal. In [67], a polynomial algorithm has been proposed that solves the maximum S-colorable subgraph problem in chordal graphs for fixed S optimally. We denote this algorithm by UDA. We use UDA in SCRSA-B to solve SCRSA problem. Before explaining SCRSA-B, first we prove a lemma.

Lemma 1. *An SCRSA instance with the set of paths $\mathcal{P}$ and an integer x are given, where every request $R_i, 1 \leq i \leq k$ has a demand size $2^{x-1} < d_i \leq 2^x$. If the demand d_i of every request R_i is increased to 2^x, the maximum number of requests in the SCRSA solution will be decreased by at most a factor of $2 + \frac{1}{\alpha}$ where $\alpha = \lfloor \frac{S}{2^x} \rfloor$.*

Proof: Let $\mathcal{R}^*$ be the set of requests in the optimal solution. We define $\mathcal{R}_{x-1}$ and $\mathcal{R}_x$ to be the optimal solution of SCRSA when the demand values are changed to 2^{x-1} and 2^x, respectively. Obviously, (1) $|\mathcal{R}_x| \leq |\mathcal{R}^*| \leq |\mathcal{R}_{x-1}|$. We note that when the demands of the requests are all uniform and equal to d, SCRSA problem with a given routing is equivalent to the problem of computing maximum $\lfloor \frac{S}{d} \rfloor$-colorable subgraph in the intersection graph of paths with unit demands. Let $\alpha = \lfloor \frac{S}{2^x} \rfloor$ and $\beta = \lfloor \frac{S}{2^{x-1}} \rfloor$. We have $|\mathcal{R}_x| = \sum_{i=1}^{\alpha} |w_i|$ where w_i is the set of nodes in the path intersection graph colored with ith color when α is the number of colors and demands are one. Similarly, $|\mathcal{R}_{x-1}| = \sum_{i=1}^{\beta} |w'_i|$ where w'_i is the set of nodes colored with ith color. Without loss of generality, consider $|w'_i| \geq |w'_{i+1}|, 1 \leq i < \beta$. We know $2\alpha \leq \beta \leq 2\alpha + 1$. We argue that for any set $W \subset \{w'_i | 1 \leq i \leq 2\alpha\}$ where $|W| = \alpha$, $|\mathcal{R}_x| \geq \sum_{j \in W} |w'_j|$. Otherwise, W is a better solution than $\mathcal{R}_x$ and it contradicts with the optimality of $\mathcal{R}_x$. Hence, $\sum_{i=1}^{2\alpha} |w'_i| \leq 2|\mathcal{R}_x|$. Consider $\beta = 2\alpha + 1$. We have

$|w'_\beta| \leq \frac{|\mathcal{R}_{x-1}|}{\beta}$. Hence, $|\mathcal{R}_{x-1}| \leq 2|\mathcal{R}_x| + \frac{|\mathcal{R}_{x-1}|}{\beta}$. Finally, $|\mathcal{R}_{x-1}| \leq (2 + \frac{1}{\alpha})|\mathcal{R}_x|$.

In SCRSA-B, we consider just the requests R_i where $d_i \leq S$; otherwise R_i cannot be routed. First we increase the demand of each request to the power of two. Then we partition all the requests to the sets $\mathcal{U}'_j, j = 2^x, 0 \leq x \leq \lfloor \log_2 S \rfloor$ such that all requests in $\mathcal{U}'_j$ have the same demand size equal to j. If S is not a power of 2 we need to add one more set, $\mathcal{U}'_S$, to the partition where it includes all the requests R_i where $2^{\lfloor \log_2 S \rfloor} < d_i \leq S$. Since all requests in every set have the same demand size, we can use UDA on path intersection graph of every set to compute the largest S-colorable subgraph optimally. Hence, we compute algorithm UDA on each set $\mathcal{U}'_j$ independently. Since UDA considers all requests have unit demand, it needs to compute maximum $\lfloor \frac{S}{j} \rfloor$-colorable subgraph on intersection graph of paths of requests in each $\mathcal{U}'_j$. The algorithm returns the UDA solution of the set $\mathcal{U}'_j$ with the largest value of maximum $\lfloor \frac{S}{j} \rfloor$-colorable subgraph that is denoted by $\mathcal{U}'_{MAX}$. Each color $c, 1 \leq c \leq \lfloor \frac{S}{j} \rfloor$ represents the spectrum interval $[(c-1)j+1, cj]$ of length j. The steps of SCRSA-B is shown in Algorithm 11.

Algorithm 11 SCRSA-B

1: Increase the demand of each request to the power of two.
2: $\mathcal{U}' = \{\mathcal{U}'_j | j = 2^x, 0 \leq x \leq \lfloor \log_2 S \rfloor\}$ such that $\mathcal{U}'_j$ is the set of requests with the same demand size j.
3: If S is not a power of 2, then add set $\mathcal{U}'_S = \{R_i | 2^{\lfloor \log_2 S \rfloor} < d_i \leq S\}$ to $\mathcal{U}'$.
4: **for all** $\mathcal{U}'_j \in \mathcal{U}'$ **do**
5: Compute the *maximum* $\lfloor \frac{S}{j} \rfloor$*-colorable subgraph* on the intersection graph of paths of requests in $\mathcal{U}'_j$ computed by UDA.
6: **end for**
7: Return $\mathcal{U}'_{MAX}$ = the set in $\mathcal{U}'$ with the largest UDA solution

Theorem 4.6.3. *SCRSA-B has a performance bound of* $O(\log S)$.

Proof: Let $\mathcal{R}^*$ and $\mathcal{R}'$ be the set of requests in the optimal solution and SCRSA-B solution respectively. Without loss of generality, we assume that S is a

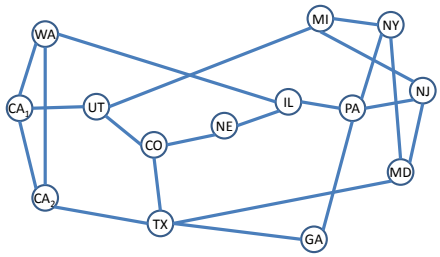
power of two. Let $\mathcal{U}^* = \{\mathcal{U}_j^* | j = 2^x, 0 \leq x \leq \lfloor \log_2 S \rfloor\}$ where $\mathcal{U}_j^* \subseteq \mathcal{R}^*$ is the set of requests in optimal solution with the demand size $2^{x-1} < j \leq 2^x$. Obviously, there exists a set $\mathcal{U}_t^* \in \mathcal{U}^*$ such that $|\mathcal{U}_t^*| \geq \frac{|\mathcal{R}^*|}{\lfloor \log_2 S \rfloor}$. Let $\mathcal{U}_t$ be the set of all requests R_i with demand size $t/2 < d_i \leq t$. We know that $\mathcal{U}_t^* \subseteq \mathcal{U}_t$. We denote the requests in the optimal solution of SCRSA problem when set of input requests is just $\mathcal{U}_t$ by $\mathcal{R}_t^p$. Definitely, $|\mathcal{R}_t^p| \geq |\mathcal{U}_t^*|$. Based on Lemma 1, we know $|\mathcal{R}_t^p|$ is at most a factor of $2 + \epsilon$ (where $\epsilon = \frac{1}{\lfloor \frac{S}{j} \rfloor}$) larger than the solution of UDA with input set $\mathcal{U}_t'$. So, $|\mathcal{R}'| > \frac{1}{2+\epsilon}|\mathcal{R}_t^p|$. Hence, $|\mathcal{R}'| \geq \frac{|\mathcal{R}^*|}{(2+\epsilon)\lfloor \log_2 S \rfloor}$.

4.6.3 SCRSA in Rings

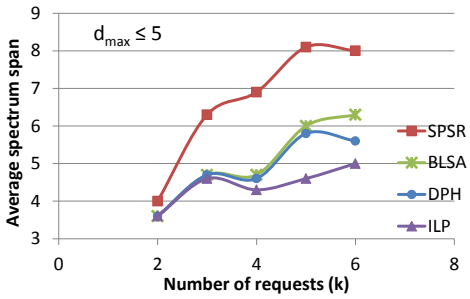
We propose an approximation algorithm called SCRSA-R for SCRSA problem when network is a ring. Similar to RSA-R, we use cut-one-link technique in SCRSA-R. After removing one edge e randomly, the induced network is a chain. Next we use the approximation algorithm proposed in [66] and used in section 4.6.1.

Theorem 4.6.4. *SCRSA-R has a performance bound of* $(4+2/(e-1))^{-1} - \epsilon \approx 1/5.16$.

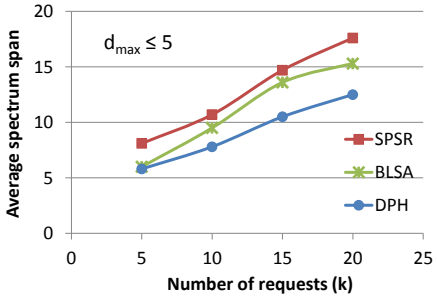
Proof: Let $\mathcal{R}^*$, $\mathcal{R}_c^*$ be the optimal solution of SCRSA problem when network graph is the ring, and the chain after removing an edge e, respectively. $\mathcal{R}'$ also denotes the solution of SCRSA-R. Based on the proof of Theorem 4.6.2, we know that (1) $|\mathcal{R}'| \geq ((2 + 1/(e-1))^{-1} - \epsilon)|\mathcal{R}_c^*|$. We will show that (2) $|\mathcal{R}_c^*| \geq \frac{1}{2}|\mathcal{R}^*|$. Let $R_e^1 \subseteq \mathcal{R}^*$ be the set of requests in optimal solution whose paths include edge e and $R_e^2 \subseteq \mathcal{R}^*$ includes the rest of the requests. Apparently, (3) $|\mathcal{R}^*| = |R_e^1| + |R_e^2|$ and (4) $|\mathcal{R}_c^*| \geq |R_e^2|$. Since, the requests in $|R_e^1|$ routed through e, they are allocated non-overlapping intervals. Therefore, just the requests in $|R_e^1|$ can be rerouted avoiding e while using the same non-overlapping intervals. Hence, (5) $|\mathcal{R}_c^*| \geq |R_e^1|$. From (3)-(5) we can conclude (2) and then using (1) and (2) the theorem is proven.



(a)



(b)



(c)

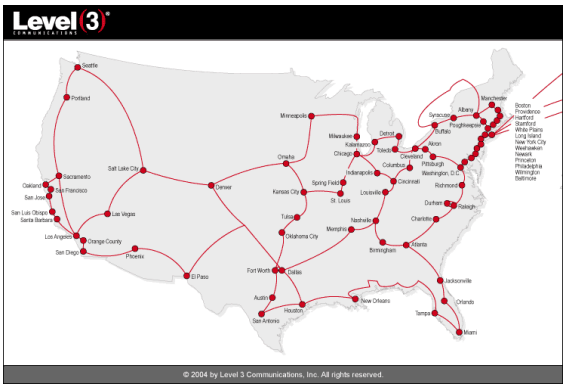
Figure 4.2: (a) The 14-node NSF Network, (b) The average spectrum span in 14-node NSF Network for $k \leq 6$ and $d_{max} \leq 5$ (c) $k \geq 5$ and $d_{max} \leq 5$

4.7 Experimental Results and Discussion

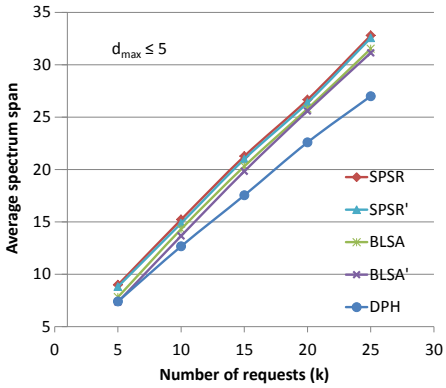
4.7.1 Off-line

In this section we present results of our extensive simulation that demonstrate the efficacy of our heuristic, DPH, for the off-line RSA problem by comparing it against (i) the optimal solution and (ii) the solutions obtained by executing the two heuristics proposed in [45].

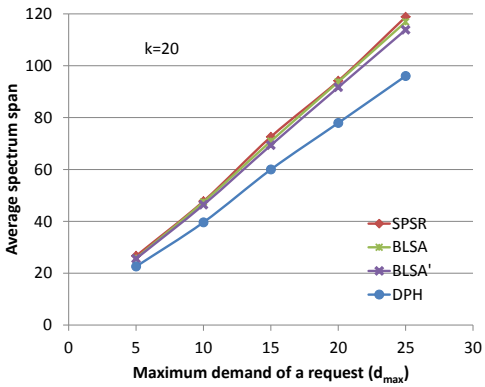
We find the optimal solution of the RSA problem by solving an ILP using the software package CPLEX. In order to minimize used spectrum, our heuristic exploits *disjoint paths* to establish routes between source and destination node pairs. One of the heuristic proposed in [45] uses shortest path with maximum spectrum reuse and the other uses balanced load spectrum allocation. We will refer to the optimal solution as ILP, our heuristic as DPH and the two heuristics in [45] as $SPSR$ and



(a)



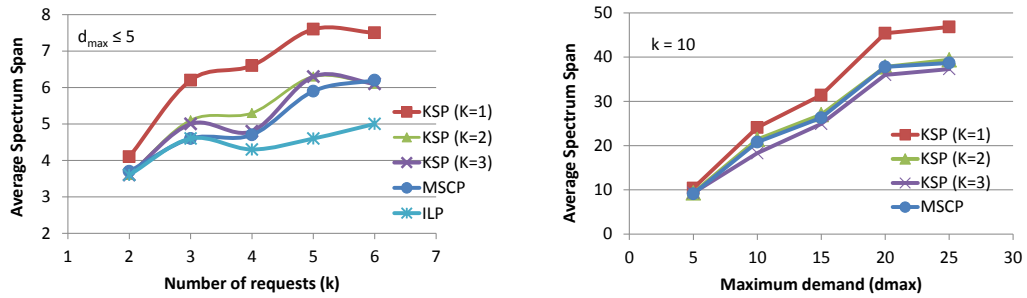
(b)



(c)

Figure 4.3: (a) Level-3 fiber network over US, (b) The average spectrum span in Level-3 network for $d_{max} \leq 5$ (c) $k = 20$

$BLSA$ respectively. All three heuristics $SPSR$, $BLSA$ and DPH operate in two phases. In the first phase it computes the routes (paths) and in the second phase it allocates spectrum to these paths. In the spectrum allocation phase of the $SPSR$ and $BLSA$ the computed paths are partitioned into sets of disjoint paths (starting from the path with the largest demand). Obviously, the paths belonging to a set can reuse the spectrum. However, paths belonging two different sets may not be able to share spectrum. DPH is not only different from $SPSR$ and $BLSA$ in the routing part, it is also different in the spectrum allocation part. In the routing part, DPH greedily selects the largest set of disjoint paths, staring with the request with the



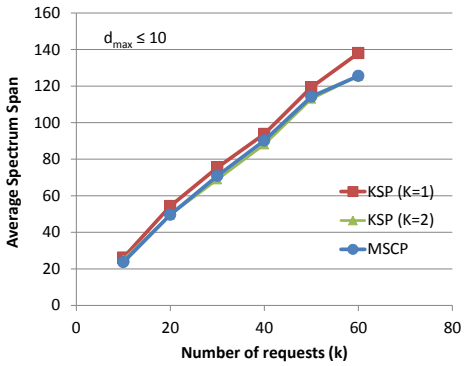
(a) (b)

Figure 4.4: (a) The average spectrum span in NSF Network for different values of k where $d_{max} \leq 5$ in On-line RSA, (b) different values of d_{max} where $k = 10$ in On-line RSA

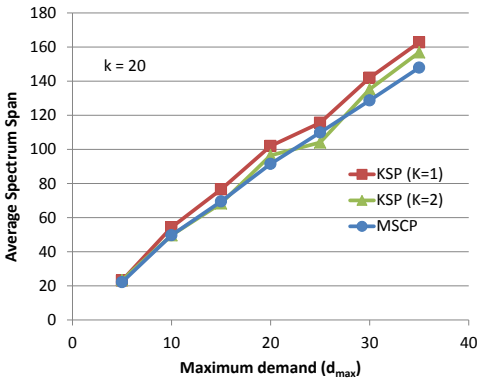
largest demand. Once such a set is found, it goes back to find another large set of disjoint paths. This process continues till all the paths belong to exactly one set of disjoint paths. This process creates a set of *disjoint path sets* $DPS_1, DPS_2, \ldots, DPS_r$. Clearly, the paths belonging to $DPS_i, \forall i, 1 \leq i \leq r$ can share spectrum. However, two paths P_i and P_j, belonging to two different disjoint path sets, may still be able to share spectrum if they do not intersect. During the spectrum allocation phase, DPH sequentially examines $DPS_1, DPS_2, \ldots, DPS_r$ and within each disjoint path set DPS_x, assigns a spectrum slice to each path sequentially. The spectrum $[a_i, b_i]$ allocated to path P_i is made in such a way that it has the smallest value for its first sub-carrier, a_i.

We perform our experiments on the NSFnet (Fig. 4.2(a)) and the fiber network of Level-3 that spans the continental United States (Fig. 4.3(a)) [68]. We view the NSFnet and Level-3 networks as examples of a small and a large network in terms of the number of nodes, respectively. In Fig. 4.2(b), we present the results obtained from ILP, $SPSR$, $BLSA$ and DPH when executed on the NSFnet. In this set of experiments, the number of requests, k, is varied from 2 to 6 with step of one. For each value of k, we generate 10 instances where each instance has k random requests.

(a)



(b)



(c)

Figure 4.5: (a) Level-3 network over Europe, (b) The average spectrum span in Level-3 network for $d_{max} \leq 10$, (c) $k = 20$

For this set of experiments all the demand values are at most 5, (i.e., $d_{max} \leq 5$). The average spectrum span computed by each of the four methods is shown in Fig. 4.2(b). It may be observed that the average spectrum span of DPH is closest to the ILP. The ratio of the average spectrum span of DPH to ILP is at most 1.2 demonstrating the closeness of the DPH to the optimal. When the number of requests is increased to larger values, such as 10, 15, 20 and beyond, the CPLEX could no longer find the optimal solution within a reasonable amount of time. For large request sets, we compare the performance of $SPSR$, $BLSA$ and DPH and present the results in

Fig. 4.2(c). It may be observed that as the size of the request set increases, DPH consistently outperforms $SPSR$ and $BLSA$ and the average reduction of spectrum span is more than 18% when the request set size exceeds 10.

We perform our next set of experiments on the Level-3 network shown in Fig. 4.3(a). In these experiments, first, we vary k from 5 to 25 with step of 5. For a specific value of k we generate 100 instances. In all these instances, the maximum demand is limited to 5 (i.e., $d_{max} \leq 5$). We present the comparative results of spectrum span computed by DPH, $SPSR$ and $BLSA$ in Table 4.1. The comparative results between DPH and $SPSR$ is shown in $DPH - SPSR$ and comparative results between DPH and $BLSA$ is shown in $DPH - BLSA$ part of the Table 4.1. In columns W, L and T, the results under % shows the percentage of times (out of 100 instances) that DPH *Wins, Loses* and *Ties* against the $SPSR$ and $BLSA$ heuristics. We say that DPH "Wins" against the $SPSR$ ($BLSA$), if the spectrum span computed by DPH is *smaller* than the span computed by $SPSR$ ($BLSA$). For example, when the number of requests, k, is 20, DPH wins 94% of times against $SPSR$ and 89% of times against $BLSA$. Moreover, average savings in the number of sub-carriers (spectrum span) using DPH against $SPSR$ is 4.44 and against $BLSA$ is 3.81, when $k = 20$. It may be noted that in the very few cases, where the DPH loses against $SPSR$ or $BLSA$, the amount (measured in terms of the number of sub-carriers) by which it loses is much smaller than the large number of cases when it wins. The average loss by DPH against $SPSR$ is 1.5 and against $BLSA$ is 2, when $k = 20$.

The dramatic improvement in performance by DPH against $SPSR$ and $BLSA$ might lead one to believe that the the efficacy of DPH is derived primarily from its spectrum assignment phase. To evaluate if this is indeed correct, we modified the spectrum allocation part of both $SPSR$ and $BLSA$ and used the same spectrum allocation technique as in DPH. We denote the modified heuristics by $SPSR'$ and

Table 4.1: Results for USA Network, $d_{max} \leq 5$

No. of Req.	*DPH-SPSR*					*DPH-BLSA*				
	W		L		T	W		L		T
	%	AVG	%	AVG	%	%	AVG	%	AVG	%
5	70	2.26	0	-	30	35	1.43	7	1.29	58
10	83	3.1	0	-	17	77	2.23	6	1.33	17
15	93	4.04	2	1.5	5	86	3.26	3	1.5	11
20	94	4.44	6	1.5	0	89	3.81	9	2	2
25	98	5.91	0	-	2	93	4.88	1	2	6

$BLSA'$. Fig. 4.3(b) depicts the average spectrum span used in all five heuristics on the same sets of instances used in previous experiments. We observe that DPH is more efficient than all the other heuristics even though $SPSR'$ and $BLSA'$ use the same spectrum allocation technique as DPH.

Modification of the spectrum allocation technique in $SPSR'$ and $BLSA'$ slightly improves their performance. However, for $k \geq 15$, DPH still performs better than $BLSA'$ and $SPSR'$ by at least 11% and 12.4% respectively. These results clearly demonstrate that the routing scheme used in DPH plays a significant role in improving its performance over $SPSR$ and $BLSA$.

We also evaluate the performance of DPH for larger values of d_{max}. In the second set of experiments, we change values of d_{max} from 5 to 25 with step of 5, while keeping the number of requests k constant at 20. We compute the average spectrum span over 100 random instances for each value of d_{max}. The results are shown in Fig. 4.3(c). As there was hardly any discernible difference between the performance of $SPSR$ and $SPSR'$, for the sake of clarity we only show the results for $SPSR$. This figure also shows that the spectrum span computed by DPH is smaller than the ones computed by $SPSR$, $BLSA$ and $BLSA'$.

It may be noted in all these experiments DPH performs better than $SPSR$ and $BLSA$ for all values of k and d_{max}, but the amount of improvement increases

with increasing values of k and d_{max}. It may also be noted that DPH takes only a few seconds in all of our experimentations, whereas the ILP takes several hours even when $d_{max} \leq 5$ and $k = 6$.

4.7.2 On-line

In this section we present the results of our extensive simulation that demonstrate the efficacy of our heuristic for the on-line RSA problem by comparing it against (i) the optimal solution and (ii) the solution obtained by executing the heuristic based on K-shortest path and First-Fit technique.

K-Shortest Path Heuristic (KSP): In this heuristic, initially K shortest paths are computed between every pair of nodes in the network using [69] algorithm with $O(k|V|^3)$. When a request R_i arrives, for every path in the K shortest paths between s_i and t_i we compute First-Fit algorithm to find the first available spectrum slice $[a_i, b_i]$ with the length of d_i. Then we select the path whose first available spectrum slice $[a_i, b_i]$ has the smallest b_i. This algorithm takes $O(Kk^2|V|^2)$ for satisfying all the k requests.

We perform our experiments on the NSFnet (Fig. 4.2(a)) and the fiber network of Level-3 that spans Europe (Fig. 4.5(a)) [68].

In Fig. 4.4(a), we present the results obtained from ILP, $MSCP$ and KSP when executed on the NSFnet. In this set of experiments, the number of requests, k, is varied from 2 to 6 with step of one. For each value of k, we generate 10 instances. In each instance we generate k requests randomly and consider them one at a time. For this set of experiments all the demand values are at most 5, (i.e., $d_{max} \leq 5$). The average spectrum span computed by each of the three methods is shown in Fig. 4.4(a). It may be observed that the average spectrum span of $MSCP$ is closest to the ILP almost in all cases. The ratio of the average spectrum span of $MSCP$ to ILP is

at most 1.28 demonstrating the closeness of the $MSCP$ to the optimal. The results in these experiments also show that $MSCP$ works better than KSP algorithm in almost all the cases even when number of paths in KSP is $K = 3$. We repeat similar experiments for larger value of k, where $k = 10$ and we change the value of d_{max} from 5 to 25. The result of these experiments is depicted in Fig. 4.4(b). It can be observed that spectrum span in $MSCP$ is at least 12% smaller than the span in KSP where $K = 1$ and it is even smaller than the one in KSP where $K = 2$. When $K = 3$ in KSP, KSP needs smaller spectrum span than $MSCP$ but its time complexity is at least 3 times $MSCP$.

We perform our next set of experiments on the Level-3 network shown in Fig. 4.5(a). In these experiments, first, we vary k from 10 to 60 with step of 10. For a specific value of k we generate 10 instances. In all these instances, the maximum demand is limited to 10 (i.e., $d_{max} \leq 10$). The average utilized spectrum span is shown in Fig. 4.5(b). These results show that $MSCP$ efficacy with respect to utilized spectrum span is almost the same as KSP when $K = 2$. We also conduct experiments for the case that values of d_{max} is varied from 5 to 25 with step of 5, while keeping the number of requests k constant at 20. We compute the average spectrum span over 10 random instances for each value of d_{max}. The results are shown in Fig. 4.5(c). According to these results, $MSCP$'s performance is better than KSP when $K = 2$ especially for larger values of d_{max}. According to the last experiments we may conclude that $MSCP$ outperforms KSP when $K = 2$ for larger values of d_{max}.

4.8 Conclusion

In this chapter we study the *Routing and Spectrum Allocation* problem in OFDM-based optical networks. We prove that the RSA problem is NP-complete when the network topology is a *chain* or a *ring* and provide approximation algorithms for the RSA problem in the network with these topologies. We also study the on-line ver-

sion of RSA problem and propose an algorithm for the *ring network* with a bounded *competitive ratio*. In addition, we provide heuristics for both off-line and on-line RSA problem in networks with arbitrary topology and measure the effectiveness of the heuristics with extensive simulation. Simulation results demonstrate that our heuristics significantly outperforms several other heuristics proposed recently for the RSA problem. Moreover, we introduce the *Spectrum Constrained Routing and Spectrum Assignment* (SCRSA) problem. The goal of the SCRSA problem is to satisfy the largest number of requests without exceeding the available spectrum span. To the best of our knowledge, the SCRSA problem has not been studied earlier. We propose approximation algorithms for SCRSA when network topology is a binary tree or a ring.

Chapter 5

INFLUENCE PROPAGATION IN SOCIAL NETWORKS IN ADVERSARIAL SETTING

It has been widely observed in various studies in social sciences and economics that an individuals' decision to adopt a product, behavior or innovation is often influenced by the recommendations of their friends and acquaintances. Motivated by this observation, the last few years have seen a number of studies on influence maximization problem in social networks [70, 71, 72, 73, 74, 75, 76]. One major goal of several of these studies is identification of k *most influential nodes* in a network. A product manufacturer may want to identify the k most influential nodes in the network, as she may want to incentivize these nodes to buy the new product by providing free samples to them, on the expectation that once these nodes are convinced about the quality of the product, they will recommend it to their friends on the social network and encourage them to buy the product. This set of k nodes, being the most influential on the network, will have the largest impact on convincing the rest of the nodes about the quality of the product. Since the manufacturer has a fixed budget for advertising, she can provide free samples only to a limited number of nodes in the network. The size of the advertising budget determines the value of the parameter k.

It may be noted that most of the studies on influence propagation are geared toward a *non-adversarial environment*, where only one manufacturer (player) is attempting to influence the nodes of a social network to buy her product. However, in a realistic market scenario, most often there exists multiple players, each attempting to sell their competing products or innovations. For example, just as Coke attempts to convince customers in an emerging market about the quality of their beverage, its main competitor, Pepsi, also does the same. Both the competitors have only a finite

advertisement budget and both of them want to derive the greatest benefit out of their advertising campaign. The goal of both the players often is to capture a share of this emerging market that is larger than its competition.

The non-adversarial influence propagation models consider scenarios where a user (a node u in a social network graph $G = (V, E)$) adopts (or does not adopt) an innovation based on how her acquaintances have adopted the innovation. In these models each node u in the social network graph is in one of the following two states: (i) u has adopted innovation A, and (ii) u has not adopted innovation A but u is *open to the idea of adoption*. One can visualize such a scenario by coloring the nodes of the social network graph with *red* if they have adopted the innovation A and with *white* if they have not adopted A yet, but are open to the idea of adopting A in the future. As the diffusion process progresses with time, by observing changing color of the nodes of the graph one can infer if innovation A is being adopted by the members of the social network. Although, we focus on influence propagation in social networks, conceptually, the scenario is identical for spread of any *contagion* through a network - be it *spread of diseases* through a *human contact network* or *spread of worms* through the Internet.

The influence (contagion) propagation models can be divided into three distinct classes:

- Class I: Non-adversarial
- Class II: Adversarial with passive adversary
- Class III Adversarial with active adversary

The problems in classes I and II can be stated as follows:

- Class I: How to identify a set of k initial (seed) nodes, so that once they are *influenced/infected*, they will infect the largest number of uninfected nodes in the network?
- Class II: Given that a subset of the nodes is already *influenced/infected*, how to identify a set of k uninfected nodes, so that when they are *immunized*, they will have the largest impact in preventing the uninfected nodes from being infected.

In most of the influence propagation models, influence propagates in a step-by-step fashion and as such there is a notion of time step (or propagation step) involved. The expected number of nodes influenced at the end of time step D is at most the expected number of nodes influenced at the end of time step $D + 1$. In other words, expected number of nodes influenced at the end of time step D is a *non-decreasing* function of D.

The Class I influence propagation problem considered in [75] may be viewed to have three dimensions, (i) the number of seed nodes activated at the beginning (*budget or cost of influence*), (ii) the expected number of activated nodes at the end of propagation (*impact or coverage of initial seed nodes*), and (iii) *time steps* for propagation. The objective of the influence maximization problem considered in [75], is to *maximize the coverage subject to a budget constraint but without any constraint on the number of time steps.*

The Class I problem considered in [75] can be stated in the following way: "Which k white nodes should be colored red initially, so that the largest number of white nodes turn to red at the end of propagation process?". The Class II problems can be stated in the following way: "Given that some nodes are already colored red,

which k white nodes should be colored blue, so that this set of nodes will have the largest impact in preventing the white nodes from turning red.

In Class I, there is no notion of an *adversary*. The red nodes are trying to convert all the white nodes into red nodes and there is no agent that is actively trying to prevent this conversion. The Class II, although it has a notion of an adversary (i.e., the blue nodes) which is trying to slow down (or stop) white-to-red conversion, at best this agent can be viewed as a *passive adversary*, because its goal is to prevent white-to-red conversion, and it is not engaged in white-to-blue conversion. This gives rise to Class III, a truly adversarial scenario, where the red agent is trying to convert all the white nodes into red, while the blue agent is trying to convert all the white nodes into blue. In this case, the blue agent can be viewed as an *active adversary* of the red agent.

The Class III models the scenario where a node u is being actively encouraged by an adversary not only *not to adopt the innovation* but also *to adopt a competing innovation*. In this case, each node u in the social network graph can be in one of following three states: (i) u has adopted innovation A , (ii) u has adopted innovation B, and (iii) u has not adopted any innovation A or B but is open to the idea of adopting either one of them. This adversarial scenario can be viewed as a classic case of a strategic conflict game between the proponent(s) and the opponent(s) of adoption of an innovation and a game is won by the proponent(s) if u decides to adopt the innovation A.

This research studies a Class III scenario where two vendors (players) are trying to sell their competing products by influencing the nodes of a social network. The goal of both the players is *to have a market share that is larger than its competition*. It considers the scenario where the first player (P_1) has already chosen the k nodes to have a large influence (coverage) on the social network. The second player is aware

of the first player's choice and the goal of the second player (P_2) is to *identify a smallest set of nodes (excluding the ones already chosen by the first player) so that the number of nodes influenced by the second player will be larger than the number of nodes influenced by the first player within D time steps.* In other words, the objective of the problem is to *minimize the cost subject to the constraint that the coverage of the second player is larger than the coverage of the first player within D time steps.* Since the goal of the second player is to win the "game" (i.e., to have a larger coverage or market share), with influencing (incentivizing) as few nodes as possible, the problem under study in this research is referred to as the "Winning with Minimum Investment" (WMI) problem. In [71], the authors study a similar problem belonging to class III. However, the objective of the problem studied in [71] is different from the one being studied in this research. The goal of the second player in the problem studied in [71] is not to defeat the first player with least amount of investment, but to maximize its own influence.

Using the same two influence propagation models introduced in [71], our contributions may be listed as follows:

- Introduction of a new influence propagation problem in an adversarial setting where the goal of the second player is to defeat the first within D time steps and least amount of cost (i.e., number of seed nodes)
- NP-Hardness proof for the problem under both the influence propagation models
- Approximation algorithm for the problem with a tight performance bound
- Experimental evaluation of the Approximation algorithm with collaboration network data

Experimental results show that utilizing the proposed algorithm, the second player can easily defeat the first, if the first player utilizes the node degree or closeness centrality based algorithms for the selection of the initial (seed) nodes. The proposed algorithm also provides better performance for the second player if she utilizes it instead of the algorithm to maximize influence proposed in [71], in the sense that it requires selection of a fewer number of seed nodes to defeat the first player.

The rest of the chapter is organized as follows. The section 5.1 summarizes related work on influence propagation. The section 5.2 describes the propagation models used in this research in detail. The sections 5.3, 5.4 and 5.5 discuss the problem statement, computational complexity and approximation algorithm results respectively. The results of experimental evaluation is presented in section 5.6 and section 5.7 concludes the chapter.

5.1 Related Work

The studies on identification of influential nodes in a social network were triggered by a paper authored by Domingos and Richardson [73]. They introduced the notion of "network value" of a node in a social network and using a Markov random field model where a joint distribution over all node behavior is specified, computed the network value of the nodes. Kempe, Kleinberg and Tardos followed up the work in [73] by providing new models derived from mathematical sociology and interacting particle systems [75]. They made a number of important contributions by providing approximation algorithms for maximizing the spread of influence in these models by utilizing the *submodularity* property of the objective functions. In addition to providing algorithms with provable performance guarantee, they also presented experimental results on large collaboration networks. Their experimental results showed that their greedy approximation algorithm significantly out-performed the node selection heuristics based on *degree centrality* and *distance centrality* [77].

The approximation algorithm proposed in [75] is compute-intensive. Accordingly, several researchers approached the issue of scalability from different directions. Chen et. al. in [72] provided improvement of the original greedy algorithm of [75] and proposed a *degree discount* heuristic to improve influence spread. Mathioudakis in [76] introduced the notion of *sparsification of influence networks* and presented an algorithm, SPINE, to compute the "backbone" of the influence network. Utilizing SPINE as a pre-processing step for the influence maximization problem, they showed that computation on the sparsified model provided significant improvements in terms of speedup without compromising accuracy. Wang *et. al.* in [78] considered the influential node identification problem in a mobile social network and presented a two step process, where in the first step, communities in the social network are detected and in the second step a subset of communities is selected to identify the influential nodes. Experimental results with data from large real world mobile social network showed that their algorithm performed an order of magnitude faster than the state-of-the-art greedy algorithm for finding the top-k influential nodes. A simulated annealing (SA) based algorithm for finding the top-k influential nodes was presented in [79]. It has been reported in [79], that using data from four real networks, the SA based algorithm performed 2-3 orders of magnitude faster than the state-of-the-art greedy algorithm.

In addition to attempts to address the scalability issue of the greedy algorithm in [75], efforts on variations of the original problem formulation and also the computation model is underway in the research community. In [74] two new problem formulations are provided. In the first formulation, the goal is to minimize the cost, subject to the constraint that coverage exceeds a minimum threshold ν without any constraint on the number of time steps. The goal of the second formulation is to minimize the number of time steps, subject to a budget constraint k and a coverage constraint ν. For the first version of the problem, the authors provide a simple

greedy algorithm and show that it provides a bicriteria approximation. For the second version, they show that even bicriteria or tricriteria approximations are hard under several conditions. In [80], the authors argue that a user (a node in the social network) may be influenced by positive recommendations from a group of friends (neighbors in the network) but that does not necessarily imply that she will adopt the product herself. However, she may pass on her positive impression about the product to another group of friends. Clearly, such a model departs from the model considered in [75].

The authors in [80] consider an "adoption maximization" problem instead of "influence maximization" problem and present both analytical and experimental results for the new problem. The authors in [81] argue that a limitation of the traditional influence analysis technique is that they only consider positive relations (agreement, trust) and ignore the negative relations (distrust, disagreement). Moreover, the traditional techniques also ignore *conformity* of people, i.e., an individual's inclination to be influenced. The paper studies the interplay between influence and conformity of each individual and computes the influence and conformity indices of individuals. The authors in [82] suggest an alternate way of measuring the influencing capability of an individual on her peers, through the individuals *reach* within the social network for certain *actions*.

All the references discussed in the last three paragraphs pertain to the class I (non-adversarial) problems as defined in the previous section. Results on study of class II problems (adversarial with passive adversary) is presented in [83]. It focuses on identification of *blockers*, the nodes that are most effective in blocking the spread of a dynamic process through a social network, and reports that simple local measures such as the degree of a node are good indicators of its effectiveness as a blocker. The blocker identification problem has been extensively studied in the public health

community, where the goal is to stop or slow down progress of an infectious disease by immunizing a small set of key individuals in the community.

As we discussed before, the WMI problem studied in this chapter belongs to Class III (adversarial with active adversary). Unfortunately, there exists only a handful of studies on problems belonging to Class III. Bharathi *et. al.* were one of the earliest to study a Class III problem [70]. They proposed a mathematical model for diffusion of multiple innovations in a network, an approximation algorithm with a $(1 - 1/e)$ performance guarantee for computing the best response to an opponent's strategy. In addition they prove that the "price of competition" of the game is at most 2. While game theoretic framework was utilized for deriving the results in [70], Carnes *et al.* used an algorithmic framework to study a Class III problem [71]. Their research primarily extends the problem studied in [75] from the Class I domain to the Class III domain. They study the follower's perspective (i.e., the player who entered the market after the first player) and investigate how a follower can maximize her influence in the network with a limited budget, given that the first player has already entered the market and influenced a certain number of key individuals (nodes in the network). They prove that the influence maximization problem for the second player is NP-complete and provide an approximation algorithm that is guaranteed to produce a solution within 63% of the optimal. Adversarial models in evolutionary game dynamics was studied by Istrate em et al. in [84].

In all the problems discussed in [70, 71] once a node adopts an innovation (i.e., changes its color from white to red or white to blue), it is not allowed to change its color, i.e., the model precludes the possibility of an individual changing her mind. However, the model considered by Nowak *et al.* in [85] there are only red and blue nodes (no white nodes) and the model allows a node to change its color from red to blue and vice-versa. Although this model was developed to capture a biological phe-

nomenon involving viruses and cells, this model can be equally effective in capturing the phenomenon of the spread of ideas and behaviors in human population. Using evolutionary game theoretic and evolutionary graph theoretic techniques, the authors establish fundamental laws that govern choices of competing players regarding strategies.

5.2 Influence Propagation Models

A number of influence propagation models for the
non-adversarial scenario have been proposed in the literature [75]. Among these, the *Linear Threshold Model (LTM)* and the *Independent Cascade Model (ICM)* have drawn most attention in the research community. As indicated earlier, the literature on influence propagation in adversarial scenario with active adversaries is very sparse [70, 71]. Bharati *et al.* in [70] and Carnes *et al.* in [71] have studied influence propagation in adversarial scenario with active adversaries, and have proposed two different models for it. Both of these two models are generalizations of the Independent Cascade Model. The model proposed in [70] is suitable for a multi-player scenario, whereas the model proposed in [71] is for two competing players. Bharati *et al.* in [70] study the problem from a game-theoretic perspective and focus on finding best response strategies for the players. Carnes *et al.* on the other hand study the problem from an algorithmic perspective. Since we study the problem with only two competing players, the models proposed in [71] are more relevant for this study than the one proposed in [70]. Accordingly, the influence propagation models of [71] are used here. Since these models, *Distance-based Mode*l (DBM) and *Wave-propagation Model* (WPM), are generalization of the ICM, we first discuss ICM and then DBM and WPM.

5.2.1 Independent Cascade Model

The social network is modeled as a graph $G = (V, E)$, where each node represents an individual. Each individual may either be *active* (i.e., has adopted innovation) or *inactive*. A node can switch from an inactive state to an active state but cannot switch back in the other direction. The propagation process from the perspective of an inactivate node $v \in V$ can be described in the following way: With passage of time, more and more of v's neighbors become active and this may cause v to become active at some time step. The activation of v in turn may trigger activation of some of v's inactive neighbors. In the ICM model there exists a set of nodes $V' \subset V$ that are active (seed nodes) initially and the rest of the nodes are inactive. Influence propagation unfolds in discrete steps following a randomized process. When a node v first becomes active in time step d, it has a single chance to activate each of its inactive neighbors w with probability $p_{v,w}$ at time step $d+1$. If v succeeds, w become active at $d + 1$. However, if v fails, it doesn't get another chance to turn w active. The process of conversion of nodes from the inactive to the active state continues, till no further activation is possible. Since v influences w with probability $p_{v,w}$, the $v - w$ edge is considered active with probability $p_{v,w}$. The set of active edges is denoted by E_a.

5.2.2 Generalized ICM for Adversarial Scenario

The ICM can be adapted to handle adversarial scenario by allowing the nodes to be in one of the following three states - (i) active by adopting innovation A, (ii) active by adopting innovation B, and (iii) inactive. We use the notation I_A and I_B to indicate the initial adopters (seed nodes) of technologies A and B respectively. The nodes in the set $V - (I_A \cup I_B)$ are the nodes that are inactive initially. The sets I_A and I_B are disjoint, i.e., $I_A \cap I_B = \emptyset$. Just as in ICM, an active node v may influence each one

of its inactive neighbors w with probability $p_{v,w}$. However, in an adversarial scenario, an inactive node w, may be in a situation where one of its active neighbor v attempts to influence w with innovation A, whereas another active neighbor u attempts to influence w with innovation B. In order to deal with this situation, the authors in [71] proposed two new models - (i) Distance-based Model, and (ii) Wave-propagation Model. The models specify the probability with which the node w will be influenced, when its active neighbors attempt to influence w with two competing technologies. The GICM operates on a random subgraph of the social network graph $G = (V, E)$, where each edge is included independently with probability $p_{v,w}$. The details of these two models are described in the following two subsections.

5.2.3 Distance-based Model

Suppose that $d_u(I, E_a)$ denotes the shortest path distance from the node u to the node set I where $I = I_A \cup I_B$ along the active edges in the edge set E_a. If u is not connected to any node of I using only the active edges E_a, then $d_u(I, E_a) = \infty$. Let $\nu_u(I_A, d_u(I, E_a))$ and $\nu_u(I_B, d_u(I, E_a))$ be the number of nodes in I_A and I_B respectively, at distance $d_u(I, E_a)$ from u along edges in E_a. The probability that node u adopts innovation $i \in \{A, B\}$ when maximum number of propagation steps is D is denoted by $P_i(u|I_A, I_B, E_a, D)$ and is computed in the following way:

if $d_u(I, E_a) \leq D$, $P_i(u|I_A, I_B, E_a, D) = \frac{\nu_u(I_i, d_u(I,E_a))}{\nu_u(I_A, d_u(I,E_a)) + \nu_u(I_B, d_u(I,E_a))}$;

otherwise, it is zero.

In this model the expected number of nodes which adopt $i \in \{A, B\}$ will be computed in the following way:

$$\sigma_j(I_A, I_B, D) = \mathbb{E}\left[\sum_{u \in V} P_i(u|I_A, I_B, E_a, D)\right]$$

where $j = 1$ if $i = A$; else $j = 2$ and the expectation is over the set of active edges.

5.2.4 Wave-propagation Model

In this model, in step $d < D$ all nodes that are at distance $d-1$ from some node in I have adopted technology A or B and all nodes that are farther than $d-1$ from I have not adopted any technology yet(where the distance is measured with respect to active edges). Every node at distance d from I chooses one of its neighbors at distance $d-1$ from I independently at random and adopt the same technology as its neighbor. For every node u, S denotes the set of neighbors of u that are closer to I than u; i.e., their distance from I is $d_u(I, E_a) - 1$. In this model $P_i(u|I_A, I_B, E_a, D)$, the probability that node u adopts innovation $i \in \{A, B\}$ in at most D steps, is computed as follows:

If $d_u(I, E_a) \leq D$, $P_i(u|I_A, I_B, E_a, D) = \frac{\sum_{v \in S} P_i(v|I_A, I_B, E_a, D)}{|S|}$;

otherwise, it is zero.

In this model the expected number of nodes which adopt $i \in \{A, B\}$ will be computed in the following way:

$$\sigma_j(I_A, I_B, D) = \mathbb{E}\left[\sum_{u \in V} P_i(u|I_A, I_B, E_a, D)\right]$$

where $j = 1$ if $i = A$; else $j = 2$ and the expectation is over the set of active edges.

5.3 Problem Statement

The WMI problem can be stated informally as follows: Given a diffusion model and the information that a subset of network nodes I_A have already adopted innovation A marketed by player P_1, what is the fewest number of nodes should player P_2 (marketing innovation B) target so that by the end of D time steps, the number of nodes that adopt innovation B will exceed the number of nodes that adopt innovation A? If $\sigma_1(I_A, I_B, D)$ and $\sigma_2(I_A, I_B, D)$ denote the expected number of nodes that adopt

innovations A and B respectively within D time steps, the objective of the WMI problem is to

$$\begin{aligned} & \text{minimize} \mid I_B \mid \\ \text{subject to} \quad & \sigma_2(I_A, I_B, D) > \sigma_1(I_A, I_B, D) \end{aligned}$$

5.4 Computational Complexity

In this section, we prove that WMI problem is NP-hard for both propagation models.

5.4.1 Distance-based Model

Decision version of WMI: Is there a set I_B where $|I_B| \leq M$ and $\sigma_2(I_A, I_B, D) > \sigma_1(I_A, I_B, D)$?

Theorem 5.4.1. *WMI is NP-hard for the distance-based model.*

Proof: In order to prove that WMI is NP-hard when diffusion is based on distance based model, we reduce the NP-compete *Set Cover* problem to WMI. The decision version of the *Set Cover* problem is defined in the following way: A ground set of elements $S = \{e_1, e_2, \ldots, e_n\}$, a collection of sets $C = \{s_1, s_2, \ldots, s_m\}$ such that $s_i \subseteq S$ and a positive integer $K \leq |C|$ are given. The question is whether there exists a collection $Q \subseteq C$ that covers all the elements in S and $|Q| \leq K$.

Given an instance of set cover problem we construct an instance of WMI. We compute $G = (V, E)$ in the following way. For every element $e_i \in S$ we add a node e_i and for every set $s_j \in C$ we add a node s_j to V. We add an edge (e_i, s_j) to E for every e_i and s_j if $e_i \in s_j$. Also, we add a node a and nodes $x_1, \ldots, x_n$ to V. Then, for every e_i we add edges (a, x_i) and (x_i, e_i) to E. Moreover, for every e_i we add a set of r nodes, $L_i = \{l_{i,j} | 1 \leq j \leq r\}$ to V and we connect them directly to e_i. We identify the value of r later in the proof. Finally, we add $n \times r$ additional nodes, $y_1, \ldots, y_{n \times r}$, to V and edges $(y_t, a), 1 \leq t \leq n \times r$ (Fig. 5.1). We consider that all edges are active;

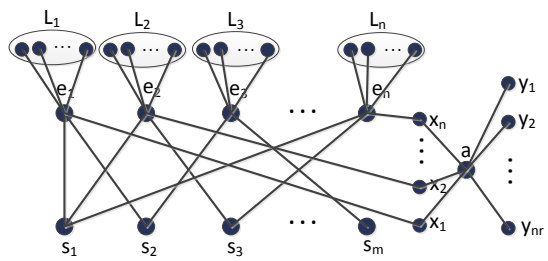


Figure 5.1: Graph $G = (V, E)$ of WMI instance in set cover reduction

i.e., $p_{u,v} = 1$ for all edges in E. We assign $D = 4$ equal to the diameter of the graph G, $M = K$ and $I_A = \{a\}$.

Now, we show that the set cover problem has a solution if and only if there is a set $I_B \subseteq V - I_A$ such that $|I_B| \leq M$ and $\sigma_2(I_A, I_B, D) > \sigma_1(I_A, I_B, D)$. First we consider that there is a collection $Q \subseteq C$ that covers S and $|Q| \leq K$. Then I_B includes all nodes s_j corresponding to the sets in Q. In this case, all e_i will be at distance one from I_B and two from I_A. So, all e_i and the nodes in L_i will adopt I_B with probability one. Moreover, the nodes $s_j \notin I_B$ are two hops away from I_B while 3 hops away from I_A. Hence, all nodes s_j will adopt I_B. Therefore, we have $\sigma_2(I_A, I_B, D) = m + n(1 + r)$; so, $\sigma_2(I_A, I_B, D) > \sigma_1(I_A, I_B, D)$.

Next, we show that if there is no collection Q of size K that covers all elements then there is no set $I_B \subseteq V - I_A$ of size M where $\sigma_2(I_A, I_B, D) > \sigma_1(I_A, I_B, D)$. Considering that set cover does not have a solution, there should be at least one e_i whose distance from I_B cannot be one; so, there is an e_i and consequently nodes in L_i that choose A with the probability at least $\frac{1}{K+1}$ and the probability that they choose B is at most $\frac{K}{K+1}$. Also, at most K nodes from $x_1, \ldots, x_n$ can be at distance less than or equal to 1 from I_B. Hence $n - K$ of them will adopt A with probability one. Therefore, we have $\sigma_2(I_A, I_B, D) \leq m + (n - 1)(1 + r) + \frac{K}{K+1}(r + 1) + K$ and $\sigma_1(I_A, I_B, D) \geq 1 + nr + n - K + \frac{1}{K+1}(r + 1)$. We choose r in our instance large enough

such that $r > \frac{(m+2K-2)(K+1)+K-1}{2}$. Then we have $1 + nr + n - K + \frac{1}{K+1}(r+1) > m + (n-1)(1+r) + \frac{K}{K+1}(r+1) + K$; so $\sigma_2(I_A, I_B, D) < \sigma_1(I_A, I_B, D)$.

5.4.2 Wave Propagation Model

Theorem 5.4.2. *WMI is NP-hard for the wave propagation model.*

Proof: Similar to Theorem 5.4.1, we reduce decision version of *Set Cover* problem to decision version of WMI when wave propagation model is used for diffusion. We construct an instance of WMI in the same way as in Theorem 5.4.1. The only change that should be made to this instance is the value of r which will be computed later.

We need to show that the set cover problem has a solution if and only if there is a set $I_B \subseteq V - I_A$ such that $|I_B| \leq M$ and $\sigma_2(I_A, I_B, D) > \sigma_1(I_A, I_B, D)$. First we consider that there is a collection $Q \subseteq C$ that covers S and $|Q| \leq K$. Then I_B includes all nodes s_j corresponding to the sets in Q. Similar to the proof of Theorem 5.4.1 we have $\sigma_2(I_A, I_B, D) = m + n(1+r)$; so, $\sigma_2(I_A, I_B, D) > \sigma_1(I_A, I_B, D)$.

Next, we show that if there is no collection Q of size K that covers all elements then there is no set $I_B \subseteq V - I_A$ of size M where $\sigma_2(I_A, I_B, D) > \sigma_1(I_A, I_B, D)$. Considering the construction of G and the fact that set cover does not have a solution , there should be at least one e_i whose distance from I_B cannot be one or smaller. Since the node x_i connected to this e_i will have probability 1 to accept A and the maximum number of nodes in first hop neighborhood of e_i that are at distance one from $I_A \cup I_B$ is $m+1$, there is an e_i and consequently nodes in L_i that choose A with the probability at least $\frac{1}{m+1}$ and the probability that they choose B is at most $\frac{m}{m+1}$. Also, at most K nodes from $x_1, \ldots, x_n$ or $y_1, \ldots, y_{n \times r}$ can be at distance less than or equal to 1 from I_B. Hence $n(r+1) - K$ of them will adopt A with probability

one. Therefore, we have $\sigma_2(I_A, I_B, D) \leq m + (n-1)(1+r) + \frac{m}{m+1}(r+1) + K$ and $\sigma_1(I_A, I_B, D) \geq 1 + n(r+1) - K + \frac{1}{m+1}(r+1)$. We choose r in our instance large enough such that $r > \frac{m^2}{2} + K(m+1) - \frac{3}{2}$. Then we have $1 + n(r+1) - K + \frac{1}{m+1}(r+1) > m + (n-1)(1+r) + \frac{m}{m+1}(r+1) + K$; so $\sigma_2(I_A, I_B, D) < \sigma_1(I_A, I_B, D)$.

5.5 Approximation Algorithm

Since we proved that finding the optimal solution for WMI is hard, in this section we propose a greedy algorithm called $GWMI$. In this algorithm either of the two propagation models discussed before can be used as the diffusion process.

Let $\omega(I_A, I_B, D)$ be $(\sigma_2(I_A, I_B, D) - \sigma_1(I_A, I_B, D))$. We define F_i to denote the amount of increase in the value of ω when node i is added to I_B; i.e., $F_i = \omega(I_A, I_B \cup \{i\}, D) - \omega(I_A, I_B, D)$. Initially I_B is empty. Hence, $\omega(I_A, I_B, D) \leq 0$. The algorithm executes through iterations and in each iteration node $i \in V - I_A$ with the maximum F_i is selected. The steps of the algorithm $GWMI$ has been shown in Algorithm 12.

Algorithm 12 GWMI

Input: $G = (V, E)$, I_A, D

Output: I_B

```
while ω(I_A, I_B, D) ≤ 0 do
    for every node i ∈ V − (I_A ∪ I_B) do
        Compute F_i
    end for
    Select node j with maximum F_j
    I_B = I_B ∪ {j}
end while
return I_B
```

In [75], it is mentioned that computing the exact value of $\sigma_1(I_A, \emptyset, D)$ efficiently is an open question. Similarly, there is no known way to compute $\sigma_1(I_A, I_B, D)$, $\sigma_2(I_A, I_B, D)$ in both propagation models efficiently. However, by sampling the active sets we can get a close approximation with high probability. Given I_A, I_B and a set of active edges E_a, computation of σ_1 and σ_2 in both propagation models has $O(n^3)$ time complexity since it needs computation of single all-pairs shortest paths. Given I_A, I_B and input graph G, using sampling, we can then approximate σ_1 and σ_2 to within $(1+\gamma)$ for any $\gamma > 0$ where the running time depends on $1/\gamma$ [71].

5.5.1 Upper Bound Computation

Theorem 5.5.1. *GWMI has a* $\log n$ *approximation ratio.*

Proof. Let I_B^t be the set of B's initial adopters selected by $GWMI$ at step t. Initially, I_B is empty and $\omega(I_A, I_B^0, D) = -\sigma_1(I_A, \emptyset, D)$. In every iteration t, the nodes in the optimal set of B's initial adopters, I_B^{opt}, will make $\omega(I_A, I_B^{t-1} \cup I_B^{opt}, D)$ positive. We denote the size of I_B^{opt} by OPT and the size of the solution of $GWMI$ by H. Therefore, There will be at least one node in $V - \{I_A \cup I_B^{t-1}\}$ that increases $\omega(I_A, I_B^{t-1}, D)$ at least by $\frac{|\omega(I_A, I_B^{t-1}, d)|}{OPT}$. Let, v_t be the node selected by $GWMI$ at iteration t. Then, $F_{v_t} \geq \frac{|\omega(I_A, I_B^{t-1}, D)|}{OPT}$. Therefore, for $t < H$ we have

$$\begin{aligned} |\omega(I_A, I_B^t, D)| &\leq |\omega(I_A, I_B^{t-1}, D)| - \frac{|\omega(I_A, I_B^{t-1}, D)|}{OPT} \\ &\leq |\omega(I_A, I_B^0, D)|(1 - \frac{1}{OPT})^t \end{aligned}$$

Also, we know that $|\omega(I_A, I_B^0, D)| = \sigma_1(I_A, \emptyset, D) \leq n$. Hence we have

$$|\omega(I_A, I_B^t, D)| \leq n(1 - \frac{1}{OPT})^t \leq ne^{\frac{-t}{OPT}}.$$

Since adding a node to I_B will increase $\omega(I_A, I_B, D)$ at least by one, we need to find the smallest t that $|\omega(I_A, I_B^t, D)| < 1$. Then adding at most one more node will make $\omega(I_A, I_B, D)$ positive. Therefore, $H \leq 1 + OPT \ln n$. We note that this proof holds for both propagation models.

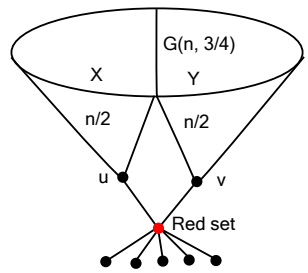


Figure 5.2: Construction of G.

5.5.2 Lower Bound Computation

We now give a construction giving the lower bound for GWMI when distance-based propagation model is used. Let X and Y be disjoint sets of $\frac{n}{2}$ vertices and $G(n, 3/4)$ be the Erdős-Renyi random graph on $X \cup Y$ with $p = 3/4$.

We take two new vertices u and v, connect u to all vertices of X and v to all vertices of Y. Now, we add a disjoint star S with $n+2$ leaves and connect the center of the star to u and v. This yields our graph G (Fig. 5.2).

We consider that the center of the star is the only initial adopter of A (red node), and $p_{u,v}$ is uniform and it is 1 for all the edges of G and $D = 3$. An optimal set of initial adopters of B (initial blue nodes) includes u, v and any of the leaves of S. We claim that the greedy algorithm $GWMI$ will select $\Omega(\log n)$ vertices with high probability, assuming n is large enough.

In order to prove this we first state a technical lemma giving a condition that G satisfies with high probability. Let $S \subseteq X \cup Y$. We say S is *fair* if

1. $|X \setminus \partial(S)| = (1/4)^{|S|}\frac{n}{2} + O(n^{3/4})$ and $|Y \setminus \partial(S)| = (1/4)^{|S|}\frac{n}{2} + O(n^{3/4})$.

   where $\partial(S)$ is the set of one hop neighbors of vertices in S.

We claim the following lemma, whose proof we defer:

Lemma 2. *With probability* $1 - o(1)$ *every set* $S \subset X \cup Y$ *with* $|S| < \frac{1}{100}\ln(n)$ *is fair. Furthermore, the induced graph on* $X \cup Y$ *has diameter* 2*, every vertex in* Y *is at distance at most* 2 *from* u *and every vertex in* X *is at distance at most* 2 *from* v*.*

Assuming Lemma 2 we prove the lower bound. In particular we prove the following: The greedy algorithm selects at least $\frac{1}{100}\ln n$ vertices from $X \cup Y$. We proceed by induction. At the first step, the greedy algorithm has to choose between a vertex in $X \cup Y$, one of u or v, or one of the vertices in the star. Selecting a vertex in the star will cause the number of blue vertices to increase by one and red vertices to decrease, a net change of two. Selecting u (or resp. v) will increase blue (and decrease red) by a total of $1 + \frac{n}{2} + \frac{n}{4}$; since every vertex in X will be at distance 1 from a blue vertex and every vertex in Y will be at distance 2 from both u and the red vertex if u is selected. On the other hand, by fairness, if a vertex x in $X \cup Y$ is selected; the increase in blue is at least $\frac{3n}{4} + \frac{n}{8} + O(n^{3/4})$; since $\frac{3n}{4} + O(n^{3/4})$ vertices are at distance 1 from x and the other $\frac{n}{4} + O(n^{3/4})$ are at distance 2 from both x and the red vertex. Therefore the greedy algorithm will select from $X \cup Y$ at the first time.

Now suppose that the greedy algorithm has selected from $X \cup Y$ a total of $k < \frac{1}{100}\ln n$ times. Let B denote the selected set, and $X' = X \setminus \partial(B)$ and $Y' = Y \setminus \partial(B)$. Every vertex in $X' \cup Y'$ is at distance two from all k blue vertices, and hence they are currently blue with probability $\frac{k}{k+1}$. Furthermore by fairness X' and Y' are both of size $(1/4)^k \frac{n}{2} + O(n^{3/4})$.

Again, the greedy algorithm must choose: If u (or similarly v) is chosen, then increase is at most

$$1 + \frac{1}{k+1}|X'| + \frac{1}{(k+1)(k+2)}|Y'| \qquad (1)$$
$$= \frac{1}{k+1}(1/4)^k\frac{n}{2} + \frac{1}{(k+1)(k+2)}(1/4)^k\frac{n}{2} + O(n^{3/4}).$$

On the other hand, if a vertex x in $X' \cup Y'$ is chosen, the increase is at least

$$\frac{1}{k+1}|\partial(x) \cap X'| + \frac{1}{k+1}|\partial(x) \cap Y'| \qquad (2)$$
$$+ \frac{1}{(k+1)(k+2)}|X' \cup Y' \setminus \partial(X)|$$
$$= 2 \cdot \frac{1}{k+1} \cdot \frac{3}{4}(1/4)^k\frac{n}{2} + \frac{1}{(k+1)(k+2)}(1/4)^{k+1}\frac{n}{2} + O(n^{3/4});$$

therefore, (2) - (1) is positive and hence the vertex in $X' \cup Y'$ will be chosen as desired. We note that this construction is for sufficiently large n and $(1/4)^k n >> n^{3/4}$.

Proof of Lemma 2. Let $S \subset X \cup Y$, with $|S| < \frac{1}{100}\ln n$. Then

$$\mathbb{E}[|X \setminus \partial S|] = (1/4)^{|S|}(|X| - |X \cap S|) = (1/4)^{|S|}\frac{n}{2} + O(\ln n).$$

Let $X_S = |X \setminus \partial S|$. Chernoff bounds imply that

$$\mathbb{P}(|X_S - \mathbb{E}[X_S]| > n^{3/4}) \leq \exp(-\Omega(\frac{n^{3/2}}{\mathbb{E}[X_S]})) \leq \exp(-\Omega(n^{1/2})).$$

Bounds for $|Y \setminus \partial S|$ follow similarly. On the other hand there are at most

$$\sum_{i=1}^{\frac{1}{100}\ln n}\binom{n}{i} \leq \frac{1}{100}\ln(n) \cdot n^{\ln n},$$

sets S. Thus union bounds imply every set is fair with probability $1 - \exp(-\Omega(n^{1/2}))$.

Note that the expected number of common neighbors between x and y in $X' \cup Y'$ is $\frac{9n}{16}$, and Chernoff bounds plus union bounds imply every pair x and y is of distance 2 (and in fact has $(1 - o(1))\frac{9n}{16}$ common neighbors). Likewise, u and a vertex in Y have $\frac{3n}{8}$ expected neighbors and Chernoff bounds imply that every pair has $(1+o(1))\frac{3n}{8}$ common neighbors. Likewise, for v and vertices in X. A union bound over all events completes the proof. $\square$

5.6 Experimental Results

In this section we evaluate the performance of our approximation algorithm, $GWMI$, on a real network data set. It has been suggested in [86] that the co-authorship graphs are representative of typical social networks. As such, we use the real collaboration network data set of the scientists posting preprints on the high-energy theory archive at www.arxiv.org, 1995-1999 [87]. This network has 8361 nodes (authors) and 15751 edges. The largest connected component has 5835 number of nodes (authors) and maximum distance between the nodes in a connected component is 19.

Our experiments were conducted on a high performance computer which is a 5K processor Dell Linux Cluster. The program is parallelized with OpenMP, optimized with Intel compiler and was executed on an 8 core compute node. The cores in the node have equal access to a common pool of shared memory. Each node is comprised of 2.66/2.83 GHz processors, 8MB cache, 16GB memory and 8 cores. Since our experiments required execution of the algorithm on a large number of instantiation of a social network (the graphs were different as their set of active edges were different), we used OpenMP for parallelization of the graph instances for the simulation with one data set.

In the first set of experiments we evaluate the performance of GWMI algorithm against the results obtained from the heuristics based on node degree and closeness centrality. These heuristics are most often used in social networks to identify most influential nodes [75]. We also compare performance of GWMI with the greedy algorithm proposed in [71] for selection of seed nodes for the second player P_2. In our model the first player P_1 is trying to market product A and the second player P_2 is trying to market product B. Since WMI problem is NP-hard and the input data set is large, computation of the optimal solution within a reasonable amount of time is

unlikely. It may be noted that there is no known way of computing the exact value of $\sigma_1(I_A, I_B, D)$ and $\sigma_2(I_A, I_B, D)$ efficiently [75]. Accordingly, we use sampling of the active edge sets to obtain close approximation of $\sigma_1(I_A, I_B, D)$, $\sigma_2(I_A, I_B, D)$ with high probability. As in the experiments reported in papers [75, 71], we assign the edge probabilities to be 0.1. In all the experiments we use WPM as the diffusion model.

The node degree based heuristic selects the nodes in the decreasing order of their degrees and the closeness centrality based heuristic selects the nodes in the increasing order of their average distance to other nodes. The distance between two nodes that are not in the same connected component is taken to be n, where n is number of nodes in the network. In the greedy algorithm proposed in [71], in every iteration the node that increases $\sigma_2(I_A, I_B, D)$ the most is selected. We refer to this algorithm as *Second Player Influence Maximization* (SPIM) algorithm. In these experiments, maximum number of propagation steps is taken to be 10, i.e., $D = 10$. In the experiments, the player P_1 used node degree based heuristic to select its k initial adopters. In our experiments, the size of initial adopters of A is varied from 20 to 100. The results of this set of experiments using the WPM is shown in Fig. 5.3. The Fig. 5.3 shows that all five sizes of the initial adopters of A (20, 40, 60, 80, 100), the GWMI algorithm required the fewest number of initial adopters of B necessary to defeat A's influence at the end of time step 10. The legend Degree-Degree in Fig. 5.3 denotes that both the players are using the node degree based heuristics to select the initial adopters. Similarly,the legend Degree-GWMI denotes that while P_1 is using the node degree based heuristics to select the initial adopters, P_2 is using the GWMI algorithm to do the same.

The Figs. 5.4 and 5.5 show the coverage (i.e., the number of nodes influenced at the end of 10 time steps) for players P_1 and P_2 respectively. Although the GWMI algorithm does not make an effort to minimize the coverage of P_1, it may be observed

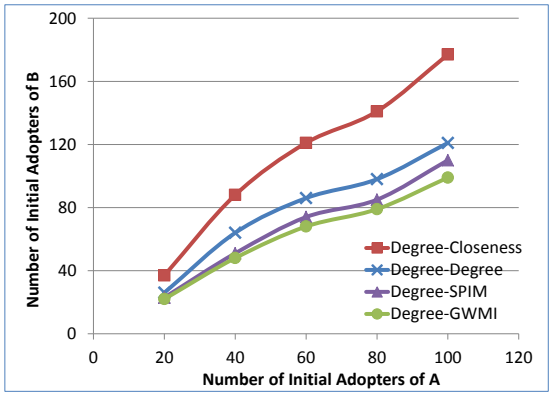


Figure 5.3: Number of initial adopters of B for different values of $|I_A|$

from the Fig. 5.4, the coverage of P_1 is less if P_2 uses GWMI instead of SPIM. Thus P_2 is better off using GWMI instead of SPIM, if in addition to be able to defeat P_1 with least investment (i.e., initial adopters), P_2 wants to have a smaller market share for P_1. The Fig. 5.5 shows the coverage of P_2 at the end of ten time steps. It may be observed from the Fig. 5.5, that at all five data points the coverage for P_2 is highest when she uses the SPIM algorithm. This is not surprising as the stated goal of SPIM is to maximize P_2's coverage (influence). However, this figure may be somewhat misleading because it does not provide the information pertaining to the number of initial adopters required by the SPIM algorithm to achieve the higher coverage. By its stated objective, the number of initial adopters required by GWMI to defeat P_1 cannot be higher than the the number of initial adopters required by SPIM. Once this is factored in, and we compute the coverage per initial adopter, we find that the coverage per initial adopter of the SPIM algorithm is very close to that of the GWMI algorithm. This is shown in Fig. 5.6.

From Fig. 5.3 it is clear that the node degree and centrality based heuristics and the SPIM algorithm require a larger number of initial adopters of B to beat A than is needed by the $GWMI$ algorithm. While this is a negative aspect of SPIM (cost), it also has a positive aspect in the sense that at the end of ten time steps,

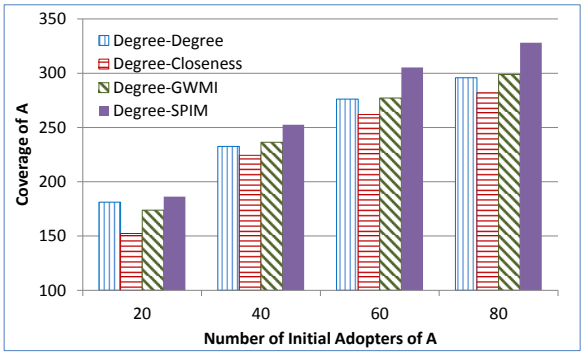


Figure 5.4: Expected number of nodes adopting A after 10 propagation steps

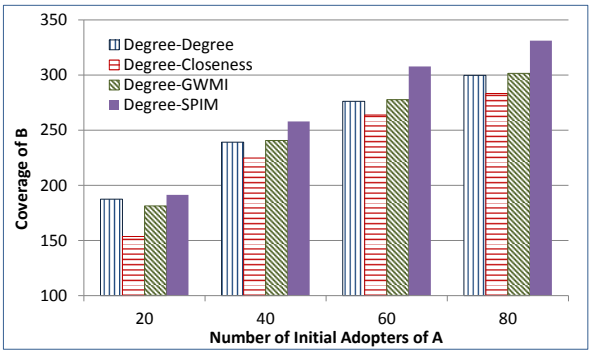


Figure 5.5: Expected number of nodes adopting B after 10 propagation steps

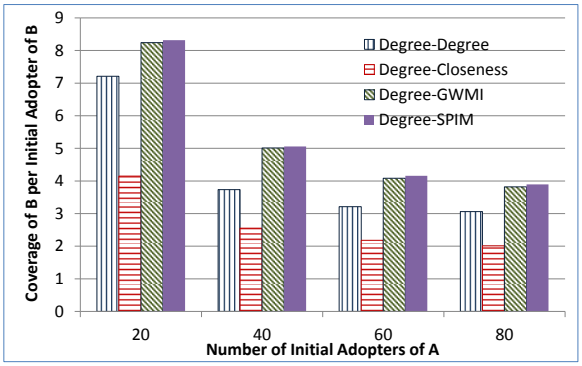


Figure 5.6: Expected number of nodes adopting B per initial adopter of B after 10 propagation steps

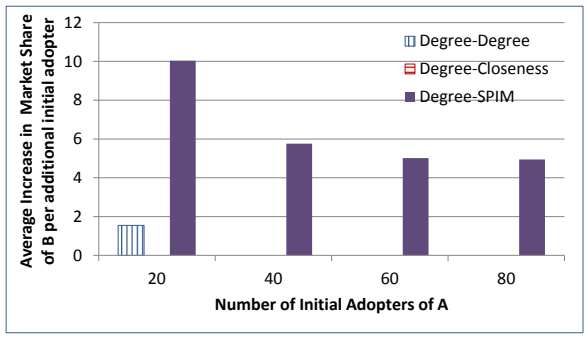


Figure 5.7: Average market share increase that innovation B can capture per additional initial adopter with respect to $GWMI$

it also secures a larger coverage for B (benefit). We compute the additional benefit provided by the additional initial adopters. Let $I_{B(X)}$ be the smallest set of initial adopters of B that is required by algorithm X to defeat A and $\sigma_{2(X)}$ be the expected number of nodes that adopt B after D propagation steps. Here X can be node-degree or centrality based heuristic or the SPIM algorithm. In the case, $(\sigma_{2(X)} - \sigma_{2(GWMI)})$ indicates the additional benefit and $(|I_{B(X)}| - |I_{B(GWMI)}|)$ indicates the additional cost. In this case, $(\sigma_{2(X)} - \sigma_{2(GWMI)})/(|I_{B(X)}| - |I_{B(GWMI)}|)$ indicates the average market share gain of B with each additional initial adopter when using algorithm X. The Fig. 5.7 depicts the results for the heuristics and SPIM. The negative gains are not shown. It may be observed from Fig. 5.7 that the average market share gain of B with each additional initial adopter diminishes with increase of the number of initial adopters of A, when it uses the SPIM algorithm.

While the stated objective of P_2 is to have a larger market share than P_1 with the fewest number of initial adopters, it may also have two other unstated objectives - (i) to have a large $\sigma_{2(X)}$ and (ii) a small $\sigma_{1(X)}$ for all X ($\sigma_{1(X)}$ be the expected number of nodes that adopt A after D time steps). Therefore while considering the benefit of the additional initial adopters, we can consider not only $(\sigma_{2(X)} - \sigma_{2(GWMI)})$ but also $(\sigma_{1(GWMI)} - \sigma_{1(X)})$. It introduces a notion of *extended benefit* by combining these

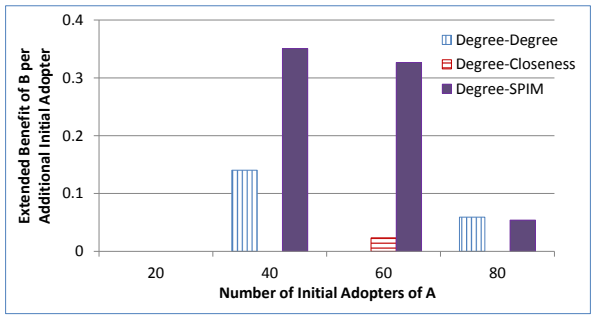


Figure 5.8: Extended benefit that B can capture per additional initial adopter with respect to $GWMI$

two factors in the following way: $(\sigma_{2(X)} - \sigma_{2(GWMI)}) - (\sigma_{1(GWMI)} - \sigma_{1(X)})$. With this notion of *extended benefit*,

$$\frac{((\sigma_{2(X)} - \sigma_{2(GWMI)}) + (\sigma_{1(GWMI)} - \sigma_{1(X)}))}{|I_{B(X)}| - |I_{B(GWMI)}|}$$

indicates the average market share gain of B with each additional initial adopter when using algorithm X. The Fig. 5.8 depicts the results for the heuristics and SPIM. It may be observed from Fig. 5.7 that when extended benefit is considered, the average market share gain of B with each additional initial adopter diminishes even more drastically with increase of the number of initial adopters of A, when it uses the SPIM algorithm. Moreover, the gain of each additional initial adopter is smaller than 1 and implies that the additional adopter is not worth its cost.

In the second set of experiments we investigate different strategies for selection of initial adopters of A when P_2 uses $GWMI$. The strategies that we consider for selection of initial adopters of A includes the greedy algorithm proposed in [75] and heuristics based on node degree and closeness centrality. In these experiments WPM is used as diffusion model and $D = 10$.

Fig. 5.9 depicts the results of these experiments. We observe that the closeness-centrality based heuristic performs poorly in comparison to other two algorithms. This is true because the number of initial adopters of B that it needs to defeat A's

overall influence (coverage) is much smaller than the size of initial adopters of A. More specifically, for closeness-centrality based heuristic, for $|I_A|$ values greater than 60, the number of initial adopters of B is less than 50% of $|I_A|$. This set of results show that if the influence maximization algorithm (IM) proposed in [75] is used for the selection of I_A, it forces P_2 to select a large set for I_B in order to be able to defeat P_1 within D time steps.

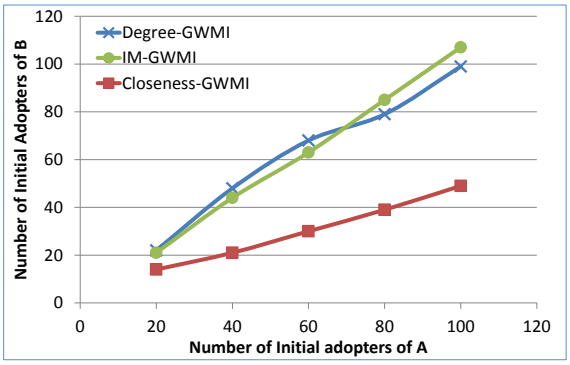


Figure 5.9: Size of initial adopters of B for different values of $|I_A|$

5.7 Conclusion

In this chapter we have introduced a new influence propagation problem in an adversarial setting where the goal of the second player is to defeat the first within D time steps and least cost, measured in terms of the number of seed nodes. Considering two different influence propagation models, we provided the NP-Hardness proof for the problem and an approximation algorithm with a tight performance bound. In addition, we evaluated the performance of the approximation algorithm with collaboration network data. We can envisage at least two new directions of research with this problem. In the first direction, P_2 is not aware of P_1's choice. In the second direction, back and forth transition of the nodes between two competing products is allowed.

Chapter 6

CONCLUSION ANS FUTURE WORK

Motivated by the prominence of efficient resource allocation in networks, in this dissertation several important resource allocation problems in communication and social networks are presented and studied. The study on communication networks are focused on three different communication networks: 1. airborne networks, 2. wireless sensor networks and 3. optical networks.

In the domain of airborne networks, the system model and architecture for Airborne Networks (AN) are discussed. The problem of maintaining the connectivity in the underlying dynamic graphs of airborne networks when trajectories of nodes are given is studied. Some techniques are developed to compute the *dynamic topology* of the AN at any instance of time and an algorithm is proposed to compute critical transmission range when all nodes are operational (non-faulty scenario). The faulty scenario is also investigated where a region may fail. An algorithm is proposed to find the minimum transmission range necessary to ensure that the surviving nodes of the network remain connected, even when all or some nodes of region fail due to an enemy attack. Moreover, the critical transmission range in delay tolerant airborne networks, CTR_D, is defined and an algorithm to compute CTR_D is proposed. There are several problems in this domain that are not explored and lead the way to the future direction of this research problems.

- In this dissertation, it is considered that the trajectories of the backbone nodes are predictable and the backbone network acts as a mobile infrastructure. However, the existence of infrastructure oriented airborne network makes it more vulnerable to the enemies in military applications. There has been some research on computation of critical transmission range in mobile ad hoc networks

where the trajectories are not predictable [10]. However, there is no such results for the faulty or delay tolerant scenarios.

- The focus of the research in this dissertation is on network connectivity and it is considered that trajectories of the backbone nodes are given in a way that the client nodes are covered all the time. However, it will be interesting to consider the following problem: For a given number of ANPs, design the flight paths for the ANPs such that the clients are covered, and the backbone network is connected all the times and the objective is to minimize transmission range of ANPs. There is also another version of this problem where transmission range of ANPs is given and the objective is to minimize the number of ANPs.

Motivated by the importance of both data collection and fault tolerance in wireless sensor networks, in this dissertation the problem of enhancing the fault tolerance capability of a data gathering tree by adding a few additional links is studied. Two fault models are considered: 1) single node failure and 2) two adjacent node failure. It is proved that the least cost tree augmentation problem is NP-complete under both types of fault scenarios. Moreover, two approximation algorithms are proposed, one for single node failure and the other for a pair of adjacent node failure, with performance bounds of two and four respectively. An important extension of this research will be as follows:

- The tree augmentation problem under more general topological region based fault models like when a fault is defined as a subgraph with diameter d.

In the domain of optical networks, the routing and spectrum allocation problem in SLICE architecture is investigated. It is proven that the RSA problem is NP-complete when the network topology is a *chain* or a *ring* and efficient approxima-

tion algorithms are designed for optical networks with ring and tree topology where the connection requests are known in advance. The on-line version of RSA problem is also studied and an algorithm for the *ring network* with a bounded *competitive ratio* is developed. In addition, heuristics for both off-line and on-line RSA problem in networks with arbitrary topology are proposed and the effectiveness of the heuristics is measured with extensive simulation. Simulation results demonstrate that our heuristics significantly outperforms several other heuristics proposed recently for the RSA problem. Moreover, in this dissertation the *Spectrum Constrained Routing and Spectrum Assignment* (SCRSA) problem is introduced. The goal of the SCRSA problem is to satisfy the largest number of requests without exceeding the available spectrum span. Approximation algorithms are designed for SCRSA in the networks with ring or binary tree topologies. The following unexplored problems can be considered as the future directions of this research problem:

- Finding out efficient approximation algorithms for RSA and SCRSA problems in networks with other specific topologies such as grid and planar topologies or even in networks with arbitrary topologies.
- Studying and analyzing the on-line version of SCRSA
- Investigating RSA problem when different modulation model can be used.

In the domain of social networks, the influence propagation in presence of active adversaries is studied. This dissertation investigates the scenario where the first player has already chosen a set of k nodes and the second player, with the knowledge of the choice of the first, attempts to identify a smallest set of nodes so that when the influence propagation process ends, the number of nodes influenced by the second player is larger than the number of nodes influenced by the first. The

identification of the smallest set of nodes to defeat the adversary is proven to be NP-Hard with two different propagation models. An approximation algorithm is proposed to solve this problem. At least two new directions of research with this problem can be envisaged:

- In the first one, the second player is not aware of the first player's choice.
- In the second one, back and forth transition of the nodes between two competing products is allowed.

REFERENCES

[1] S. Milner, S. Thakkar, K. Chandrashekar, and W. Chen, "Performance and scalability of mobile wireless base-station-oriented networks," *ACM SIGMOBILE* MC^2R , vol. 7, 2003.

[2] N. R. C. Committee on Evolution of Untethered Communications, *The Evolution of Untethered Communications.* The National Academies Press, 1997. [Online]. Available: $http://www.nap.edu/openbook.php?record\_id = 5968$

[3] J. L. Burbank, P. H. Chimento, B. K. Haberman, and W. T. Kasch, "Key Challenges of Military Tactical Networking and the Elusive Promise of MANET Technology," *IEEE Communication Magazine*, November 2006.

[4] M. Conti and S. Giardano, "Multihop ad-hoc Networking: the Reality," *IEEE Communications Magazine*, April 2007.

[5] S. Milner, J. Llorca, and C. Davis, "Autonomous reconfiguration and control in directional mobile ad hoc networks," *Circuits and Systems Magazine, IEEE*, vol. 9, no. 2, pp. 10 –26, quarter 2009.

[6] P. Gupta and P. Kumar, "The capacity of wireless networks," *IEEE Transactions on Information Theory*, vol. 46, no. 2, pp. 388 –404, mar 2000.

[7] R. Ramanathan and R. Rosales-Hain, "Topology control of multihop wireless networks using transmit power adjustment," in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2, 2000, pp. 404 –413 vol.2.

[8] J. Cabrera, R. Ramanathan, C. Gutierrez, and R. Mehra, "Stable topology control for mobile ad-hoc networks," *Communications Letters, IEEE*, vol. 11, no. 7, pp. 574 –576, july 2007.

[9] J. Wu and F. Dai, "Mobility-sensitive topology control in mobile ad hoc networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 6, pp. 522 –535, june 2006.

[10] P. Santi, "The critical transmitting range for connectivity in mobile ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 4, 2005.

[11] A. Sen, B. Shen, L. Zhou, and B. Hao, "Fault-tolerance in sensor networks: A new evaluation metric," in *Infocom*, 2006.

[12] A. Sen, S. Murthy, and S. Banerjee, "Region-based connectivity: a new paradigm for design of fault-tolerant networks," in *HPSR*, 2009.

[13] S. Neumayer, G. Zussman, R. Cohen, and E. Modiano, "Assessing the vulnerability of the fiber infrastructure to disasters," in *Infocom*, 2009.

[14] S. Neumayer and E. Modiano, "Network reliability with geographically correlated failures," in *Infocom*, 2010.

[15] K. Fall, "A delay-tolerant network architecture for challenged internets," in *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '03. New York, NY, USA: ACM, 2003, pp. 27–34. [Online]. Available: http://doi.acm.org/10.1145/863955.863960

[16] J. P. G. Sterbenz, R. Krishnan, R. R. Hain, A. W. Jackson, D. Levin, R. Ramanathan, and J. Zao, "Survivable mobile wireless networks: Issues, challenges, and research directions," in *Proceedings of the 1st ACM Workshop on Wireless Security*, ser. WiSE '02. New York, NY, USA: ACM, 2002, pp. 31–40. [Online]. Available: http://doi.acm.org/10.1145/570681.570685

[17] S. Jain, K. Fall, and R. Patra, "Routing in a delay tolerant network," in *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '04. New York, NY, USA: ACM, 2004, pp. 145–158. [Online]. Available: http://doi.acm.org/10.1145/1015467.1015484

[18] J. Alonso and K. Fall, "A linear programming formulation of flows over time with piecewise constant capacity and transit times," *Intel Research Technical Report IRB-TR-03-007*, 2003.

[19] Y. Cao and Z. Sun, "Routing in delay/disruption tolerant networks: A taxonomy, survey and challenges," *Communications Surveys Tutorials, IEEE*, vol. 15, no. 2, pp. 654–677, 2013.

[20] M. Huang, S. Chen, Y. Zhu, B. Xu, and Y. Wang, "Topology control for time-evolving and predictable delay-tolerant networks," in *2011 IEEE 8th Interna-*

tional Conference on Mobile Adhoc and Sensor Systems (MASS), 2011, pp. 82–91.

[21] M. Huang, S. Chen, Y. Zhu, and Y. Wang, "Cost-efficient topology design problem in time-evolving delay-tolerant networks," in *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, 2010, pp. 1–5.

[22] A. Tiwari, A. Ganguli, and A. Sampath, "Towards a Mission Planning Toolbox for Airborne Networks: Optimizing Ground Coverage Under Connectivity Constraints," in *IEEE Aerospace Conference*, March 2008, pp. 1–9.

[23] B. Epstein and V. Mehta, "Free Space Optical Communications Routing Performance in Highly Dynamic Airspace Environments," in *IEEE Aerospace Conference Proceedings*, 2004.

[24] R. Diestel, *Graph Theory.* Springer, 2005.

[25] J. M. H. Olmsted and C. G. Townsend, "On the Sum of Two Periodic Functions," *The Two-Year College Mathematics Journal*, vol. 3, 1972.

[26] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms.* McGraw Hill, 2001.

[27] O. Incel and B. Krishnamachari, "Enhancing the data collection rate of tree-based aggregation in wireless sensor networks," in *Secon*, 2008.

[28] X.-Y. Li, Y. Wang, and Y. Wang, "Complexity of data collection, aggregation, and selection for wireless sensor networks," *IEEE Transactions on Computers*, vol. 60, pp. 386 – 399, 2011.

[29] E. Kranakis, D. Krizanc, and E. Williams, "Directional versus omnidirectional antennas for energy consumption and k-connectivity of networks of sensors," in *Proc. of 8th International Conference on Principles of Distributed Systems*, 2004, pp. 357–368.

[30] Z. Yu, J. Teng, X. Bai, D. Xuan, and W. Jia, "Connected coverage in wireless networks with directional antennas," in *Infocom*, 2011.

[31] Y. Wang and G. Cao, "Minimizing service delay in directional sensor networks," in *Infocom*, 2011.

[32] C. A. Balanis, *Antenna Theory: Analysis and Design, 2nd ed.* Wiley, 1997.

[33] J. L. Bredin, E. D. Demaine, M. Hajiaghay, and D. Rus, "Deploying sensor networks with guaranteed capacity and fault tolerance," in *MobiHoc*, 2005.

[34] M. Hajiaghayi, N. Immorlica, and V. Mirrokni, "Power optimization in fault-tolerant topology control algorithms for wireless multi-hop networks," *IEEE/ACM Transactions on Networking*, vol. 15, pp. 1345 – 1358, 2007.

[35] F. Wang, M. Thai, Y. Li, X. Cheng, and D.-Z. Du, "Fault-tolerant topology control for all-to-one and one-to-all communication in wireles networks," *IEEE Transactions On Mobile Computing*, vol. 7, pp. 322–331, 2008.

[36] G. N. Frederickson and J. Ja'Ja', "Approximation algorithms for several graph augmentation problems," *SIAM J. on Computing*, vol. 10, pp. 270–283, 1981.

[37] S. Khuller and B. Raghavachari, "Improved approximation algorithms for uniform connectivity problems," *Algorithms*, vol. 21, pp. 434–450, 1996.

[38] M. Garey and D. Johnson, *Computers and intractability. A guide to the theory of NP-completeness.* Freeman, 1979.

[39] E. Petrank, "The hardness of approximation: Gap location," *computational complexity*, vol. 4, pp. 133–157, 1994.

[40] D. B. West, *Introduction to Graph Theory, 2nd ed.* Prentice Hall, 2001.

[41] R. E. Tarjan, "Finding optimum branchings," *Networks*, vol. 7, pp. 25–35, 1977.

[42] Cisco visual networking index: Forecast and methodology, 2011-2016. [Online]. Available: http://www.cisco.com/en/US/solutions/ collateral/ns341/ns525/ns537 /ns705/ns827/white_paper_c11-481360.pdf

[43] W. Shieh, "Ofdm for flexible high-speed optical networks," *Journal of Lightwave Technology*, vol. 29, no. 10, pp. 1560 –1577, 2011.

[44] M. Jinno, H. Takara, B. Kozicki, Y. Tsukishima, Y. Sone, and S. Matsuoka, "Spectrum-efficient and scalable elastic optical path network: architecture, benefits, and enabling technologies," *IEEE Communications Magazine*, vol. 47, no. 11, pp. 66 –73, 2009.

[45] Y. Wang, X. Cao, and Y. Pan, "A study of the routing and spectrum allocation in spectrum-sliced elastic optical path networks," in *INFOCOM*, 2011, pp. 1503–1511.

[46] M. G. Luby, "Tight bounds for dynamic storage allocation," *SIAM Journal on Discrete Mathematics*, vol. 9, no. 1, pp. 155–166, Feb. 1996.

[47] K. Christodoulopoulos, I. Tomkos, and E. A. Varvarigos, "Routing and spectrum allocation in ofdm-based optical networks with elastic bandwidth allocation," in *GLOBECOM*, 2010, pp. 1–6.

[48] A. N. Patel, P. N. Ji, J. P. Jue, and T. Wang, "Routing, wavelength assignment, and spectrum allocation in transparent flexible optical wdm (fwdm) networks," in *Photonics in Switching*, 2010.

[49] M. Jinno, B. Kozicki, H. Takara, A. Watanabe, Y. Sone, T. Tanaka, and A. Hirano, "Distance-adaptive spectrum resource allocation in spectrum-sliced elastic optical path network [topics in optical communications]," *IEEE Communications Magazine*, vol. 48, no. 8, pp. 138 –145, 2010.

[50] M. Klinkowski and K. Walkowiak, "Routing and spectrum assignment inspectrum sliced elastic optical path network," *IEEE Communications Letters*, vol. 15, no. 8, pp. 884–886, 2011.

[51] K. Christodoulopoulos, I. Tomkos, and E. Varvarigos, "Elastic bandwidth allocation in flexible ofdm-based optical networks," *Journal of Lightwave Technology*, vol. 29, no. 9, pp. 1354 –1366, 2011.

[52] ——, "Dynamic bandwidth allocation in flexible ofdm-based networks," in *OFC/NFOEC*, 2011.

[53] X. Wan, N. Hua, and X. Zheng, "Dynamic routing and spectrum assignment in spectrum-flexible transparent optical networks," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 4, no. 8, pp. 603 –613, 2012.

[54] T. Takagi, H. Hasegawa, K. Sato, Y. Sone, B. Kozicki, A. Hirano, and M. Jinno, "Dynamic routing and frequency slot assignment for elastic optical path networks that adopt distance adaptive modulation," in *OFC/NFOEC*, 2011.

[55] A. Castro, L. Velasco, M. Ruiz, M. Klinkowski, J. P. Fernández-Palacios, and D. Careglio, "Dynamic routing and spectrum (re)allocation in future flexgrid optical networks," *Compututer Networks*, vol. 56, no. 12, pp. 2869–2883, 2012.

[56] G. Shen and Q. Yang, "From coarse grid to mini-grid to gridless: How much can gridless help contentionless?" in *OFC/NFOEC*, 2011.

[57] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1990.

[58] A. L. Buchsbaum, H. Karloff, C. Kenyon, N. Reingold, and M. Thorup, "Opt versus load in dynamic storage allocation," *SIAM Journal on Computing*, vol. 33, no. 3, pp. 632–646, 2004.

[59] S. V. Pemmaraju, S. Penumatcha, and R. Raman, "Approximating interval coloring and max-coloring in chordal graphs," *Journal of Experimental Algorithmics*, vol. 10, 2005.

[60] C. Nomikos, A. Pagourtzis, and S. Zachos, "Minimizing request blocking in all-optical rings," in *INFOCOM*, vol. 2, 2003, pp. 1355–1361.

[61] Z. Bian and Q.-P. Gu, "1.5-approximation algorithm for weighted maximum routing and wavelength assignment on rings," *Information Processing Letters*, vol. 109, no. 8, pp. 400–404, 2009.

[62] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, 1980.

[63] H. A. Kierstead, "A polynomial time approximation algorithm for dynamic storage allocation," *Discrete Mathematics*, vol. 87, no. 2-3, pp. 231–237, 1991.

[64] M. C. Golumbic, M. Lipshteyn, and M. Stern, "Representing edge intersection graphs of paths on degree 4 trees," *Discrete Mathematics*, vol. 308, no. 8, pp. 1381 – 1387, 2008.

[65] T. Erlebach and K. Jansen, "The complexity of path coloring and call scheduling," *Theoretical Computer Science*, vol. 255, no. 1-2, pp. 33–50, 2001.

[66] R. Bar-yehuda, M. Beder, and Y. Cohen, "Approximation algorithms for bandwidth and storage allocation," 2005.

[67] M. Yannakakis and F. Gavril, "The maximum k-colorable subgraph problem for chordal graphs," *Information Processing Letters*, vol. 24, no. 2, pp. 133–137, 1987.

[68] Level 3 Communications, Network Map. [Online]. Available: http://nsssc.superb.net/img/l3-usmap.gif

[69] J. Y. Yen, "Finding the k shortest loopless paths in a network," *Management Science*, vol. 17, no. 11, pp. 712–716, 1971.

[70] S. Bharathi, D. Kempe, and M. Salek, "Competitive influence maximization in social networks," in *Proceedings of the 3rd international conference on Internet and network economics*, ser. WINE'07, 2007, pp. 306–311.

[71] T. Carnes, C. Nagarajan, S. M. Wild, and A. van Zuylen, "Maximizing influence in a competitive social network: a follower's perspective," in *Proceedings of the ninth international conference on Electronic commerce*, ser. ICEC '07, 2007, pp. 351–360.

[72] W. Chen, Y. Wang, and S. Yang, "Efficient influence maximization in social networks," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '09, 2009, pp. 199–208.

[73] P. Domingos and M. Richardson, "Mining the network value of customers," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '01, 2001, pp. 57–66.

[74] A. Goyal, F. Bonchi, L. V. S. Lakshmanan, and S. Venkatasubramanian, "Approximation analysis of influence spread in social networks," *arXiv:1008.2005v4*, 2011. [Online]. Available: http://arxiv.org/abs/1008.2005v4

[75] D. Kempe, J. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '03, 2003, pp. 137–146.

[76] M. Mathioudakis, F. Bonchi, C. Castillo, A. Gionis, and A. Ukkonen, "Sparsification of influence networks," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '11, 2011, pp. 529–537.

[77] S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications*, 1st ed., ser. Structural analysis in the social sciences. Cambridge University Press, 1994, no. 8.

[78] Y. Wang, G. Cong, G. Song, and K. Xie, "Community-based greedy algorithm for mining top-k influential nodes in mobile social networks," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '10, 2010, pp. 1039–1048.

[79] Q. Jiang, G. Song, C. Gao, Y. Wang, W. Si, and K. Xie, "Simulated annealing based influence maximization in social networks," 2011. [Online]. Available: http://www.aaai.org/ocs/index.php/AAAI/AAAI11/paper/view/3670

[80] S. Bhagat, A. Goyal, and L. V. Lakshmanan, "Maximizing product adoption in social networks," in *Proceedings of the fifth ACM international conference on Web search and data mining*, ser. WSDM '12, 2012, pp. 603–612.

[81] H. Li, S. S. Bhowmick, and A. Sun, "Casino: towards conformity-aware social influence analysis in online social networks," in *Proceedings of the 20th ACM international conference on Information and knowledge management*, ser. CIKM '11, 2011, pp. 1007–1012.

[82] K. Dave, R. Bhatt, and V. Varma, "Modelling action cascades in social networks," 2011. [Online]. Available: http://www.aaai.org/ocs/index.php/ICWSM/ICWSM11/paper/view/2741

[83] H. Habiba, Y. Yu, T. Y. Berger-Wolf, and J. Saia, "Finding spread blockers in dynamic networks," in *Proceedings of the Second international conference on Advances in social network mining and analysis*, ser. SNAKDD'08, 2010, pp. 55–76.

[84] G. Istrate, M. V. Marathe, and S. S. Ravi, "Adversarial models in evolutionary game dynamics," in *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, ser. SODA '01, 2001, pp. 719–720.

[85] M. A. Nowak, C. E. Tarnita, and T. Antal, "Evolutionary dynamics in structured populations," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 365, no. 1537, pp. 19–30, 2010.

[86] M. E. J. Newman, "The structure of scientific collaboration networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 98, no. 2, pp. 404–409, 2001.

[87] M. Newman, *http://networkdata.ics.uci.edu/data/hep-th/*. [Online]. Available: http://networkdata.ics.uci.edu/data/hep-th/