Design, Analysis and Computation in Wireless and Optical Networks

by

Chenyang Zhou

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

Approved November 2018 by the
Graduate Supervisory Committee:

Andrea Richa, Co-chair
Arunabha Sen, Co-chair
Guoliang Xue
Krzysztof Walkowiak

ARIZONA STATE UNIVERSITY

May 2019

ABSTRACT

In the realm of network science, many topics can be abstracted as graph problems, such as routing, connectivity enhancement, resource/frequency allocation and so on. Though most of them are NP-hard to solve, heuristics as well as approximation algorithms are proposed to achieve reasonably good results. Accordingly, this dissertation studies graph related problems encountered in real applications. Two problems studied in this dissertation are derived from wireless network, two more problems studied are under scenarios of FIWI and optical network, one more problem is in Radio-Frequency Identification (RFID) domain and the last problem is inspired by satellite deployment.

The objective of most of relay nodes placement problems, is to place the fewest number of relay nodes in the deployment area so that the network, formed by the sensors and the relay nodes, is connected. Under the fixed budget scenario, the expense involved in procuring the minimum number of relay nodes to make the network connected, may exceed the budget. In this dissertation, we study a family of problems whose goal is to design a network with “maximal connectedness” or “minimal disconnectedness”, subject to a fixed budget constraint. Apart from “connectivity”, we also study relay node problem in which degree constraint is considered. The balance of reducing the degree of the network while maximizing communication forms the basis of our d-degree minimum arrangement(d-MA) problem. In this dissertation, we look at several approaches to solving the generalized d-MA problem where we embed a graph onto a subgraph of a given degree.

In recent years, considerable research has been conducted on optical and FIWI networks. Utilizing a recently proposed concept “candidate trees” in optical network, this dissertation studies counting problem on complete graphs. Closed form expressions are given for certain cases and a polynomial counting algorithm for general cases is also presented. Routing plays a major role in FiWi networks. Accordingly to a novel path length metric which emphasizes on "heaviest edge", this dissertation proposes a polynomial algorithm on single path computation. NP-completeness proof as well as approximation algorithm are presented for multi-path routing.

Radio-frequency identification (RFID) technology is extensively used at present for identification and tracking of a multitude of objects. In many configurations, simultaneous activation of two readers may cause a "reader collision" when tags are present in the intersection of the sensing ranges of both readers. This dissertation addresses slotted time access for Readers and tries to provide a collision-free scheduling scheme while minimizing total reading time.

Finally, this dissertation studies a monitoring problem on the surface of the earth for significant environmental, social/political and extreme events using satellites as sensors. It is assumed that the impact of a significant event spills into neighboring regions and there will be corresponding indicators. Careful deployment of sensors, utilizing "Identifying Codes", can ensure that even though the number of deployed sensors is fewer than the number of regions, it may be possible to uniquely identify the region where the event has taken place.

*To the loving memory of my grandmother, **Xingzhen Zhang**,*

who is blessing from heaven

ACKNOWLEDGMENTS

This dissertation summaries my Ph.D. career and presents a portion of my work carried out at Arizona State University's. This dissertation could never have been done without the all the help, support and encouragement of the people who I wish to acknowledge.

First and foremost, I would like to express my sincerest and deepest appreciation and gratitude to my advisers, Prof. Arunabha Sen and Prof. Andrea Richa for their patience and continuous faith in me. I am always impressed by their enthusiasm on research and teaching. Much of my research results were inspired and motivated by Prof. Sen and Prof. Richa. I do appreciate any guidance and encouragement from them. This dissertation would not have been possible without their help.

Secondly, I am also extremely grateful to my dissertation committee members Prof. Guoliang Xue and Prof. Krzysztof Walkowiak for insightful feedback towards my research. I took Prof Xue's class at ASU and learnt much useful knowledge about optimization. Prof Walkowiak gave me valuable instructions on experiment design and helped me finalize the dissertation.

I would like to thank my lab mates - Xinhui Hu, Jin Zhang, Mengxue Liu, Arun Das, Kaustav Basu, Zahra Derakhshandeh, Joydeep Banerjee and Anisha Mazumder. It is indeed my honor to have opportunities to work with these brilliant people solving hard research problems. Brickyard is such a lovely place to me because of them.

I would like to thank my friends at ASU and Tempe - Andres Mora, Yongming Zhang, Zheng Chang, Liang Xu, Cheng pan and all who companied me during my time in Arizona. My life would not have been such enriching nor fulfilling without all the joy and encouragement from my lovely friends.

I would like to show my deepest love to my girlfriend - Lingyan Li. We knew each other from high school and fell in love short after we went to college. We were at different cities during undergraduate study and I came to USA along first. Lingyan never complained and is always supportive to any decision I make. Knowing her is the luckiest thing I have ever had. And every moment I spent with her is the most treasurable memory in my life.

Finally, I would like to thank my family - my dad, mom, grandpa, grandma and my twin cousins for their unweaving support. Ph.D. is a long journey, it is my family that gives me the strongest support and always backs me up.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

Chapter 1

INTRODUCTION

Many applications in real world can be formed as networks. Modern technologies ranging from wireless communication to Internet, social networks to satellites systems - all are networked. Graph theory is a powerful tool to analysis and solve abstract models and our daily life benefits from effort of computer science researchers. Although extensive research has been done on topics like routing, resource allocation, connectivity enhancement and so on. There are still a lot of more new areas we never explored. This requires us to have a in-depth understanding of nature of networks and motivates us a long march to unstudied problems. This dissertation focuses on graph theoretical problems in network design, more specially, we focus on wireless network on two-dimensional plane and optical FiWi network.

The two-dimensional plane is the most essential and commonly used model in wireless network. Especially for sensor networks, real applications are often under two dimensional scenarios. Due to low power of sensors, transmission restriction is always considered and therefore many research has focused on sensor/relay nodes placement or communication protocols. However, to best of our knowledge, few of them took account for budget limitation. In real world, budget is in fact the priority and we usually conduct work under limited resource. In this dissertation, we study a very common but easily ignored problem: how to place relay nodes given the location of sensors under certain budget. This problem is attractive not only because it involves graph connectivity, but also because we need to make tradeoff towards our objective since not all requirement can be fulfilled. More specially, we point out 3 possible objectives and give approximation or inapproximatebility for each one.

Apart from connectivity, degree of graph is also a consideration when budget or computation power is limited. For instance, a router which serves as relay node usually has connection limitation. If the number of device connected to the router exceeds certain value. The quality of communication is very likely to be affected. Therefore, it is of our interest to consider the following problem: given a network/graph, how to choose subset of edges such that the degree of every node is no larger than a given integer d while some objective, like total distance or total weight, is minimized. When $d = 2$, it becomes the classic Minimum Linear Arrangement(MLA) problem. Though extensive research has been done on MLA, there have been no developments into versions of the problem involving degree higher than 2. In this dissertation, we will look at several approaches to solve the generalized d-degree minimum arrangement (d-MA) problem where we embed a graph onto a subgraph of a given degree.

In recent years, Elastic Optical Networks (EON) attracts a large amount of attention of research. One of key challenges that appears in EON study is the need to distribute large volumes of data to many users at the same time in an efficient and cost-effective way. Due to recent research, it is found that one of the most efficient approach for optimizing multicast flows is based on using pre-generated "candidate tree", where each tree represents a steiner tree given all source and destination terminals while other nodes are optional. Clearly, we should handle "candidate tree" generation problem. A very nature question would be "how many candidate trees can be generated?", therefore in this dissertation, we study the case where the topology is a complete graph with n nodes and p of them are target nodes. We gave an counting algorithm for general cases, and show closed form expression when $p = 2$ and 3. Moreover, we provide an steiner tree generating algorithm.

Though significant progress has been made on the design of advanced FiWi network architectures as well as access techniques and routing protocols/algorithms over the last few years. It is still challenging how to design routing algorithms for the wireless front end only or for both the wireless and optical domains of FiWi access networks. A large number of wireless, integrated opticalwireless, multipath, and energy-aware routing algorithms were proposed. Among them, Frank Aurzada etc invented a new unified analytical framework to allow capacity and delay evaluations of a wide range of FiWi network routing algorithms and provide important design guidelines for novel FiWi network routing algorithms that leverage the different unique characteristics of disparate optical fiber and wireless technologies [3]. In their work, they considered a new path length metric involving "heaviest edge" and achieved good result in simulation. To better facilitate this novel metric, in this dissertation, we gave the solution to compute shortest path under new metric and study multi-path problem which is not presented previously.

Apart from exploring problems in wireless and optical network, this dissertation also explores resource allocation techniques in cyber-physical systems, specifically in Radio-Frequency IDentification (RFID) systems. RFID systems extensively use *readers* and *tags* for identification of objects with unique identifiers and are required to allow readers fast and accurate access to tags available in the environment. Reader and Tag type devices are utilized in the Radio-Frequency IDentification technology for identification and tracking of objects. A tag can be read by a reader when the tag is within the readers sensing range. However, when tags are present in the intersection area of the sensing ranges of two or more readers, simultaneous activation of the readers may cause reader collision. In order to ensure collision-free reading, a scheduling scheme is needed to read tags in the shortest possible time. We study this scheduling problem in a stationary setting and the reader minimization problem in a mobile setting. We show that the optimal schedule construction problem is NP-complete and provide a heuristic algorithm that we evaluate our techniques through simulation.

For application of identification code on events monitoring, as Earth is almost a sphere, we use a soccer ball (a sphere) as a model. From the model, we construct a *Soccer Ball Graph* (SBG), and show that the SBG has at least 26 sets of Identifying Codes of cardinality ten, implying that there are at least 26 different ways to deploy ten satellites to monitor the Earth. Finally, we also show that the size of the minimum Identifying Code for the SBG is *at least* nine.

The remainder of this dissertation dissertation is organized as follows: Chapter 2 presents the budget constraint problem in relay node placement with approximation algorithm. In Chapter 4, a brief overview of the concept “candidate trees” is outlined and some computation results are given. The (d-MA) problem is studied in Chapter 3. The path computation problem in FiWi networks using new metric is addressed in 5. Reader Scheduling in RFID system is discussed in Chapter 6 and identification code monitoring problem is address in Chapter 7. Finally, in Chapter 8 this dissertation is concluded, and possible extensions of the presented work is outlined.

Chapter 2

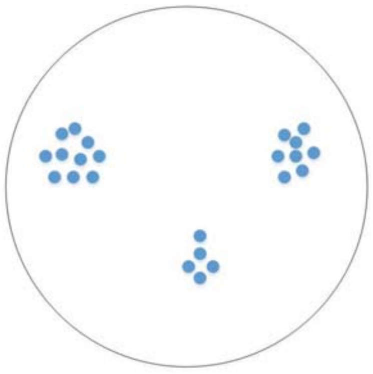
RELAY NODE PLACEMENT UNDER BUDGET CONSTRAINT

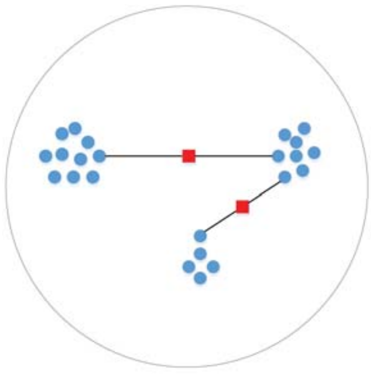
2.1 Introduction

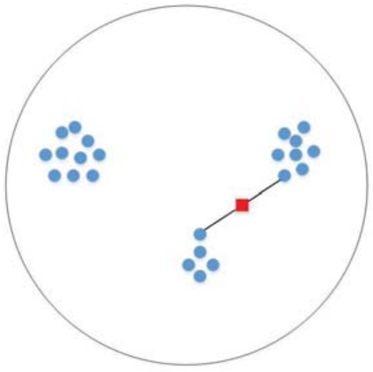
The relay node placement problem, because of its importance in wireless sensor networks, has been studied fairly extensively in the last few years [40, 30, 36, 51, 62, 23, 44, 5, 37, 32, 41]. The problem has been studied in several different scenarios. In one scenario, a number of sensors (nodes) have been placed in a deployment area and the objective is to place the fewest number of relay nodes in the deployment area such that (i) each sensor node is within the communication range of at least one relay node and (ii) the network formed by the relay nodes is connected. This is a *two tiered network model* where the relay nodes serve as cluster heads (or higher tier nodes) to form a connected network topology for delivery of data collected by the sensors (lower tier nodes). Relay node placement problems have also been studied with a single tier network model, where each sensor node is not required to be in direct contact with a relay node, as they have the capability of forwarding packets received from other sensor nodes. In a single tier network model, data collected at a sensor node is delivered to the data collection point by multiple hops through other sensor and relay nodes. In this chapter, we focus our attention to the single tier network model where a set of sensor nodes have already been placed in the deployment area, and the goal is to place *at most* a specified number of relay nodes to realize a certain objective. The objective most often is to place as few relay nodes in the deployment area as possible so that the resulting network comprising of sensor and relay nodes is *connected*. As the deployment of relay nodes involves *cost*, it may not be possible to acquire and deploy the number relay nodes necessary to make the entire network connected, particularly when one has to operate under a fixed budget. Although in this scenario, one has to give up the idea of having a network *connecting all the sensor nodes*, one would still like to have a network with high level of "*connectedness*". In a recent paper [43], we introduced the notion of "*connectedness*" in a *disconnected graph* and provided two *metrics* to measure it. The first metric to measure *connectedness* of a disconnected graph presented in [43] is the *size of the largest connected component* of the graph. We argue that a *larger size of the largest connected component* in a disconnected graph is an indicator of a *higher degree of connectedness* of the graph.The second metric to measure *connectedness* of a disconnected graph is the *number of connected components* of the graph. In a manner similar to the size of the largest connected component, we argue that a *lower number of connected components* in a disconnected graph is also an indicator of a *higher degree of connectedness* of the graph.

The problem scenario studied in this chapter is depicted diagrammatically in Fig 2.1. By *communication range*, we refer to the upper bound on transmission range.

Consider a set of twenty three sensor nodes (shown as blue circles) deployed as shown in Fig. 2.1a. Since the mathematical abstraction of the relay node placement problem corresponds to the *Geometric Steiner Tree Problem* [39], and the terms *Steiner Points* and *terminal points* are used in the abstraction, where the Steiner Points and terminal points correspond to the locations of the relay and sensor nodes respectively. In this chapter we have used the terms "sensor nodes" and "terminal points" interchangeably. In Fig. 2.1a there are three clusters - the first one with ten terminal points, the second one with eight, while the third with five. The intra-


(a) Deployment of terminal points


(b) Optimal solution for Budget of 2


(c) Optimal solution for objective 1, but not for objective 2

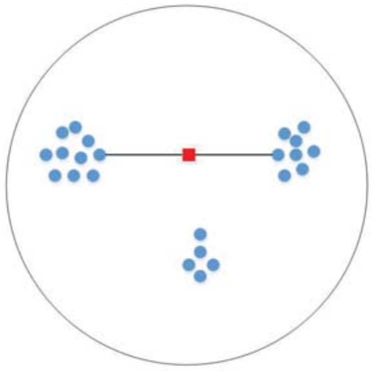

(d) Optimal solution for both objectives 1 and 2

Figure 2.1: Figure showing variation in placing relay nodes for different objectives and budget constraints

cluster distances are within the communication range, whereas the inter-cluster ones are not. Suppose that the maximum inter cluster distance is less than twice the communication range, and as such only one relay node is sufficient for connecting any two clusters. If we have the option of placing two relay nodes (shown as red squares), then under both metrics of connectedness, the placement of relay nodes as shown in Fig. 2.1b is an optimal solution. However, if we have a budget for only one relay node, the solution shown in Fig. 2.1c is an optimal solution under budget constraint according to the first metric of connectedness. This is true as there are exactly two connected components which is the best that can be achieved with only one relay node. However, in this solution, the largest connected component has only thirteen nodes and is not optimal according to the second metric. Fig. 2.1d shows the placement of the relay node which is optimal under budget constraint for the second metric, having the largest connected component with eighteen terminal points. It may be noted that this placement also results in an optimal solution under budget constraint according to the first metric.

Expanding on the concept of "*connectedness of a disconnected graph*" introduced in [43], this chapter introduces two additional *metrics* to measure *"connectedness"* of a disconnected graph. The third metric is the *size of the smallest connected component* of the graph. We argue that a *larger size of the smallest connected component* can also be viewed as an indicator of a *higher degree of connectedness* of the graph. It may be noted that two important attributes of a disconnected graph are the *size* and the *number* of components. Among the three metrics, the first and the third takes into account only to the *size* (largest and smallest) and the second takes into account only the *number* of components. As one can argue that a metric to measure connectedness of a disconnected graph that pays attention only to the size or to the number of connected components is *myopic* and a non-myopic metric to measure connectedness of a disconnected graph should take into account, both the size and the number of connected components simultaneously. To address this issue, in section 2.7 we propose a fourth metric to measure the "connectedness" of a disconnected graph that takes into account both the size as well as the number of connected components.

In this chapter, we first show that the network design problems whose goal is to achieve maximal "connectedness" in a disconnected graph using the first three metrics are NP-complete. We present (i) an approximation algorithm with a performance bound of $\frac{1}{10}$ for the first metric and (ii) inapproximability results for the second and the third metrics. In addition, we present future direction of our research on this topic. Although resource constrained version of relay node placement problems have been studied in literature [44, 5, 37], to the best of our knowledge, a formal treatment of the study of the "connectedness" of a disconnected graph has not been undertaken earlier.

2.2 Related Works

The relay node placement problem in wireless sensor networks has been studied extensively in the last few years [40, 36, 51, 62, 44, 5, 37, 39]. Most of the studies can be categorized in the following way: (i) single-tiered network versus two-tiered network [39, 45], (ii) 1-connected network versus k-connected network ($k \geq 2$) [39, 27], (iii) homogeneous transmission range of nodes versus heterogeneous transmission range of nodes [39, 40, 23], (iv) location (for placement of relay nodes) unconstrained problem versus location constrained problem [39, 44].

Lin and Xue introduced the Steiner Minimum Tree with Minimum Number of Steiner Points and Bounded Edge Length problem (SMT-MSPBEL) in [39]. This problem is exactly the same as the placement of the fewest number of relay nodes in the deployment area, so that the network formed by the sensor and the relay nodes are connected. They proved that the SMT-MSPBEL problem is NP-hard and presented an approximation algorithm with a performance bound of 5. Later Chen et al. in [10] proved that the Lin-Xue algorithm is actually a 4-approximation algorithm. In addition, they also presented a 3-approximation algorithm for this problem. Cheng et. al. in [11] presented another 3-approximation algorithm for the with faster execution time.

The relay node placement problem studied in [39] was conducted with respect to a single-tiered network. Pan et al. in [45] studied a two-tiered network model where the sensor nodes are grouped into clusters, where each sensor node in a cluster is within the communication range of it's clusterhead, which is a relay node. In this model, the sensor nodes transmit sensed data to the clusterhead, which relays it to the data collection point, with multiple hops through other relay nodes. In recent times, several authors have focused their attention on study of both single tier and two tier networks [62, 40].

The relay node placement problem studied in [39] had the goal of making the network formed by the sensors and relay nodes connected (i.e., 1-connected). Bredin et. al. in [27] generalized the relay node placement problem with a goal of designing a k-connected network where $k > 1$. They presented polynomial time O(1)-approximation algorithm for the problem. For a special case, when $k = 2$, Kashyap et al. presented a 10-approximation algorithm in [1].

The transmission range of sensor and relay nodes were assumed to be homogeneous (identical) in [39]. Follow up research on this topic introduced heterogeneity of transmission range at two levels. The first level of heterogeneity was introduced in [40] where transmission range of the sensor and the relay nodes were different. They proved its NP-hardness of the problem and presented a 7-approximation algorithm for the case where connectivity $k = 1$. Zhang et al. in [62] presented a 14-approximation algorithm for case where $k = 2$. The second level of heterogeneity was introduced in [23] where different sensor nodes were allowed to have different transmission ranges. They present approximation algorithms for scenarios where unidirectional or bidirectional paths connect the sensor and relay nodes.

The relay node placement problem, studied in [39] can be viewed as unconstrained location problem in the sense that there were no constraints on the locations where the relay nodes can be placed. In the constrained version of the problem studied in

[44], the relay nodes can only be placed in the pre-defined set of candidate locations. In this setting they study two problems. The goal of these two problems is to deploy the minimum number of relay nodes to ensure that each sensor node is connected through a bidirectional path to one/two base stations respectively. The analyze the computational complexity of the problems and present a framework of polynomial time $\mathcal{O}(1)$-approximation algorithms.
The authors in [5] considered a *deployment budget* and studied the tradeoff between the network throughput, the deployment budget, and overall system coverage. The issues arising out of relay node deployment budget was also considered by Li et. al. in [37]. They study the problem of deployment of road side units (RSUs) to improve overall network performance in a vehicular ad-hoc network environment. The objective of their study is to find optimal locations for RSUs that maximizes the number of vehicles that can receive message from the RSUs, subject to the budget and delay constraints.
Although resource constrained version of relay node placement problems has been studied in literature [44, 5, 37], to the best of our knowledge, aside from our own investigation in [43], the study of connectedness of a disconnected graph has not be undertaken earlier.

2.3 Problem Formulation

As discussed earlier, the goal of this study is to design sensor networks with a high degree of *connectedness* even when operating in an environment where the number of available relay nodes is less than the number of relay nodes necessary to make all the sensor nodes connected. As a first step in this direction, we formalize the notion of *connectedness* in three different ways. Accordingly, we have three well defined problems and formal statements of these three problems are provided below. The input to the sensor network design problem is: (i) the locations of a set of sensor nodes (terminal points) $P = \{p_1, p_2, \ldots, p_n\}$ in the Euclidean plane, (ii) the communication range R of the sensor nodes, and (iii) a budget B on the number of relay nodes that can be placed in the deployment area. From the set of points P and communication range R, we construct a graph $G = (V, E)$ in the following way: Corresponding to each point $p_i \in P$ we create a node $v_i \in V$ and two nodes v_i and v_j have an edge $e_{i,j} \in E$ if the distance between the corresponding points p_i and p_j is at most R. It may be noted that the graph $G = (V, E)$ so constructed may be *disconnected* (i.e., it might comprise of a number of *connected components*). The purpose of deploying the relay nodes is to make the *augmented graph*, $G' = (V', E')$, (comprising of sensor and relay nodes) *connected*. Suppose that m relay nodes are deployed at points $Q = \{q_1, q_2, \ldots, q_m\}$. Corresponding to every point $q_i \in Q$ there is a node $v_i \in V' - V$ and there is an edge between two nodes v_i and v_j in V' if the distance between the corresponding points is at most R. With unlimited budget B, sufficient number of relay nodes can be deployed to ensure that the graph $G' = (V', E')$ is connected. However, if the budget is smaller than the minimum number of relay nodes necessary to make the graph $G' = (V', E')$ connected, this goal will be unachievable. However, in this scenario also, one would like to have the graph $G' = (V', E')$ *as much connected as possible.* This gives rise to a *connectedness* maximization problem. The goal of creating the graph $G' = (V', E')$ with *maximal connectedness or least disconnectedness* can be achieved by (i) deploying the relay nodes in a fashion that *maximizes the size of the largest connected components of* $G' = (V', E')$, or (ii) deploying the relay nodes in a fashion that *maximizes the size of the smallest connected components of* $G' = (V', E')$, or (iii) deploying the relay nodes in a fashion that *minimizes the number of connected* *components of* $G' = (V', E')$. We refer to (i) as *Budget Constrained Relay node Placement for Maximizing the size of the Largest Connected Component* (BCRP-MLCC) problem, (ii) as *Budget Constrained Relay node Placement for Maximizing the size of the Smallest Connected Component* (BCRP-MSCC) problem and (iii) as *Budget Constrained Relay node Placement for Minimizing the Number of Connected Components* (BCRP-MNCC) problem.

Next, we provide formal definitions of the three problems:

Given the locations of n sensor nodes in the Euclidean plane $P = \{p_1, p_2, \ldots, p_n\}$, transmission range R and a budget B on the number of relay nodes that can be deployed, is it possible to find a set of points $Q = \{q_1, q_2, \ldots, q_m\}$, where $m \leq B$, in the same plane where relay nodes can be deployed, so that:

(i) *BCRP-MLCC Problem:* the *size of the largest connected component* in the graph $G' = (V', E')$ corresponding to the point set P and Q is at least X, for a pre-specified value X? (The graph construction rule from the point set P and Q is described earlier in the section).

(ii) *BCRP-MSCC Problem*: the *size of the smallest connected component*, of the graph $G' = (V', E')$ corresponding to the point set P and Q is at least Y, for a pre-specified value Y?

(iii) *BCRP-MNCC Problem*: the *number of connected components* in the graph $G' = (V', E')$ corresponding to the point set P and Q is at most Z, for a pre-specified value Z?

2.4 Problem Solution

The unconstrained version of the relay node placement problem is known as the *Steiner Tree Problem with Minimum Number of Steiner Points with Bounded Edge*

Length (STP-MSPBEL) and was studied in [39].

STP-MSPBEL Problem: Given a set of n terminals points (location of sensor nodes) $P = \{p_1, p_2, ..., p_n\}$ in the Euclidean plane, and positive constants R and B, is there a tree T spanning a point set $Q \supseteq P$ such that each edge in the tree has a length no greater than R and the number of points in $Q \setminus P$, called Steiner points is at most B?

The authors in [39] have shown that the STP-MSPBEL is NP-complete. As the STP-MSBEL problem is a special case of both BCRP-MNCC and BCRP-MLCC problems, and STP-MSBEL is NP-complete, we can conclude that all three problems we study are NP-complete.

2.4.1 BCRP-MLCC

In this subsection, we first present an approximation algorithm for BCRP-MLCC with a constant factor $\frac{1}{10}$ performance guarantee. As our approximation algorithm is based on one approximation algorithm for the K Minimum Spanning Tree (K-MST) problem, first we describe the K-MST problem.

Minimum K-Spanning Tree Problem (K-MST) : Given a graph $G = (V, E)$, real numbers K, B, and a weight function $w : E \rightarrow \mathbb{N}$, is there a spanning tree T of G of at least K nodes, so that $\sum_{e \in T} w(e) \leq B$.

The K-MST problem is NP-hard [18]. The approximation algorithm for the K-MST problem provided by Garg in [18] guarantees a performance bound of 2. We will present this algorithm and make use of it for developing an approximation algorithm for BCRP-MLCC problem. It may be recalled that an instance of the BCRP-MLCC is specified by the locations of a set of sensor nodes (terminal points)

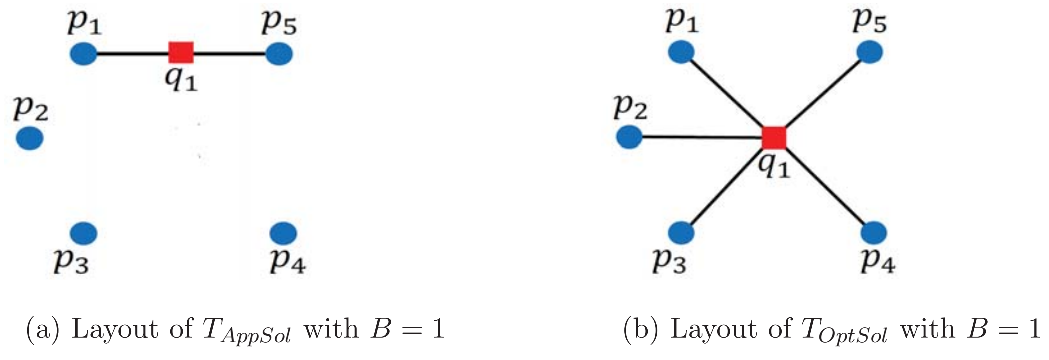
$P = \{p_1, p_2, \ldots, p_n\}$ in the Euclidean plane. Consider a complete weighted graph $G = (V, E)$ constructed from the set of points P where each node $v_i \in V$ corresponds to a point $p_i \in P$ and weight on the edge $w(v_i, v_j)$ between the nodes v_i and v_j is set equal to $(\lceil d(p_i, p_j)/R \rceil - 1)$, where $d(p_i, p_j)$ is the Euclidean distance between the points p_i and p_j and R is the transmission range of the sensors. We then apply the following algorithm on the constructed graph $G = (V, E), |V| = n$.

Algorithm 1 Approximation Algorithm for BCRP-MLCC

```
1: for $i = n$ to 1 do
2:     Set $K = i$ and apply Garg's algorithm to find the spanning tree $T$
3:     if $weight(T) \leq B$ then return $T$
4:     end if
5: end for
```

We denote P' as the vertex set from resulting tree T and $AppSol = |P'|$. Each node of P' corresponds to a point $p_i \in P$, and very naturally we have $P' \subset P$. An edge in T connecting nodes p_i and p_j corresponds to an *straight line segment* I_{p_i,p_j} between points p_i and p_j. As our goal is to have a deployment of relay nodes, we then show how to place them on T. Denote all relay nodes as Q'. We want to insert relay nodes on edges of T so that every edge is subdivided into small pieces. In more detail, for any straight segment $I_{u,v}$, relay nodes are placed at distance R (transmission range) apart starting from one, w.l.o.g. say u, to the other one v. It is easily verifiable that every edge needs $\lceil d(p_i, p_j)/R \rceil - 1$ relay nodes. As required, $|Q'| \leq B$, where B is the budget for the number of relay nodes. Thus, we obtain a new tree T_{AppSol} formed by node set $P' \cup Q'$ and layout of T_{AppSol} is our desired connected component.

We will refer to one optimal solution of the BCRP-MLCC problem as T_{OptSol} which is also a tree-style layout after deployment of relay nods. Similarly, T_{OptSol} can be viewed as point set $P'' \cup Q''$, where $P'' \subset P$ and $OptSol = |P''| \geq |P'|$. In addition, budget constraint is satisfied, i.e., $|Q''| \leq B$. Without loss of generality, we can assume that T_{OptSol} is the optimal solution which uses the *fewest* number of relay nodes. The following lemma was established in [39].



(a) Layout of T_{AppSol} with $B = 1$ (b) Layout of T_{OptSol} with $B = 1$

Figure 2.2: Example of layout of T_{AppSol} and T_{OptSol}.

Lemma 1. *There exists a shortest length optimal Steiner tree for STP-MSPBEL such that every Steiner point has degree at most five.*

Proof. Please refer to [39] for proof. □

The Lemma States that the optimal solution to STP-MSBEL, denoted as **OptST-P**, has a layout on a two dimensional plane such that every Steiner point has degree at most five. Very naturally, it is true for any connected component if the graph cannot be connected. Since we consider the largest connect component, according to Lemma 1, the tree T_{OptSol} formed by $P'' \cup Q''$ also has a layout in two dimensional plane such that no $q_i \in Q''$ has degree greater than five. Layout of such a tree is shown in Figure 3.

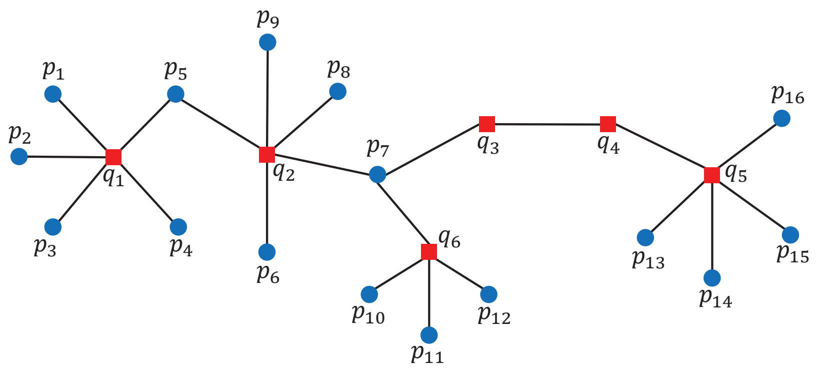


Figure 2.3: Layout of T_{OptSol} whose relay nodes have degree no more than 5

Definition: *Generalized Depth First Search* (GDFS): This is almost the same as Depth First Search [12], we maintain a sequence of the order when a node is traversed. However, a node is again enumerated when coming back from one of its neighbors and going to the next one. It implies a node may appear multiple times in the final resulting sequence.

One possible sequence generated by the GDFS on T_{OptSol} layout shown in Figure 2.3 starting from the p_1 is as following: $p_1, q_1, p_2, q_1, p_3, q_1, p_4, q_1, p_5, q_2, p_6, q_2, p_7, q_6, p_{10}, q_6, p_{11}, q_6, p_{12}, q_6, p_7, q_3, q_4, q_5, p_{13}, q_5, p_{14}, q_5, p_{15}, q_5, p_{16}, q_5, q_4, q_3, p_7, q_2, p_8, q_2, p_9, q_2, p_5, q_1, p_1$. Some nodes appear multiple times because it is still counted when going backwards.

W.l.o.g, we refer terminal points as $p-type$ points and relay nodes as $q-type$ points.

Lemma 2. *The number of occurrences of any $q-type$ point in the sequence produced by GDFS on T_{OptSol} is at most five.*

Proof. In Lemma 1 it was established that every Steiner point has degree at most five. Since the $q - type$ points are equivalent to Steiner points, by the property of DFS, we will not encounter such nodes more than 5 times, hence the claim follows. □

Definition: *Logical Hamiltonian Path*: Given a connected graph $G = (V, E)$, a Logical Hamiltonian Path, $LHP = v_1, v_2, ..., v_n$ is a permutation of vertices, such that $e(v_i, v_{i+1})$ is an edge of LHP and corresponds to a path from v_i to v_{i+1} in G.

Given a T_{OptSol} and its layout, define $cost(P)$ be the number of relay nodes on P if P is a path in T_{OptSol}. Then $cost(LHP) = \sum_{e \in LHP} cost(e)$. We have the following lemma:

Lemma 3. *If T_{OptSol} contains B relay nodes, there is an LHP s.t. $cost(LHP) \leq 5B$ and we can construct a corresponding graph C^* with at most $5B$ relay nodes while using straight segments between terminal points.*

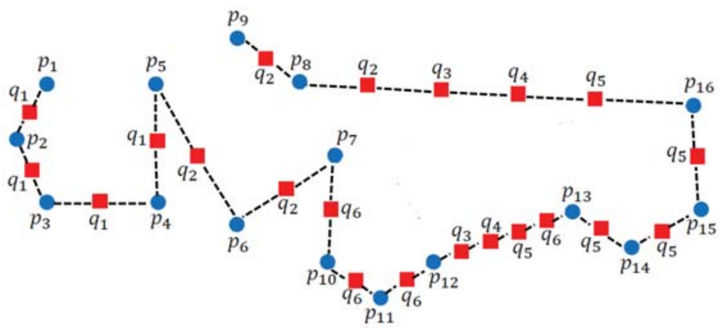
Proof. Without loss of generality, we say there are n terminal points in T_{OptSol}.

First we do GDFS on T_{OptSol} and let seq be the corresponding sequence. We construct a $LHP = \{p_1^*, p_2^*, ..., p_n^*\}$ of T_{OptSol} such that p_i^* is a terminal point and p_i^*'s order is determined by its first appearance in seq, i.e., for $i < j$, p_i^*'s first appearance is ahead of p_j^*'s. One possible LHP of Figure 3 could be $\{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_{10}, p_{11}, p_{12}, p_{13}, p_{14}, p_{15}, p_{16}, p_8, p_9\}$. We say two nodes (u, v) are directly connected, if u can reach v via only relay nodes in T_{OptSol}. Otherwise, (u, v) are indirectly connected. As an example in Figure 3, (p_1, p_2) are directly connected while (p_{16}, p_8) are indirectly connected. We then construct desired C^* from LHP.

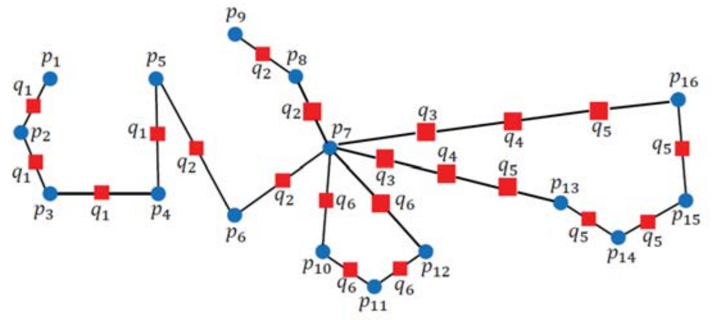
Let $P_{u,v}$ be the path from u to v on T_{OptSol}, and $I_{u,v}$ be the straight segment between u and v. Since we study 2-dimensional scenario, straight segment is the shortest distance between any two points, $I_{u,v}$ uses fewest relay nodes to connect (u, v), i.e., $\lceil d(u,v) \rceil - 1 \leq cost(P_{u,v})$. Hence, if (p_i^*, p_{i+1}^*) are directly connected, we add $I_{p_i^*,p_{i+1}^*}$ to C^*. Otherwise, (p_i^*, p_{i+1}^*) are indirectly connected. We may assume $P_{p_i^*,p_{i+1}^*} = \{p_i^*, x_1, x_2, ...x_k, p_{i+1}^*\}$, where x_i are other terminal points and their order is obtained from *seq*. For example, if we consider P_{p_{16},p_8}, then it is $\{p_{16}, p_7, p_8\}$. By property of DFS, every two consecutive terminals are directly connected. From above argument, we have $cost(P_{p_i^*,p_{i+1}^*}) = cost(P_{p_i^*,x_1}) + \sum cost(P_{x_i,x_{i+1}}) + cost(P_{x_k,p_{i+1}^*}) \geq \lceil d(p_i^*, x_1) \rceil - 1 + \sum(\lceil d(x_i, x_{i+1}) \rceil - 1) + \lceil d(x_k, p_{i+1}^*) \rceil - 1$. Clearly, we add $I_{p^*i,x_1}, I_{x_k,p_{i+1}^*}$ and all $I_{x_i,x_{i+1}}$ to C^* in this case. Here we should notice, by replacing path with straight segments, the total length may increase, however, we use other terminal points as intermediate nodes and they do not count as budget. In addition, though not explicitly mentioned, we always use $P_{p_i^*,p_{i+1}^*}$ as edge $e(p_i^*, p_{i+1}^*)$ of LHP. We call LHP "logical" because it is not exactly a hamiltonian path though it traverses every node at least once.

Following the two rules above until p_n^* is considered, we eventually obtain C^*. It is easy to check, we would not use more relay nodes than those appearing in *seq*. By lemma 2, we claim relay nodes required in C^* is no more than $cost(LHP)$ and $cost(LHP) \leq 5B$. □

Figure 2.4 shows one desired LHP and corresponding C^* from T_{OptSol} in figure 2.3. We use dash lines in 2.4a since LHP is not a real path and "edges" only stand for logical order and cost. While in 2.4b, edges are solid lines because they represent real intervals. We note here, we use the same label of relay nodes from T_{OptSol} such that it gives us a direct impression how each edge is constructed and how relay nodes repeat. For example, (p_1, p_2) are directly connected via q_1. Therefore, it does not take more than one node linking p_1 and p_2 using straight interval. And we draw a q_1 on I_{p_1,p_2} to indicate that. Similarly, (p_5, p_6) are directly connected via q_2. Hence we



(a) Structure of LHP



(b) Layout of C^*

Figure 2.4: Examples of LHP and C^*

draw q_2 on I_{p_5,p_6} to show another "propagation" of q_2. On the other hand, (p_{12}, p_{13}) are indirectly connected, thus we draw two segments I_{p_{12},p_7} and $I_{p_7,p_{13}}$ in C^* with repetition of relay nodes. While reflecting in 2.4a, p_{12} has a logical edge to p_{13} using at most 4 relay noes. One may notice that C^* is not necessary a tree, but it must contain a spanning tree.

In the following we give an algorithm that finds a subpath of the logical path LHP which contains at least $\frac{n}{10}$ number of $p - type$ points and costs at most $\frac{B}{2}$. It may be recalled that LHP is a permutation of n terminal points with costs at most $5B$. Again, without loss of generality, we assume that the p points on the path LHP are numbered sequentially from p_1^* to p_n^*.

Algorithm 2 Algorithm to find a subpath, P', of LHP that contains at least $\frac{n}{10}$ p points and costs at most $\frac{B}{2}$.

```
1: $S = \emptyset$ ($S$ is a set of $p$ points that will be in the subpath)
2: $i = 1$
3: $Total\_Cost = 0$
4: for $i = 1$ to $n - 1$ do
5:     $S = S \cup \{p_i\}$
6:     if $|S| \geq \frac{n}{10}$ then return $S$
7:     end if
8:     if $Total\_Cost + cost(e(p_i^*, p_{i+1}^*)) \leq \frac{B}{2}$ then
9:         $Total\_Cost = Total\_Cost + cost(e(p_i^*, p_{i+1}^*))$
10:    else
11:        $S = \emptyset$
12:        $Total\_Cost = 0$
13:    end if
14: end for
15: $S = S \cup \{p_n^*\}$ return $S$
```

Lemma 4. *There exists a subpath, P', of LHP, that includes at least $\frac{n}{10}$ $p - type$ points and whose cost is at most $\frac{B}{2}$ (n is the total number of p points).*

Proof. Consider Algorithm 2 below. The algorithm attempts to construct such subpath P' by sequentially scanning terminal points $\{p_1^*, \ldots, p_n^*\}$ on the logical path LHP, by adding one point at a time to the set S, starting from the point p_1^*. If at any point of time, it finds a set S such that $|S| \geq \frac{n}{10}$ and the cost of current set S is at most $\frac{B}{2}$, it returns that set and the algorithm terminates. On the other hand, if current cost exceeds threshold $\frac{B}{2}$, but $|S| < \frac{n}{10}$, it discards this set and resets *Total_Cost* to zero. Since the total cost of LHP is at most $5B$ and it contains n terminal points, this resetting can take place at most nine times and we have at most 10 disjoint sets, each set forms a subpath with cost no more than $\frac{B}{2}$. By *Pigeon Hole principle*, one of these 10 sets contains at least $\frac{n}{10}$ $p - type$ points and hence we obtain desired P'. Here we should notice, from p_i^* to p_{i+1}^*, there could be other terminal nodes(for example from p_{16} to p_8, p_7 is also included). However, it does not decrease the cardinality of set S and the lemma still holds. □

Now we present Garg's algorithm for $K - MST$ problem.

Definition: *Garg's algorithm*: Given a weighted graph $G = (V, E)$ and number k, the algorithm finds a subgraph with exactly k vertices, and the total cost is at most 2 times the optimal.

Lemma 5. *Given T_{OptSol}, let G_V be the complete graph using P' as vertices set and all straight segments between any two terminals as edge set. If we set $k = \frac{n}{10}$ ($n = |P'|$) and run Garg's algorithm on G_V, it returns a subgraph(tree) with total cost at most B.*

Proof. This proof is quite intuitive. By Lemma 4 we know there exists a subpath, P', which includes at least $\frac{n}{10}$ terminal points and whose cost is at most $\frac{B}{2}$. From proof of Lemma 3, P' corresponds to a subgraph of C^*, say C', whose cost is no more than cost of P' and with the same vertex set. Since we use only straight segment to construct C^*, both C^* and C' are subgraphs of G_V. By the guaranteed 2 performance bound of Garg's algorithm we must find one Spanning Tree with cost at most B (for the fact spanning tree of C' is such a candidate). Hence the lemma follows. $\square$

Theorem 1. *Algorithm 1 is a $\frac{1}{10}$-approximation algorithm, i.e., $\frac{AppSol}{OptSol} \geq \frac{1}{10}$.*

Proof. From Lemma 5, we know that with cost B, Algorithm 1 will be able to find a Spanning Tree with at least $k \geq \frac{OptSol}{10}$ nodes. By enumerating value of k in decreasing order, we must find a feasible tree and $AppSol \geq k$. Hence the theorem holds. $Garg's$ algorithm runs in $O(mn^4 \log n)$ time, and our Algorithm 1's running time is $O(mn^5 \log n)$. $\square$

2.4.2 BCRP-MSCC

In this part, we show inapproximability of **BCRP-MSCC**.

Theorem 2. *There is no polynomial-time approximation algorithm for BCRP-MSCC with approximation better than $\frac{1}{2}$ unless P = NP.*

Proof. Suppose there exists such approximation algorithm. W.l.o.g, we may assume we are given n terminal points and their locations. If n is a even number, let (a, b) be the coordinate of a point with minimum a value, we can add one more sensor node with coordinate $(a-R, b)$. It is easy to check that we do not need to spend any budget to connect the new node and it does not bring any benefit for connectivity. Hence, we can always assume n is an odd number. Now, we set Y(the size of the smallest component) to be n and run the approximation algorithm on this instance, we obtain the size of the smallest component, say *app*. Again, we have two scenarios: 1) If the algorithm says $app \leq \lfloor \frac{n}{2} \rfloor$, let *opt* be the optimal solution, then by our assumption $\frac{app}{opt} \geq \frac{1}{2} => opt \leq app * 2 < n$. If all terminals can be connected using extra steiner points, the size of the minimum component is n. Hence in this scenario, we know that it is impossible to link all terminals under given budget. 2) Otherwise, the algorithm says $app > \lfloor \frac{n}{2} \rfloor$. We should notice that if the graph is disconnected, there are at least 2 components. By pigeon hole principle, the size of minimum components is at most $\lfloor \frac{n}{2} \rfloor$. Hence if $app > \lfloor \frac{n}{2} \rfloor$, it implies the graph can be connected. In either way, we have a confident YES/NO answer to STP-MSPBEL problem, therefore unless $P = NP$, there is no approximation better than $\frac{1}{2}$. □

2.4.3 BCRP-MNCC

In this part, we show inapproximability of **BCRP-MNCC**.

Theorem 3. *There is no polynomial-time approximation algorithm for BCRP-MNCC within approximation $2 - \epsilon$ for any $\epsilon > 0$, unless $P = NP$.*

Proof. We use contradiction again. Suppose there is such an approximation algorithm so that it will not generate more than $2-\epsilon$ times the optimal number. We then run this algorithm on a given instance by setting $Z = 1$, here Z is the number of components. If the algorithm returns only one component, we know that there is a deployment connecting all sensor nodes. Otherwise, the algorithm returns at least 2 components. Since we assume approximation ratio is at most $2 - \epsilon$. The optimal solution has at least $\frac{2}{2-\epsilon}$ components. As the number of components must an integer and $\frac{2}{2-\epsilon} > 1$, there are at least 2 components in optimal solution which implies with current budget it is impossible to connect the graph. In either case, we can give a confident YES/NO answer to STP-MSPBEL problem. Hence, unless P = NP, there is no polynomial-time approximation algorithm for BCRP-MNCC within approximation $2 - \epsilon$ for any $\epsilon > 0$. □

2.5 Mathematical Programming Formulation

In this section we provide a mathematical programming formulation for the problem to find the optimal solution. Since our problem is NP-hard in terms of complexity we expect that the instance sizes for which we would be able to find optimal solution in a reasonable amount of time are limited. Our decision variables are the locations of relay nodes and our objective function is to maximize connectivity of the network based the size of the largest connected component. An idea to improve the computation is to reduce the solution space. We have reduced the search space to the rectangle with vertices $(x_{min}, y_{min}), (x_{max}, y_{min}), (x_{min}, y_{max}), (x_{max}, y_{max})$ such that x_{min} and x_{max} are the minimum and maximum x-coordinate of the sensors and y_{min} and y_{max} are the minimum and maximum y-coordinate of the sensors.

2.5.1 Formulation

Sets:

S: set of all existing sensor nodes.

K: set of relay nodes.

N: set of entire nodes, such that $N = S \cup K$.

Parameters:

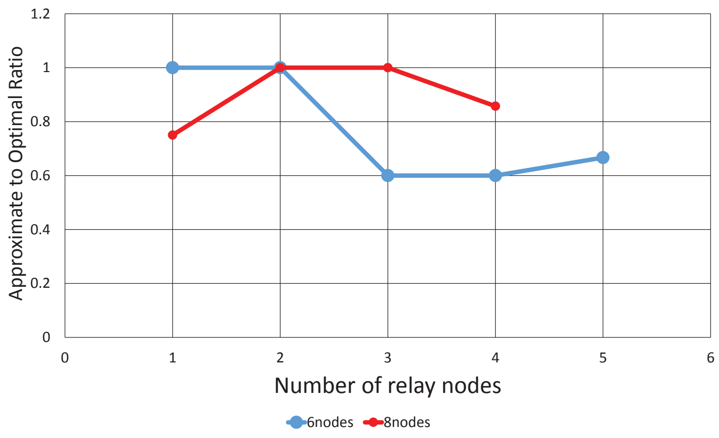
x_i: x coordinate of the sensor node i in Euclidean plane.

y_i: y coordinate of the sensor node i in Euclidean plane.

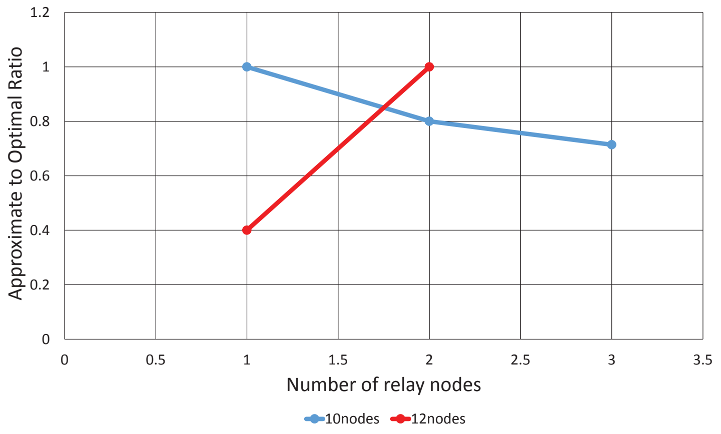
a_{ij}: binary parameter, which is equal to one if distance between two sensor nodes i and j is less than or equal to R and zero otherwise.

x_{min}: minimum value of x_i, such that $i \in S$.

x_{max}: maximum value of x_i, such that $i \in S$.



(a) 6 and 8 sensor nodes



(b) 10 and 12 sensor nodes

Figure 2.5: Results of experimental evaluation of the approximation algorithm

y_{min}: minimum value of y_i, such that $i \in S$.

y_{max}: maximum value of y_i, such that $i \in S$.

R: transmission range for sensors and relays

M: a very large number

Variables:

w_i: x coordinate of relay node i in Euclidean plane.

z_j: y coordinate of relay node j in Euclidean plane.

d_{ij}: Euclidean distance between two nodes i and j, such that $i \in N$ and $j \in K$.

a'_{ij}: binary variable, which is equal to one if distance between two nodes i and j is less than or equal to R and zero otherwise, such that $i \in N$ and $j \in K$.

c_{ij}: binary variable, which is equal to one if nodes i and j are connected and zero otherwise, such that $i, j \in N$.

b_{ikj}: binary variable, which is equal to one if the distance between nodes i and k is less than or equal to R *and* nodes k and j are connected to each other, and zero otherwise, such that $i, j, k \in N$.

G: Size of the largest connected component in the network graph.

α_i: Binary auxiliary variable to find the size of the largest connected component, where $i \in S$

Model:

$$\max(G) \tag{2.1}$$

Subject to:

$w_i \leq x_{max} \qquad \forall i \in K \qquad (1)$

$x_{min} \leq w_i \qquad \forall i \in K \qquad (2)$

$z_i \leq y_{max} \qquad \forall i \in K \qquad (3)$

$y_{min} \leq z_i \qquad \forall i \in K \qquad (4)$

$d_{ij}^2 = (w_i - w_j)^2 + (z_i - z_j)^2, \qquad \forall\ i, j \in K \qquad (5)$

$d_{ij}^2 = (x_i - w_j)^2 + (y_i - z_j)^2, \qquad \forall\ i \in S, j \in K \qquad (6)$

$d_{ij} - (1 - a'_{ij})M \leq R, \qquad \forall\ i \in N, j \in K \qquad (7)$

$c_{ij} \leq \sum_{k \in N} b_{ikj} + a_{ij}, \qquad \forall\ i, j \in S \qquad (8)$

$c_{ij} \leq \sum_{k \in N} b_{ikj} + a'_{ij}, \qquad \forall i \in N, j \in K \qquad (9)$

$b_{ikj} \leq a_{ik} c_{kj}, \qquad \forall\ i, k \in S, j \in N \qquad (10)$

$b_{ikj} \leq a'_{ik}c_{kj}, \qquad \forall\ i, j \in N, k \in K \qquad (11)$

$b_{ikj} \leq \sum_{l\in S} b_{klj}(1 - a_{il}) + \sum_{l\in K} b_{klj}(1 - a'_{il}) + a_{kj}, \qquad \forall\ i, j, k \in S \qquad (12)$

$b_{ikj} \leq \sum_{l\in K} b_{klj}(1 - a'_{il}) + \sum_{l\in S} b_{klj}(1 - a_{il}) + a'_{kj}, \qquad \forall\ i \in S, k \in N, j \in K \qquad (13)$

$b_{ikj} \leq \sum_{l\in K} b_{klj}(1 - a'_{il}) + \sum_{l\in S} b_{klj}(1 - a'_{li}) + a'_{kj}, \qquad \forall\ i, j \in K, k \in N \qquad (14)$

$b_{ikj} \leq \sum_{l\in K} b_{klj}(1 - a'_{il}) + \sum_{l\in S} b_{klj}(1 - a_{il}) + a'_{jk}, \qquad \forall\ i \in S, k \in K, j \in S \qquad (15)$

$b_{ikj} \leq \sum_{l\in K} b_{klj}(1 - a'_{il}) + \sum_{l\in S} b_{klj}(1 - a'_{il}) + a_{kj}, \qquad \forall\ i \in K, k \in S, j \in S \qquad (16)$

$G = \sum_{i\in S}(\alpha_i \sum_{j\in S} c_{ij}), \qquad (17)$

$\sum_{i\in S} \alpha_i = 1, \qquad (18)$

Objective function of this model is to maximize the size of the largest connected component. Constraints (1)-(4) refer to the fact that the optimal location of the relay nodes must be in the space between the sensor nodes. The reason to consider this is to reduce the search space of the problem. In constraints (5) and (6), Euclidean distance between two relay nodes and the one between a sensor and a relay node are computed, respectively. Constraint (7) forces a'_{ij} to become zero when the distance between node i and relay j is greater than R. Constraints (8) and (9) say that nodes i and j are connected if either i and j are at a distance less than or equal to R from each other or there is at least one node that connects i and j as an intermediate node and that node is at a distance less than or equal to R from node i. Constraints (10) and (11) are used to define the binary variable b_{ikj}. Constraints (13)-(16) show that if nodes i and j are connected using an intermediate node k, this node must be either at a distance less than or equal to R from node j or there is at least one other node out of range of i which connects node k and node j. Constraint (17) finds size of the largest connected component using auxiliary variable α_i. Constraint (18) indicates that only one of the values of α_i can be one.

2.5.2 Linearization

The proposed mathematical model is nonlinear. In order to increase the efficiency and decrease running time we have linearized the model and transformed it to a MILP model. Constraints (1), (2), (3), (4), (7), (8), (9) and (18) are already linear. Constraints (10) to (17) are simply linearized using McCormick Envelope method since the only nonlinear part is the multiplication of two binary variables.
Constraints (5) and (6) are in the form of Euclidean norm inequalities. Using linearization method introduced in [4] we define n directions from origin with length one equally dividing the Euclidean plane. Consider the inner product of distance vector between two nodes and each of these directions. If the inequality holds for all of these inner products then it would also hold for the Euclidean distance with an error which is dependent on n. The value of n should be large enough such that the distance approximation does not interfere with optimality of the solution, but it should not be arbitrarily large since it will add constraints to our MILP and increase the running time.

2.6 Experimental Results

In this section we present our experimental results on the efficacy of our approximation algorithm. Using the mathematical programming formulation presented in Section 2.5, we compute the optimal solution and compare the results obtained from the Algorithm 1 given in Section 6.4. For our experiments, we have used synthetic random data with 6, 8, 10, 12 sensor nodes in a 5×10 deployment area, varying the number of the relay nodes from 1 to 5. It may be noted that experimental results with a larger number of sensor nodes, or larger number of relay nodes could not be provided as the time to compute the optimal solutions using the formulation given in Section 2.5 turned out to be unacceptably high. We have computed the optimal solution using CPLEX solver for AMPL on a Intel Core i7 machine with 8GB RAM and 2.3 GHz processor. The approximation algorithm was also executed on the same machine. We have plotted the ratio between the approximate and the optimal solutions for the problem in Figure 2.5. It may be noted that although our analysis in Section 6.4 shows that the ratio between the approximate to optimal solutions could not be any lower than 0.1, in our experiments this ratio was never any lower than 0.4. Also, the computation time for the approximate solution was only a fraction of the time required for finding the optimal solution.

2.7 Future Direction

It may be noted that while BCRP-MLCC/BCRP-MSCC problems focus only on the size of the largest/smallest connected component while ignoring the number of components, the BCRP-MNCC problem focuses only on the number of connected components while ignoring the size of these components. One can argue all these three metrics are myopic in nature as they pay attention either only to the *size* of the largest/smallest component or to the *number* of components, two key attributes of a disconnected graph. To address this concern, we propose a fourth metric to measure the "connectedness" of a disconnected graph that takes into account both the size as well as the number of connected components. In the following we define the metric.

Every disconnected graph comprises of a number of components of differing sizes. It might comprise of one component of size n, where n is the number of nodes of the graph, or n components of size one, or any other distribution in between. Accordingly, any graph can be described by the *frequency distribution* of its component sizes.

We define C_i as the number of connected components of size i in $G = (V, E)$, $0 \leq C_i \leq n, \forall i, 1 \leq i \leq n$ (as shown in Table 2.1). A *Component Frequency Distribution*

Frequency Distribution of Connected Components in $G = (V, E)$						
C_i **= Number of connected components of size** i**,** $n = \|V\|$						
$G = (V, E), \mathcal{C}$	C_n	C_{n-1}	C_{n-2}	$\cdots$	C_2	C_1
$G_1 = (V_1, E_1), \mathcal{C}_1$	x_n	x_{n-1}	x_{n-2}	$\cdots$	x_2	x_1
$G_2 = (V_2, E_2), \mathcal{C}_2$	y_n	y_{n-1}	y_{n-2}	$\cdots$	y_2	y_1

Table 2.1: Component Frequency Distribution Table

Vector (CFDV) $\mathcal{C}$ is defined as a vector of size n where the i-th entry of the vector specifies the number of connected components of the graph $G = (V, E)$. The weight of a CFDV $\mathcal{C}$ is defined as follows: $w(\mathcal{C}) = \Sigma_{i=1}^{n} C_i.(n+1)^{i-1}$. It may be noted $w(\mathcal{C})$ takes into account both the size and the number of components of the graph, with larger components being assigned with higher weights than the smaller components. We define the *"disconnectivity"* of a graph $G = (V, E)$, $\delta(G)$, with CFDV $\mathcal{C}$ as $\delta(G) = 1/w(\mathcal{C})$. *Example:* Consider two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ each with 6 nodes. The CFDVs of G_1 and G_2 ($\mathcal{C}_1$ and $\mathcal{C}_2$) are given as [0, 0, 1, 0, 0, 2] and [0, 0, 0, 2, 0, 0] respectively, implying that G_1 comprises of one component of size 4 and two components of size 1, while G_2 comprises of two components of size 3. With these parameter values, $w(\mathcal{C}_1)$ is 345 and $w(\mathcal{C}_2)$ is 198 which in turn determines $\delta(G_1) = 0.002899$ and $\delta(G_2) = 0.005051$. As the value of $\delta(G_1)$ is smaller than $\delta(G_2)$, it implies that G_1 is *less disconnected* (or, *more connected*) than G_2. It may be noted that we arrive at this conclusion, because we chose to give higher weights to components with a larger size. The optimization problem using this metric is referred to as the *Budget Constrained Relay node Placement for Minimum Disconnectivity* (BCRP-MD) problem. Next, we provide a formal definition of the BCRP-MD problem.

BCRP-MD Problem: Given the locations of n sensor nodes in the Euclidean plane

$P = \{p_1, p_2, \dots, p_n\}$, transmission range R and a budget B on the number of relay nodes that can be deployed, is it possible to find a set of points $Q = \{q_1, q_2, \dots, q_m\}$, where $m \leq B$, in the same plane where relay nodes can be deployed, so that the *disconnectivity*, $\delta(G')$, of the graph $G' = (V', E')$ corresponding to the point set P and Q is at most D, for a pre-specified value D?

It is true that the two key attributes of a disconnected graph (size and number of components) could have been combined in ways other than the one proposed here, $w(\mathcal{C}) = \Sigma_{i=1}^{n} C_i.(n+1)^{i-1}$. However, this way of combining the two attributes has the advantage that, just as given the Component Frequency Distribution Vector (CFDV) $\mathcal{C}$, one can easily compute the "disconnectivity" $\delta(G)$, given $\delta(G)$ and n, the number of nodes in G, one can easily recompute the CFDV of G.

Chapter 3

APPROACHES TO MINIMUM D-DEGREE ARRANGEMENT

3.1 Introduction

Many systems in the world such as cellular networks, the post service, or transportation pathways can be modeled as networks or graphs. The practical applications of graph algorithms generally seek to achieve some goal while minimizing some cost such as money or distance. An application of our problem can be seen in overlay networks in telecommunications. An overlay network is a virtual network that is built on top of another network. It is a logical network where the links between nodes represent the physical paths connecting the nodes in the underlying infrastructure. The underlying physical network may be incomplete, but as long as it is connected, we can build a complete overlay network on top of it. Since some nodes may be overloaded by traffic, we can reduce the strain on the overlay network by limiting the communication between nodes. Some edges, however, may have more importance than others so we must be careful about our selection of which nodes are allowed to communicate with each other. The balance of reducing the degree of the network while maximizing communication forms the basis of our d-degree minimum arrangement problem.

3.1.1 Previous Work

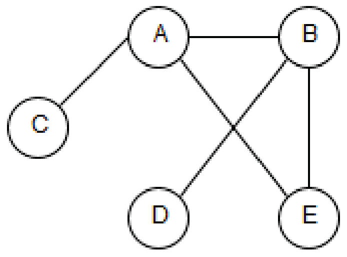
The minimum linear arrangement problem is known to be NP-hard. Rao and Richa [47] presented an approximation ratio of $O(\log n)$ in 2004. Since then, Charikar, Hajiaghayi, Karloff, and Rao [6] and independently, Feige and Lee [17] lowered the approximation ratio to $O(\sqrt{\log n} \log \log n)$. The problem of graph embedding in

d-dimensions was studied by Even, Naor, Rao, and Schieber [16], who obtainted an approximation guarantee of $O(\log n \log \log n)$ by applying their spreading metric framework. Later, Charikar, Makarychev, and Makarychev [7] presented a divide-and-conquer approach to lower the approximation ratio to $O(\sqrt{\log n})$. However, no work has been done in the direction of the d-degree minimum arrangement problem.

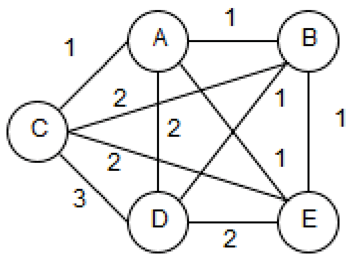
3.1.2 Problem Description

MLA

Our problem can be seen as a variation of the minimum linear arrangement problem, which can be defined in the context of our problem as the following: Given an undirected, unweighted graph $G = (V, E)$ where the edges represent a set P of communication request pairs, produce a complete graph G' on V the using the shortest-path metric on G. Then find a subgraph of degree at most two in G' - which may be a path, a cycle, or a collection of cycles and paths - that minimizes the sum of the shortest-path distances between all node pairs in P. Figure 3.1 depicts the construction of G' from a graph G.



(a) $G = (V, E)$



(b) Complete graph G' constructed from $V(G)$ with shortest-path metric

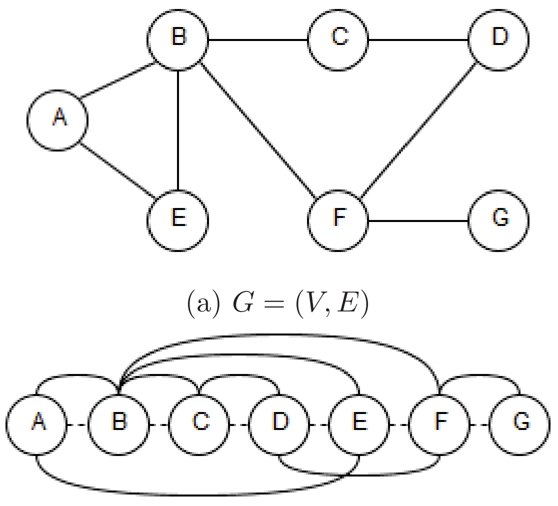
Figure 3.1: Minimum linear arrangement setup

An alternative definition for the MLA problem is as follows: Given an undirected, unweighted graph, produce a linear ordering (permutation) of the nodes such that the sum of the cost of the edges in the ordering is minimized. The cost of an edge (u, v) in the input graph is defined to be equal to the difference of the positions of u and v given by the permutation of the nodes. Each edge in the input represents a request between the pair of edge nodes (i.e, for communication or a path).

Figure 3.2 depicts an example of a graph G and a possible linear ordering of its nodes. The cost of the request (B, F) in this ordering is given by $|Pos(F) - Pos(B)| = 6 - 2 = 4$. The total cost of this ordering is given by:

$$\sum_{\forall (u,v) \in E} |Pos(u) - Pos(v)|$$

The weighted version of the problem uses the same setup but defines a weight associated with each edge or request pair. The cost for a pair of nodes is then



(a) $G = (V, E)$

(b) Linear ordering of $V(G)$

Figure 3.2: Example of a linear arrangement

modified to be the distance between the nodes on the permutation multiplied by the weight of the edge in the input graph G.

$$\sum_{\forall(u,v)\in E} (|Pos(u) - Pos(v)| * w(u,v))$$

d-degree minimum arrangement

Our project focuses on the generalized version of the first definition of the MLA problem where we are looking for a subgraph is a graph with degree at most d, instead of 2, in the complete graph constructed on the shortest-path metric. The cost of a request in our representation is defined as the sum of the weights of the edges in the shortest path between the two nodes in our solution subgraph.

Our problem can be more formally described as the following: Given a complete, undirected graph $G = (V, E)$ that satisfies triangular inequality and a set of pair of nodes $P = \{(x_1, y_1), \ldots, (x_p, y_p)\}$ such that $x_i, y_i \in V$, find a subgraph $G' = (V, E')$ of max degree at most d such that the sum of the costs of the shortest (x_i, y_i)-paths in G', over all $(x_i, y_i) \in P$, is minimum.

$$
\begin{aligned}
min \sum_{(s,t)} C(s \to t) & \qquad \forall (s,t) \in P \\
min \sum_{(s,t)} \left(\sum_{(u,v) \in s \to t} w(u,v) \right) & \qquad \forall (s,t) \in P
\end{aligned}
$$

Where $s \to t$ denotes the shortest-distance path between s and t in G' and $C(s \to t)$ denotes the cost of that path according to the definition of d-degree minimum arrangement.

3.1.3 Integer Linear Program

Our problem can be formulated as an integer linear programming problem as follows. The variable $x_{u,v}^{s,t}$ is an indicator variable determining whether edge $(u, v) \in E$ is used in the selected (s, t)-shortest path in G'. The subscripts s, t represent every (s, t) pair and u, v represent the edge uv in that direction.

The following represents the constraints representing flow conservation.

$$
\begin{aligned}
\sum_{(s,v) \in E} x_{s,v}^{s,t} - \sum_{(u,s) \in E} x_{u,s}^{s,t} &= 1 \\
\sum_{(u,v) \in E} x_{u,v}^{s,t} - \sum_{(v,p) \in E} x_{v,p}^{s,t} &= 0 \qquad \text{for all } v \in V \setminus \{s,t\} \\
\sum_{(u,t) \in E} x_{u,t}^{s,t} - \sum_{(t,v) \in E} x_{t,v}^{s,t} &= 1 \\
x_{u,v}^{s,t} &\in \{0, 1\}
\end{aligned}
$$

The outgoing flow from the source node and the incoming flow to the sink node must be 1. The change in flow in every other node along the path must be 0 to maintain flow conservation.

The cost of each path can be calculated as the following.

$$C_{s,t} = \sum_{(u,v)\in E} x^{s,t}_{u,v} * w(u,v) \qquad \text{for all (s,t)}$$

The goal of the algorithm is to minimize the sum of the costs for every (s,t) pair in P.

$$\sum_{\forall(s,t)\in P} C_{s,t}$$

The degree constraint can be represented using the indicator variable y_e dependent on whether or not the edge e has been selected by some (s,t) path.

$$y_{u,v} \geq x^{s,t}_{u,v} \qquad \text{for all (s,t)}$$

$$y_{v,u} \geq x^{s,t}_{u,v} \qquad \text{for all (s,t)}$$

The degree constraint can then be formulated as the following.

$$\sum_{u:(u,v)\in E} y_{u,v} \leq d \qquad \text{for all v} \in \text{V}$$

3.2 Project Work

3.2.1 Approaches

The minimum linear arrangement problem is NP-hard, implying that an efficient, polynomial-time algorithm for solving it has not been discovered. While this does not immediately imply that the d-MA problem, for $d > 2$, is NP-hard, it provides good indication that this is probably the case. Hence, our goal for the multi-degree MLA problem was to find an efficient heuristic and then compare the results of our algorithm to the respective optimal solutions output by the ILP.

Top-Down Pruning

An initial approach to the multi-degree MLA problem was the idea of a greedy, top-down pruning algorithm. The inspiration for this approach came from the idea of the decreasing number of neural links in the brain as humans age. There may be specific connections that are crucial and others that can be removed. We can model the neural map of the brain at the start as a complete graph where connections are gradually removed. This led to the hypothesis about the existence of an efficient algorithm that takes a dense graph as input and produces a sparse graph fulfilling some criteria - such as minimizing distances between certain vertices - as output. This algorithm would start with the complete graph and remove edges according to a heuristic, until a subgraph satisfying the degree constraint remained. For the top-down approach we first considered a simple heuristic that minimizes the sum of the cost of all requests given and the weights of all edges remaining in the graph. The heuristic can be expressed as

$$min\left(\sum_{(u,v)\in P} C_H(u,v) + \sum_{e\in H} w_e\right)$$

where H is a subgraph of G. In the context of an algorithm, we would consider all edges where both vertices have degree larger than the required degree and choose the edge that minimizes the above sum when removed. This can be expressed as the following where e is the edge in consideration:

$$min_{e\in H}\left\{\sum_{(u,v)\in P} C_{H\setminus\{e\}}(u,v) + \sum_{e'\in H\setminus\{e\}} w_{e'}\right\}$$

Let e be the edge that minimizes the formula above. Then e would be a good candidate to be removed from the graph. An algorithmic representation is given by the following.

Algorithm 3 d-LA Algorithm

```
1: procedure GREEDY TOP-DOWN $d$-MA
2:     Initially $H = G$
3:     while $\exists v \in V$ with $deg(v) > d$ do
4:         Let $E' = \{(u,v)$ such that $deg(u) > d$ and $deg(v) > d\}$
5:         Pick $e \in E'$ s.t. $e$ minimizes $\sum_{(u,v)\in P} C_{H\setminus\{e\}}(u,v) + \sum_{e'\in H\setminus\{e\}} w_{e'}$
6:         Let $H = H\setminus e$
7:     end while
8: end procedure
```

This algorithm, however, has a high chance of failing to terminate. It would fail if there is only one vertex left with higher degree, as shown in Figure 3.3. Suppose the degree constraint is 2 for this given problem. The gray node represents the node that violates the degree.

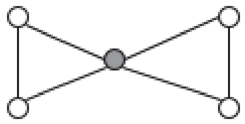


Figure 3.3: Bad structure for top-down algorithm

To avoid this, we modified the algorithm and added a second phase where it can choose to remove any edge with only one node of a higher degree to produce a solution of degree d or less. This algorithm still experienced problems when it produced a structure in which the only possible edges that it could choose would disconnect the graph and result in infinite in cost for certain requests. If the algorithm did not remove these edges, however, the solution would violate the degree constraint. Figure 3.4 shows some examples of bad structures that could be produced by the algorithm. The left figure represents a portion of a larger graph where all three edges depicted are cut-edges. Since the middle node has degree 3, one of its edges must be removed resulting in a disconnected graph. The right figure represents a possible graph where there is no possible way to remove enough edges while keeping the graph connected.

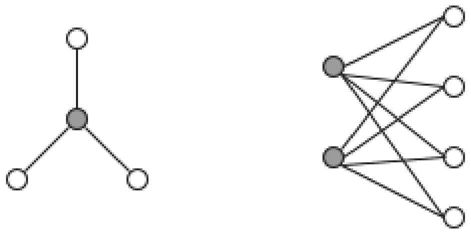


Figure 3.4: Bad structures for top-down algorithm

Thus, we considered the following variation for pruning. Instead of allowing the algorithm to select any edge of a higher degree, we tried to remove edges in perfect matchings to keep all vertices at roughly the same degree. Figure 3.5 shows an example of reducing a graph from degree 3 to degree 2 by removing a matching.

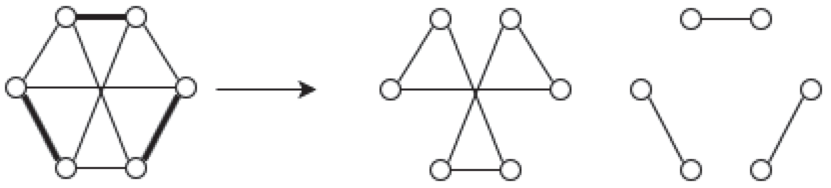


Figure 3.5: Example of removing a matching from a graph

There are problems in this approach as well since it is not possible to guarantee that perfect matchings exist at every point in the algorithm. Figure 3.6 shows an example of the algorithm failing when trying to remove a matching. Since the algorithm selects one edge on every iteration by greedy principles, it does not globally know whether the edge that it selects is part of a matching or not. As a result, the algorithm will fail if it reaches a point where the only possible edges it can select do not exist.

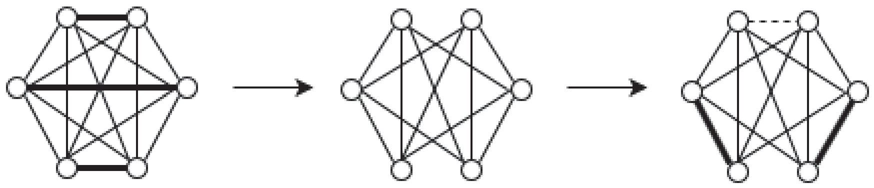


Figure 3.6: Example of bad matching

Since there were many factors to consider in controlling the structure of the solution, we shifted our focus away from the idea of pruning and instead considered a bottom-up growing approach.

Bottom-Up Growing

For our growing approach we considered a greedy, bottom-up heuristic that started with an empty edge set and added edges on every iteration. We defined the cost of this edge set as the sum of the costs of all request pairs. The cost of the empty edge set initially sums up to infinity due to the lack of connections between the pairs. During each iteration of the algorithm, we would first consider edges whose vertices have the lowest degree. We would choose to include the edge that maximally decreased the cost of the edge set until we are forced to consider the next highest degree. We continue adding edges in this manner until either the solution set reaches the degree limit or until the cost can no longer be minimized. The algorithm for this greedy heuristic is as follows.

Algorithm 4 d-MA Algorithm

```
1: procedure Greedy Bottom-Up $d$-MA
2:     $currentSmallestCost \leftarrow \infty$
3:     $currentDegree \leftarrow 0$
4:     $edgeSet \leftarrow \emptyset$
5:     while $currentDegree < d$ do
6:         for all edge $e = (x, y)$ in $E$ such that degree of $x, y < d$ do
7:             $currentCost = \sum_{(s,t) \in P}(C_{edgeSet \cup e}(s, t))$
8:             if $currentCost < currentSmallestCost$ then
9:                 $currentSmallestCost \leftarrow currentCost$
10:                $bestEdge \leftarrow e$
11:            end if
12:        end for
13:        if $bestEdge \neq null$ then
14:            $edgeSet = edgeSet \cup bestEdge$
15:        else
16:            $currentDegree = currentDegree + 1$
17:        end if
18:    end whilereturn $edgeSet$
19: end procedure
```

3.2.2 Upper Bound

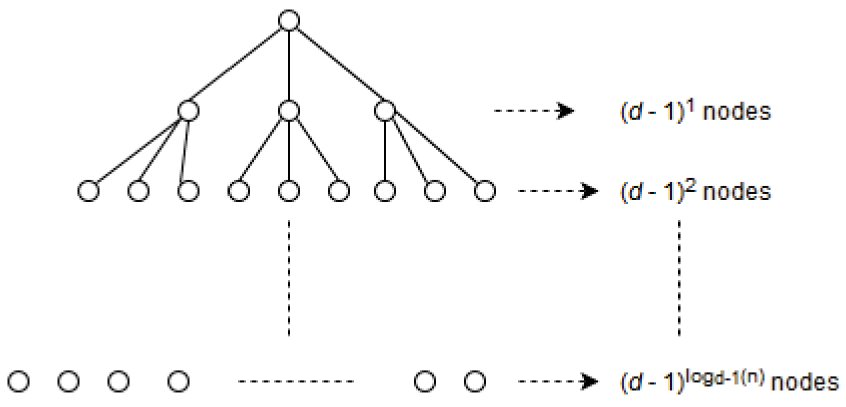


Figure 3.7: A complete $(d-1)$-ary tree (where $d = 4$ in this case).

For the sake of simplicity, we will only consider uniform graphs in this section. Let P be the set of the required pairs and $d > 2$. For a P of any size, construct an arbitrary complete $(d-1)$-ary tree T containing all the nodes in V. We want to embed the nodes of G into T and use the edges of T to connect the pairs in P. Simply put, we want to construct a solution with the structure of T. The height of T is

$$\log_{(d-1)} n$$

The length of a (u, v)-path in T is less than or equal to twice the height of the tree. Let $P_{u,v}$ represent the path connecting u and v in T. Therefore, the summation across all pairs in P becomes

$$\sum_{(u,v)\in P} length(P_{u,v}) \leq 2|P| \log_{(d-1)} n$$

The cost for using an any $(d-1)$-ary tree is

$$O(|P| \log_{(d-1)} n)$$

3.2.3 Optimal Structure for $|P| = \Theta(n^2)$

If the set of required communication pairs consists of all pairs of nodes, the optimal solution will consist of a complete d-ary tree.

Lower Bound for $|P| = \Theta(n^2)$

Since there are $|P|$ requests, the contribution of each request in the final solution is at least 1. Then the cost of the final solution is at least equal or greater than the size of P. Thus the trivial lower bound of the solution is:

$$\Omega(|P|)$$

If $|P| = \Theta(n^2)$, then the lower bound is approximately $n * (n-1)/2 = \Omega(n^2)$.

To achieve a tighter lower bound, for each node u, construct a tree rooted at u by putting the maximum number of nodes as close to u as possible on every level. This tree optimizes the distances from u to all other nodes, since running a breadth-first search would produce the same structure. The number of nodes on every level i edges away from u can be expressed as:

$$\sum_{i=1}^{x} (d-1)^i$$

Where $x \approx \lfloor \log_{d-1} n \rfloor$ represents the number of levels in the tree.

Therefore, the sum of the lengths of all paths from every other node to u can be represented by:

$$\sum_{v \in V} length(P_{u,v}) = \sum_{i=1}^{\log_{d-1} n} (d-1)^i * i$$

Where $(d-1)^i$ represents the number of $v \in V$ such that the length of $P_{u,v}$ is i.

Consider orienting the tree rooted at u with the u at the top and each successive layer one level below the previous. This tree has a height of $\lfloor \log_{d-1} n \rfloor \leq \log_{d-1} n$ levels. Select the bottom half of the tree, resulting in $\frac{\log_{d-1} n}{2}$ levels. Each level in the bottom half has more nodes than each corresponding level in the top half, so therefore there are at least $\frac{n}{2}$ nodes in the bottom $\frac{\log_{d-1} n}{2}$ levels.

The sum of the length of the paths from the nodes in these bottom $\frac{\log_{d-1} n}{2}$ levels, denoted $V*$, is represented by:

$$\begin{aligned}\sum_{v \in V*} length(P_{u,v}) &\geq \frac{n}{2} * \frac{\log_{d-1} n}{2} \\ &= \Omega(n \log_{d-1} n)\end{aligned}$$

This is the lower bound for any particular u appearing in a pair in the set P. If the problem requires all pairs, then we can calculate the total lower bound by constructing a u-rooted tree for all nodes $u \in V$. Therefore, the lower bound would become:

$$\Omega(n^2 \log_{d-1} n)$$

Optimal Bound

When the problem requires one to connect close to all pairs of nodes, i.e. when $|P| = \Theta(n^2)$, the upper bound shown in section 3.2.2, would become:

$$\begin{aligned}&O(|P| \log_{d-1} n) \\ &= O(n^2 \log_{d-1} n)\end{aligned}$$

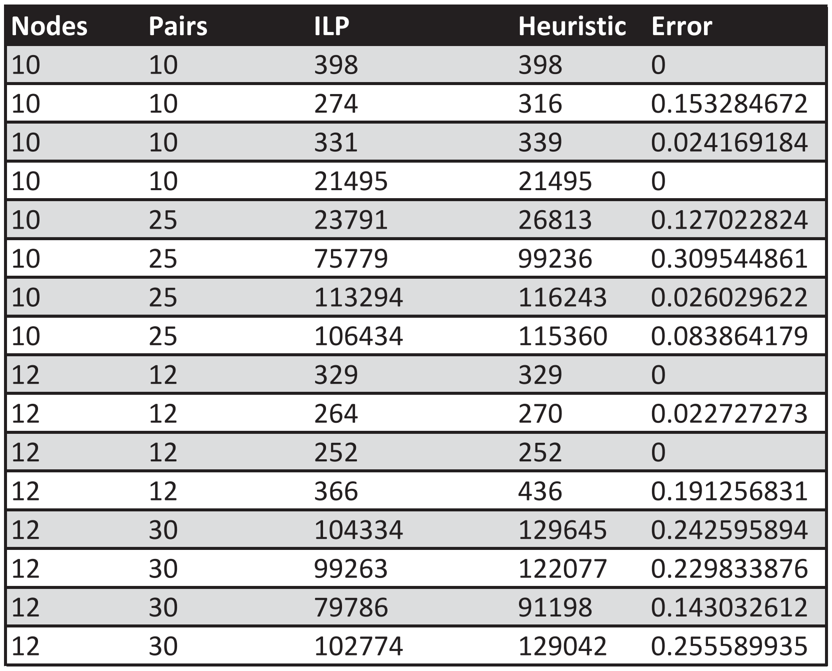
Therefore, when the problem requires a large number of pairs, the upper bound and the lower bound are the same. The $d-1$-ary tree structure gives the the optimal bound on the runtime as:

$$\Theta(n^2 \log_{d-1} n)$$

3.3 Results

3.3.1 Comparison of Heuristic to Optimal Solution

Figure 3.8 shows simulation results for small graphs of sizes from 10 to 12 nodes with various numbers of pairs for $d = 2$. It compares the solutions from the bottom-up heuristic against the optimal solution from the ILP. On several occasions the heuristic achieved the same optimal result as the ILP. The margin of error of the heuristic varies between 0 and approximately 25%. Figure 3.9 shows a graph of the average and standard deviation of error from Figure 3.8 grouped by number of nodes and number of pairs.



Nodes	Pairs	ILP	Heuristic	Error
10	10	398	398	0
10	10	274	316	0.153284672
10	10	331	339	0.024169184
10	10	21495	21495	0
10	25	23791	26813	0.127022824
10	25	75779	99236	0.309544861
10	25	113294	116243	0.026029622
10	25	106434	115360	0.083864179
12	12	329	329	0
12	12	264	270	0.022727273
12	12	252	252	0
12	12	366	436	0.191256831
12	30	104334	129645	0.242595894
12	30	99263	122077	0.229833876
12	30	79786	91198	0.143032612
12	30	102774	129042	0.255589935

Figure 3.8: Results of graphs with 10-12 nodes

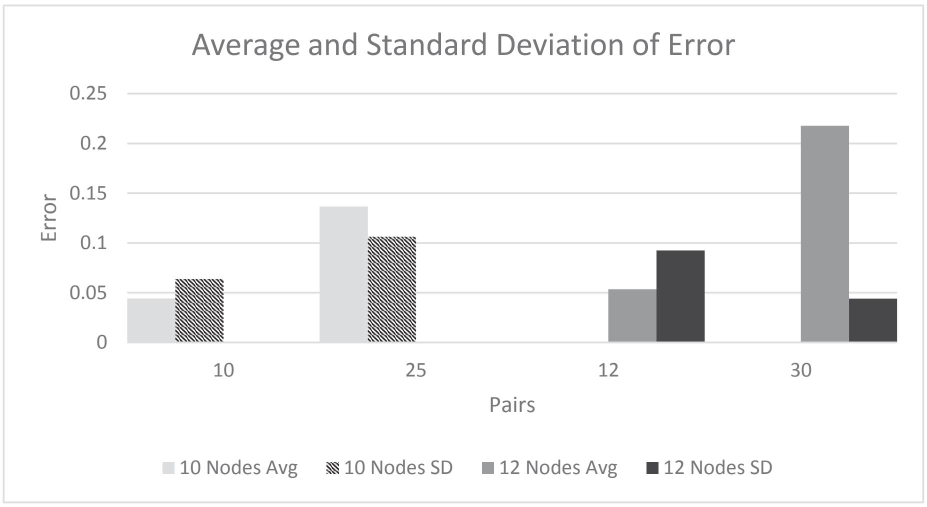


Figure 3.9: Average and standard deviation of error for Figure 3.8

Figure 3.10 shows a sample of the simulation results for larger graphs of 15 to 20 nodes with various numbers of pairs for $d = 3$. Since the ILP cannot run on a large number of variables efficiently, we relaxed the conditions to allow non-integer solutions. As a result, the difference between the heuristic and the relaxed ILP solutions increases. The relaxed ILP solutions are denoted by the LP column. Since it is difficult to determine the difference between the optimal integer solution and the relaxed solution, the actual margin of error of the heuristic is unknown. Figure 3.11 shows a graph of the average and standard deviation of error of the results from Figure 3.10 and 2 other samples - a total of 4 samples for each node-pair combination - grouped by number of nodes and number of pairs.

Nodes	Pairs	LP	Heuristic	Error
15	30	47053.41121	112929	1.40001728
15	30	50710.66667	91303	0.800469329
15	50	86671.97321	209402	1.416028991
15	50	95753.04	232574	1.428894164
20	20	399.2262443	856	1.144147616
20	20	365.4089669	773	1.115437961
20	50	88186.66765	244006	1.766926186
20	50	84747.77304	215811	1.546509392
20	75	160702.3242	370268	1.304061263
20	75	93458.45749	264180	1.826710467
20	100	172613.3825	531166	2.077200575
20	100	296368.8627	634956	1.142451789

Figure 3.10: Sample of results of graphs with 15-20 nodes

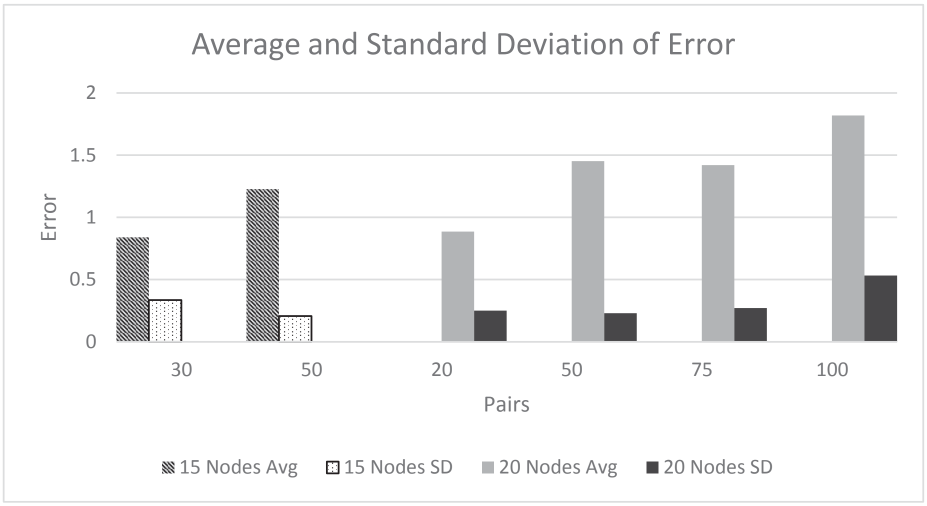


Figure 3.11: Average and standard deviation of error for Figure 3.10

3.3.2 Comparison of Heuristic to Arbitrary Tree

Figure 3.12 shows simulation results for the heuristic against variations of the $(d-1)$-ary tree for graphs of 10 to 20 nodes and a small number of pairs for $d = 3$. The variations are constructed as follows: The random tree column represents the solution from a $(d-1)$-ary tree that was constructed arbitrarily during the simulation. The complete tree was constructed from adding shortcut edges wherever possible to the random tree until the degree limit was reached for the maximum number of nodes in hopes of decreasing the cost. The sorted tree column represents a saturated $(d-1)$-ary tree that was constructed by sorting the nodes by the number of times they appear in P, and then constructing the tree such that the nodes that appear the most are placed closest to the top in an attempt to reduce cost to the nodes that are most commonly used. In general, the heuristic performs better than the sorted tree, which performs better than the complete tree. Figure 3.13 is a visual representation of the average performance of the various algorithms from Figure 3.12.

Nodes	Pairs	Heuristic	Random Tree	Complete Tree	Sorted Tree
10	25	40	60	46	51
10	25	39	81	56	47
12	30	49	87	63	59
12	30	52	94	74	65
15	50	100	181	141	137
15	50	99	176	128	139
20	100	228	393	316	346
20	100	228	404	327	277

Figure 3.12: Results of uniform graphs with 10-20 nodes

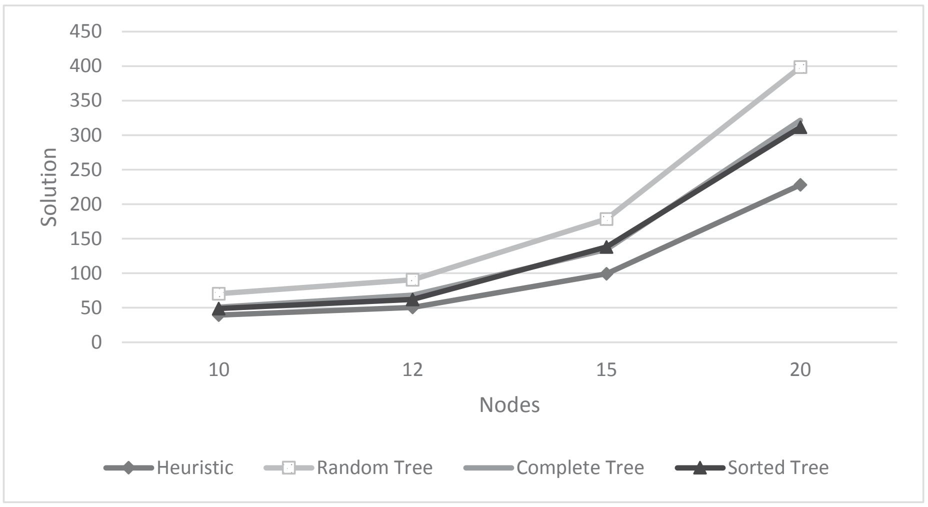


Figure 3.13: Performance of algorithms in Figure 3.12

Figure 3.14 shows simulation results for the heuristic against the $(d-1)$-ary tree for larger graphs of 30 to 70 nodes and all pairs for varying degrees. Since the problem requires all pairs, there is no difference between the sorted tree and the complete tree. The heuristic again performs better than both the random $(d-1)$-ary tree as well as the complete tree. Since the complete tree outperforms the random tree is all cases, the difference column is calculated from only the error from the complete tree to the heuristic. The difference varies between approximately 40 to 60 percent and generally increases with node size and degree. Figure 3.15 graphs the error from Figure 3.14 grouped by node size across increasing degree.

Nodes	Degree	Heuristic	Random Tree	Complete Tree	Difference
30	3	1270	2134	1845	0.452755906
30	4	1014	1722	1541	0.519723866
30	5	898	1543	1256	0.398663697
30	6	810	1397	1195	0.475308642
50	3	4364	7397	6619	0.516727773
50	4	3355	5842	5086	0.515946349
50	5	2944	5119	4549	0.54517663
50	6	2693	4658	4100	0.522465652
70	3	9575	16620	15067	0.573577023
70	4	7345	12672	11528	0.569503063
70	5	6316	11095	10207	0.616054465
70	6	5822	10238	9455	0.624012367

Figure 3.14: Results of uniform graphs with 30-70 nodes

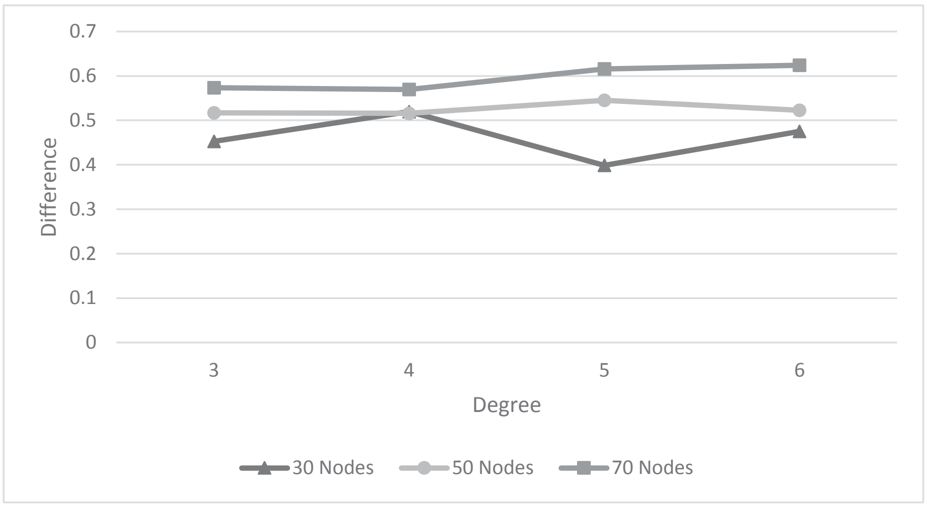
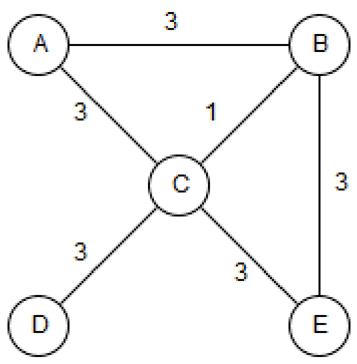


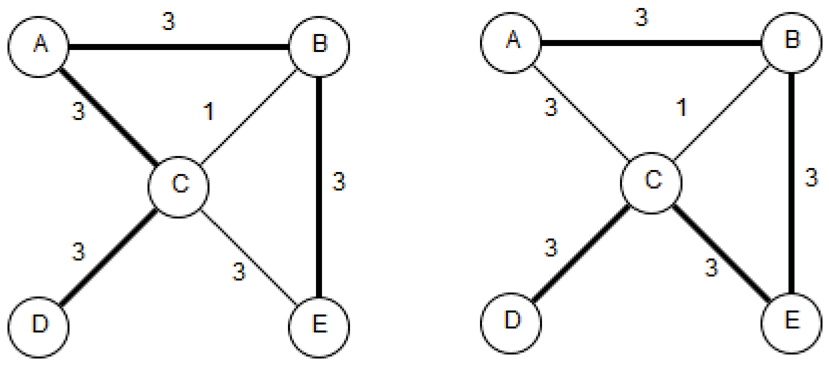
Figure 3.15: Difference from Figure 3.14 grouped by number of nodes

3.4 Other Variations

In the problem definition, we described our input as the complete graph derived from another graph using the shortest-path metric. For completeness, will show in this section how our heuristic performs poorly if the graph G' is not complete or it it does not satisfy triangle inequality.



(a) A non-complete graph G.



(b) Two possible solutions for MLA on G.

Figure 3.16: Solutions on incomplete graph

3.4.1 Incompleteness

Let Figure 3.16 (a) be an example of an incomplete graph. Suppose we are solving the MLA problem for this graph given the constraints that $d = 2$ and $P = (x, y)$ for all pairs $x, y \in V$. Figure 3.16 (b) shows the only two possible solutions for our example. For the solution subgraph to satisfy both the pair and degree constraints, four edges must be selected from the outermost layer of the graph. The edge (B, C) with the minimum weight cannot be selected.

Figure 3.17 shows an example of our algorithm failing on this incomplete graph. Since our algorithm uses a greedy heuristic, it will select (B, C) as the first edge and

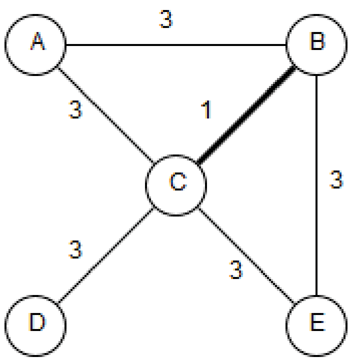


Figure 3.17: Example of heuristic failing on degree constraint.

will not be able to connect all pairs due to the degree constraint. There is no way to connect (A, C), (C, E), and (C, D) while keeping node C at a degree of 2.

3.4.2 Arbitrary Edge Lengths

Figure 3.18 shows an example of our algorithm performing poorly on a graph with arbitrary edge lengths. The graph in figure 4 is constructed from the graph in Figure 1 by adding some exponentially large edges represented by a length of 2^N. As in Figure 3, our algorithm will first select (B, C) to include in the final edge set. The algorithm is then forced to accept at least one of the "bad" edges, either (A, E), (A, D), or (D, E) to connect all pairs while maintaining the degree constraint. As a result, the magnitude of the difference between the solution of our algorithm and the optimal solution is exponential. This exponential difference between solutions can be avoided if all edge lengths satisfied the triangle inequality property as given in our problem definition.

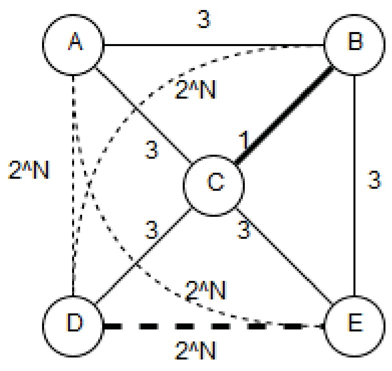


Figure 3.18: Example of heuristic's poor performance in a non-metric scenario.

3.5 Conclusion

We have shown that when $|P| = \Theta(n^2)$, a complete $(d-1)$-ary tree gives an asymptotically optimal solution. Through the simulations, however, it seems the greedy heuristic still outperforms the tree structure even on graphs with a large P. One reason for this difference may be due to the algorithmic construction process of the complete $(d-1)$-ary tree in simulations, as it added some edges that were selected randomly onto the random $(d-1)$-ary tree. Those shortcut edges that were selected may not have been the best choices for the tree. Since the heuristic greedily chooses every edge, it is reasonable for the solution of the heuristic to outperform that of the complete tree even when they both contain the same number of edges. As shown by the simulation results, the optimal bound on the solution is an asymptotic optimal bound; it does not guarantee the exact optimal solution given a specific problem. Future work may include revising the method for adding extra edges in the complete $(d-1)$-ary tree to improve upon the cost of its solutions. We can also run simulations with larger graph sizes to determine the threshold for the performance of the $(d-1)$-ary tree construction. We can vary the size of P to determine the ratio of performance of the $(d-1)$-ary tree to the heuristic for smaller sizes of P. Additionally, computing an accurate lower bound for the tree construction would also provide more context for the comparisons.

Chapter 4

ON MINIMAL STEINER TREES IN A COMPLETE GRAPH

4.1 Introduction

In a number of networking problems one needs to find multiple paths between a source-destination node pair or multiple trees spanning all the nodes of the network. Accordingly a number of algorithms for generating k shortest paths between a specified source-destination node pair or k best spanning trees connecting all the nodes of a network are known [15]. An underlying assumption in these algorithms is that k is at most equal to the number of paths that exists between a specified source-destination node pair, or k is at most equal to the number of spanning trees that exists in the graph. Cayley's formula provides the number of labeled Spanning Trees of a complete graph [56]. As such if one wants to generate k spanning trees it can easily be checked if that will be feasible. However, to the best of our knowledge no such formula exists for (i) the number of paths between a source-destination node pair in a complete graph, and (ii) the number of Steiner Trees with p terminal nodes in a complete graph of n nodes.

To model multicasting in Elastic Optical Networks (EON), Walkowiak *et. al.* in [59] have developed the notion of *candidate trees*, which is basically a set of Steiner trees that originate at the source node and include all the receiver nodes. In this setting one needs to generate k-shortest Steiner trees for solving optimization problems in EONs, both in static and dynamic environment. As there is no counter-part of Cayley's formula for Steiner Trees, the effort to generate k-shortest Steiner trees is bound to fail if k is greater than the number of Steiner Trees with p terminal nodes in a complete graph of n nodes.

We undertook this study with a goal to find the counter-part of Cayley's formula for Steiner Trees, i.e., a function $ST(n, p)$ which provides the number of Steiner trees in a complete graph with n nodes and the cardinality of the set of terminal nodes is p. Although we didn't completely reach our goal, we believe that we made significant progress in that direction. Specifically, we (i) found a closed form expression for $ST(n, p)$ when $p = 2$ and $p = 3$ (In other words, we reached our goal of finding the counter part of Cayley's formula for Steiner trees when the cardinality of the set of terminal nodes is 2 or 3), (ii) developed two formulations, one recursive and one non-recursive that provides the number of Steiner trees in a complete graph with n nodes and p terminal nodes, (ii) developed (based of one of the formulations in (ii)), an algorithm that not only computes the number of Steiner trees in a complete graph with n nodes and p terminal nodes, but also generates all those Steiner trees. In the following sections we present (i), (ii) and (iii).

It may be noted that although a counter-part of Cayley formula for Steiner Tree is not known, there are algorithms to generate all Steiner Trees of a given graph [13, 25]. Obviously, the *number* of Steiner Trees can be computed by enumerating all the Steiner Trees. However, as the number of Steiner Trees can be very large, the complexity of such algorithms in general is quite high. For example, the complexities of computation of finding the *number* of Steiner Trees in a graph as is given in [25] are $O(n^3 2^{n-p})$ and $O(nm + m3^p)$, where n, m, p are the number of nodes, edges and terminal nodes of the graph respectively. While both of these algorithms are of exponential time complexity, the complexity of finding the number of Steiner Trees in a complete graph with n nodes and p terminal nodes using Algorithm 1 presented in this chapter is $O(np)$. Douardo *et.al.* in [13] have provided an algorithm for generating all the Steiner Trees. Their algorithm produces the first Steiner Tree in polynomial time and all subsequent trees are produced with $O(n)$ delay [13]. As the number of Steiner Trees in a complete graph can be exponential, if this algorithm is used to compute the number of Steiner Trees in a complete graph with n nodes and p terminal nodes, the complexity of this algorithm will also be exponential.

4.2 Computation of $ST(n,p)$ when $p=2$

As indicated earlier, in many networking applications one needs to find k (shortest) paths between a source-destination node pair. However, to the best of our knowledge there is not known simple expression that provides the number of distinct paths that exist between a pair of nodes in a complete graph with n nodes. The expression $ST(n,p)$ when $p=2$ presented here provides that number.

Theorem 4.

$$ST(n,2) = \begin{cases} 1 & \textit{if } \; n = 2 \\ \lfloor (n-2)!e \rfloor & \textit{if } \; n > 2 \end{cases}$$

Proof. When $n=2$, it is trivial to see that $ST(n,2)=1$. For any tree with $n>2$ nodes has at least 2 leaves. As all leaves in a Steiner Tree must be terminal nodes, and when the number of terminal nodes $p=2$, these are are only two leaves of the tree. The internal nodes of the tree are non-terminal nodes and in this case the tree is a *chain*. The number of internal (i.e., non-terminal) nodes can vary from 0 to $n-2$. In order to compute $ST(n,2)$, we need to account for all these possibilities for the number of internal nodes. Suppose that the number of internal nodes (i.e., nodes with degree 2) is t. There are $\binom{n-2}{t}$ options for the selection of the internal nodes and for each option, there are $t!$ permutations of their positions in the chain. Accordingly, we have

$$\begin{aligned} ST(n,2) &= \sum_{t=0}^{n-2} \binom{n-2}{t} \cdot t! = \sum_{t=0}^{n-2} \frac{(n-2)!}{t! \cdot (n-2-t)!} \cdot t! \\ &= \sum_{t=0}^{n-2} \frac{(n-2)!}{(n-2-t)!} = (n-2)! \sum_{t=0}^{n-2} \frac{1}{t!} \\ &= (n-2)! \sum_{t=0}^{\infty} \frac{1}{t!} - (n-2)! \sum_{t=n-1}^{\infty} \frac{1}{t!} \end{aligned} \tag{4.1}$$

Using identity

$$e = \sum_{t=0}^{\infty} \frac{1}{t!}$$

we have

$$ST(n,2) = (n-2)!e - (n-2)! \sum_{t=n-1}^{\infty} \frac{1}{t!}$$

. Since $ST(n,2)$ is an integer, it suffices to show $(n-2)! \sum_{t=n-1}^{\infty} \frac{1}{t!} < 1$.

$$(n-2)! \sum_{t=n-1}^{\infty} \frac{1}{t!} = \sum_{t=n-1}^{\infty} \frac{1}{(n-1) \cdot n \cdot ... \cdot t} < \sum_{t=1}^{\infty} \frac{1}{2^t} = 1 \tag{4.2}$$

Accordingly, we have $ST(n,2) = \lfloor (n-2)!e \rfloor$ $\square$

We will use $\sum_{x=0}^{n} \frac{n!}{x!}$ very often in the following content, thus we denote $g(n) = \sum_{x=0}^{n} \frac{n!}{x!}$. From above computation, we have $g(n) = 1$ if $n = 0$ and $g(n) = \lfloor n!e \rfloor$ for $n \geq 1$.

4.3 Computation of $ST(n,p)$ when $p=3$

It was noted in the previous section that when $p=2$, the structure of the Steiner Tree is a chain. When $p=3$, we can have two different possibilities: (i) one of the three terminal nodes is an *internal* node of the tree, (ii) none of the three terminal nodes is an *internal* node of the tree. In case (i) the Steiner tree will be made up of two chains, connecting the two terminal nodes that are leaves to the one terminal node which is an internal node of the tree. In case (ii)the Steiner tree will be made up of three chains, connecting the three terminal nodes that are leaves to the one non-terminal node that serves as the center of a Star structured Steiner tree. As a result, the total number of Steiner trees when $p=3$ is equal to the sum of the number of Steiner trees that are created in case (i) and the number of Steiner trees that are created in case (ii). We compute the number of Steiner trees in these two cases separately.

4.3.1 Computation of $ST(n,3)$ when the Steiner tree has a two chain structure

In this case, one of the terminal nodes is an internal node. There are $\binom{3}{1}$ options for the selection of the terminal node that will be an internal node. Suppose that the number of non-terminal internal nodes is $t, 0 \leq t \leq n-3$. There are $\binom{n-3}{t}$ options for the selection of these internal nodes. Out of these t nodes, p nodes can be chosen in $\binom{t}{p}$ to form the first chain and the remaining $(t-p)$ nodes can form the second chain. The nodes in the first chain can be permuted in $p!$ different ways and the nodes in the second chain can be permuted in $(t-p)!$ different ways. By summing over all values of t from 0 to $n-3$, we obtain the number of Steiner trees for case (i). Thus

$$ST(n,3)_{case\ (i)} = \binom{3}{1} \sum_{t=0}^{n-3} \sum_{p=0}^{t} \binom{n-3}{t} \binom{t}{p} p!(t-p)!$$

After simplification

$$ST(n,3)_{case\ (i)} = 3m! \left[\sum_{t=0}^{m} \frac{t}{(m-t)!} + \sum_{t=0}^{m} \frac{1}{(m-t)!} \right]$$

where $m = n - 3$.

4.3.2 Computation of $ST(n, 3)$ when the Steiner tree has a star structure

In this case, none of the three terminal nodes is an internal node. One of the remaining $(n-3)$ nodes serves as the *center* of the star. There are $\binom{n-3}{1}$ options for the selection of the non-terminal node that will serve as the center of the star. The structure of the Steiner Tree in this case is composed of three chains, where each chain is a path from a terminal node to the center node. The nodes that appear on these chains are non-terminal nodes. Suppose that the number of non-terminal internal nodes that appear on these three chains (exclude the non-terminal node which is the center of teh star structure) is $t, 0 \leq t \leq n-4$. There are $\binom{n-4}{t}$ options for the selection of these internal nodes. Out of these t nodes, r nodes can be chosen in $\binom{t}{r}$ to form the first chain and the remaining $(t-r)$ nodes can form the second chain. The nodes in the first chain can be permuted in $t!$ different ways and the nodes in the second chain can be permuted in $(t-r)!$ different ways. By summing over all values of t from 0 to $n-3$, we obtain the number of Steiner trees for case (i). Thus

$$ST(n,3)_{case\ (ii)} = \sum_{t=0}^{n-4} \sum_{p=0}^{t} \sum_{q=0}^{t-p} \binom{n-3}{1} \binom{n-4}{t} \binom{t}{p} \binom{t-p}{q} p!q!(t-p-q)!$$

After simplification

$$ST(n,3)_{case\ (ii)} = \frac{1}{2}(m'+1)! \left[\sum_{t=0}^{m'} \frac{t^2}{(m'-t)!} + 3 \sum_{t=0}^{m'} \frac{t}{(m'-t)!} + 2 \sum_{t=0}^{m'} \frac{1}{(m'-t)!} \right]$$

where $m' = n - 4$.

In order to compute $ST(n,3)_{case\ (i)}$ and $ST(n,3)_{case\ (ii)}$, we need to know the values of $\sum_{t=0}^{r}\frac{t^2}{(r-t)!}$, $\sum_{t=0}^{r}\frac{t}{(r-t)!}$ and $\sum_{t=0}^{r}\frac{1}{(r-t)!}$ and set $r=m$ for computation of $ST(n,3)_{case\ (i)}$ and $r=m'$ for $ST(n,3)_{case\ (ii)}$. The computation of these values are presented in the Appendix section.

4.3.3 Computation of $ST(n,3)$

$$\begin{aligned}ST(n,3) &= ST(n,3)_{case\ (i)} + ST(n,3)_{case\ (ii)}\\ &= 3m!\left[\sum_{t=0}^{m}\frac{t}{(m-t)!}+\sum_{t=0}^{m}\frac{1}{(m-t)!}\right]+\\ &\quad \frac{1}{2}(m'+1)!\left[\sum_{t=0}^{m'}\frac{t^2}{(m'-t)!}+3\sum_{t=0}^{m'}\frac{t}{(m'-t)!}+2\sum_{t=0}^{m'}\frac{1}{(m'-t)!}\right]\end{aligned}$$

where $m=m'+1=n-3$.

4.3.4 Computation of $\sum_{t=0}^{r}\frac{1}{(r-t)!}$

In section 4.2 we have shown that $\sum_{t=0}^{n-2}\frac{(n-2)!}{(n-2-t)!}=g(n-2)$. Replacing $(n-2)$ by r, we have $\sum_{t=0}^{r}\frac{r!}{(r-t)!}=g(r)$, or $\sum_{t=0}^{r}\frac{1}{(r-t)!}=\frac{g(r)}{r!}$

4.3.5 Computation of $\sum_{t=0}^{r}\frac{t}{(r-t)!}$

$$\sum_{t=0}^{r}\frac{r!t}{(r-t)!} = \sum_{t=0}^{r}\frac{r!}{t!}(r-t) = rg(r)-\sum_{t=0}^{r}\frac{r!t}{t!}$$

$$\sum_{t=0}^{r}\frac{r!t}{t!} = \sum_{t=0}^{r}\frac{r!}{(t-1)!} = \sum_{t=0}^{r-1}\frac{r!}{t!} = g(r)-1$$

Easy to see, we also have:

$$\sum_{t=0}^{r-1}\frac{r!}{t!} = rg(r-1)$$

Hence,

$$\sum_{t=0}^{r} \frac{t}{(r-t)!} = \frac{1}{r!}(rg(r) - g(r) + 1)$$

4.3.6 Computation of $\sum_{t=0}^{r} \frac{t^2}{(r-t)!}$

$$\begin{aligned} \sum_{t=0}^{r} \frac{r!}{(r-t)!}t^2 &= \sum_{t=0}^{r} \frac{r!}{t!}(r-t)^2 \\ &= \sum_{t=0}^{r} \frac{r!}{t!}(r^2 - 2rt + t^2) \\ &= r^2 g(r) - 2r(g(r) - 1) + \sum_{r=0}^{r} \frac{r!}{t!}t^2 \end{aligned} \tag{4.3}$$

$$\begin{aligned} \sum_{t=0}^{r} \frac{r!}{t!}t^2 &= \sum_{t=0}^{r} \frac{r!}{(t-1)!}t \\ &= \sum_{t=0}^{r-1} \frac{r!}{t!}t + \sum_{t=0}^{r-1} \frac{r!}{t!} \\ &= r(g(r-1) - 1) + rg(r-1) \\ &= 2rg(r-1) - r \\ &= 2g(r) - r - 2 \end{aligned} \tag{4.4}$$

Thus, $\sum_{t=0}^{r} \frac{t^2}{(r-t)!} = \frac{1}{r!}\left[r^2 g(r) - 2rg(r) + 2g(r) + r - 2\right]$

Theorem 5.

$$ST(n,3) = 3(n-2)g(n-2) + \frac{n-3}{2}[(n^2 - 10n + 13)g(n-4) + n - 3] + 3$$

4.4 Computation of $ST(n,p)$ when $4 \le p \le n$

In this section, we provide an algorithm to find the total number of minimal Steiner trees with n nodes and p terminal nodes. The algorithm is based on Prüfer sequence [56] associated with a tree. If the nodes of a tree is numbered from 1 to n, the Prüfer sequence associated with tree is a unique sequence integers of size $n - 2$. The leaf nodes of the tree do not appear in the sequence while each interior node appears at least once.

Cayley's formula that provides the number of spanning trees in a complete graph with n nodes as n^{n-2} can be derived using the Prüfer sequence. A Steiner tree of a graph is defined to be a tree that connects the set of specified *terminal* or required nodes in the graph. We will refer to a Steiner tree as a *minimal* Steiner tree if removal of any node of the tree fails to connect the set of specified terminal nodes. We assume that the nodes of the complete graph is labeled from 1 to n and a size of the specified set of terminal nodes is p. Suppose that a minimal Steiner tree connecting the set of p terminals has t nodes and k leaves, where $t \geq p$ and $2 \leq k \leq p$. The size of the Prüfer sequence corresponding to this tree is $t - 2$ and the the elements of the sequence are the labels of the internal nodes of the tree. It may be noted that for any $t, p \leq t \leq n$ there are $\binom{n-p}{t-p}$ ways of selecting the non-terminal nodes. In addition, there are $\binom{p}{k}$ ways to select leaf nodes. Next we focus on the number of different Prüfer sequences that can be generated from the Steiner trees, subject to the constraints stated earlier. Clearly, it is a surjection from $t - 2$ positions to $t - k$ indices.

After obtaining the size, next we need to consider the configuration of the Prüfer sequence. Clearly, it is a surjection from $t - 2$ positions to $t - k$ integers (labels on the nodes). Let $T(n, m)$ be the total number of surjections from n positions to m integers, then $T(n, m) = m!S(n, m)$,where $S(n, m)$ is the Stirling number of the second kind, which provides the number of different ways of partitioning n elements into m nonempty sets. Finally, let $ST(n, p)$ represent the total number of minimal Steiner trees of a graph with n nodes and p terminals. From the arguments above, we have

$$\begin{aligned} ST(n,p) &= \sum_{t=p}^{n} \binom{n-p}{t-p} \sum_{k=2}^{p} \binom{p}{k} T(t-2, t-k) \\ &= \sum_{t=p}^{n} \binom{n-p}{t-p} \sum_{k=2}^{p} \binom{p}{k} (t-k)! S(t-2, t-k) \end{aligned} \tag{4.5}$$

We can efficiently compute $S(n,m)$ efficiently by using the recurrence relation $S(n,m) = mS(n-1,m) + S(n-1,m-1)$.

In the following we provide the algorithm for computing $ST(n,p)$,

Computational complexity of the algorithm is $O(np)$.

4.5 Prüfer Sequence Generating Algorithm

It may be noted that Algorithm 1 only computes the number of minimal Steiner trees in a complete graph with n nodes and p terminals. In most of the applications, such as the one needed in Elastic Optical Networks, one need to know not only the number of Steiner trees that exist, but also a way to compute these trees. Accordingly, we also provide an algorithm for generating all Prüfer Sequences corresponding to the minimal Steiner Trees.

Once we have the Prüfer Sequence, the corresponding minimal Steiner tree can easily be generated following the algorithm given in [56]. W.l.o.g, we label the terminal nodes from 1 to p. Similar to Algorithm 1, we enumerate the number of nodes, t, as well as the number of leaves, k, among p terminals. Then, we partition $t-2$ into $t-k$ non-empty groups and generate all permutations of such $t-k$ groups. For any partition, we consider elements in each group forming a string. After that, we can sort groups lexicographically based on their representation strings. Next, we generate all permutations of interior node labels and match each node label to the corresponding position in the group.

Algorithm 5 Algorithm for computing the number of minimal Steiner trees with n nodes and p terminals

```
1: $S(0,0) = 1$
2: for $t = 1$ to $n$ do
3:     for $k = 1$ to $p$ do
4:         $S(t,k) = k \cdot S(t-1,k) + S(t-1,k-1)$
5:     end for
6: end for
7: for $t = 1$ to $n$ do
8:     for $k = 1$ to $p$ do
9:         $T(t,k) = k! \cdot S(t,k)$
10:    end for
11: end for
12: for $t = p$ to $n$ do
13:    for $k = 2$ to $p$ do
14:        $ST(n,p) += \binom{n-p}{t-p} \cdot \binom{p}{k} \cdot T(t-2, t-k)$
15:    end for
16: end for
17: return $ST(n,p)$
```

Algorithm 6 Algorithm for generating all Prüfer Sequences

```
1: $result\_set = \emptyset$
2: for $t = p$ to $n$ do
3:     for $k = 2$ to $p$ do
4:         choose $p - k$ numbers from [1,p] and choose $t - p$ numbers from [p+1,n],
   let the set be $U = \{u_1, u_2, ..., u_{t-k}\}$.
5:         for each partition of {1,2,...,t-2} into t-k groups do
6:             sort partition group lexicographically, let $g_i$ stands for group at $i - th$
   position.
7:             for each permutation of $U$ do
8:                 Let $A$ be a sequence of length $t - 2$.
9:                 for $i = 1$ to $t - k$ do
10:                    for each $pos$ in $g_i$ do
11:                        set $A_{pos} = u_i$.
12:                    end for
13:                end for
14:                $result\_set = result\_set \cup A$
15:            end for
16:        end for
17:    end for
18: end for
19: return $result\_set$
```

It takes $O(n)$ to generate each Prüfer Sequence and it takes $O(n)$ to convert one Prüfer Sequence to a Steiner tree.

Chapter 5

ON SHORTEST SINGLE/MULTIPLE PATH COMPUTATION PROBLEMS IN FIBER-WIRELESS (FIWI) ACCESS NETWORKS

5.1 Introduction

Path computation problems are arguably one of the most well studied family of problems in communication networks. In most of these problems, one or more weight is associated with a link representing, among other things, the *cost, delay* or the *reliability* of that link. The objective most often is to find a *least weighted path* (or "shortest path") between a specified source-destination node pair. In most of these problems, if a link l is a part of a path P, then the contribution of the link l on the "length" of the path P depends only on the weight $w(l)$ of the link l, and is oblivious of the weights of the links traversed before or after traversing the link l on the path P. However, in a recent paper on optical-wireless FiWi network [3], the authors have proposed a path length metric, where the contribution of the link l on the "length" of the path P depends not only on its own weight $w(l)$, but also on the weights of all the links of the path P. As the authors of [3] do not present any algorithm for computing the shortest path between the source-destination node pair using this new metric, we present a polynomial time algorithm for this problem in this chapter. This result is interesting because of the nature of new metric proposed in [3], one key property on which the shortest path algorithm due to Dijkstra is based, that is, *subpath of a shortest path is shortest*, is no longer valid. We show that even without this key property, not only it is possible to compute the shortest path in polynomial time using the new metric, it is also possible to compute the shortest path in polynomial time, with a variation of the metric proposed in [3].

The rest of the chapter is organized as follows. In section 5.3, we present the path length metric proposed for the FiWi network in [3] and a variation of it. In section 5.4 we provide algorithms for computing the shortest path using these two metrics. As multi-path routing offers significant advantage over single path routing [50, 42, 60, 38], we also consider the problem of computation of a pair of node disjoint paths between a source-destination node pair using the metric proposed in [3]. We show that while the single path computation problem can be solved in polynomial time in all these cases, the disjoint path computation problem is NP-complete. The contribution of the chapter are as follows;

- Polynomial time algorithm for single path routing (metric 1) in FiWi networks

- Polynomial time algorithm for single path routing (metric 2) in FiWi networks

- NP-completeness proof of disjoint path routing (metric 1) in FiWi networks

- Optimal solution for disjoint path routing (metric 1) in FiWi networks using Integer Linear Programming

- One approximation algorithm for disjoint path routing in FiWi networks with an approximation bound of 4 and computation complexity $O((n + m)log\ n)$

- One approximation algorithm for disjoint path routing in FiWi networks with an approximation bound of 2 and computation complexity $O(m(n + m)log\ n)$

- Experimental evaluation results of the approximation algorithm for disjoint path routing in FiWi networks

5.2 Related Work

The Fiber-Wirelss (FiWi) network is a hybrid access network resulting from the convergence of optical access networks such as Passive Optical Networks (PONs) and wireless access networks such as Wireless Mesh Networks (WMNs) capable of providing low cost, high bandwidth last mile access. Because it provides an attractive way of integrating optical and wireless technology, Fiber-Wireless (FiWi) networks have received considerable attention in the research community in the last few years [20, 19, 64, 63, 3, 60, 38]. The minimum interference routing algorithm for the Fi-Wi environment was first proposed in [63]. In this algorithm the path length was measured in terms of the number of hops in the wireless part of the FiWi network. The rationale for this choice was that the maximum throughput of the wireless part is typically much smaller than the throughput of the optical part, and hence minimization of the wireless hop count should lead to maximizing the throughout of the FiWi network. However, the authors of [3] noted that minimization of the wireless hop count does not always lead to throughput maximization. Accordingly, the path length metric proposed by them in [3] pays considerable importance to the traffic intensity at a generic FiWi network node. The results presented in this chapter are motivated by the path length metric proposed in [3].

5.3 Problem Formulation

In the classical path problem, each edge $e \in E$ of the graph $G = (V, E)$, has a weight $w(e)$ associated with it and if there is a path P from the node v_0 to v_k in the graph $G = (V, E)$

$$v_0 \overset{w_1}{\rightarrow} v_1 \overset{w_2}{\rightarrow} v_2 \overset{w_3}{\rightarrow} v_3 \ldots \overset{w_k}{\rightarrow} v_k$$

then the *path length* or the *distance* between the nodes v_0 and v_k is given by

$$w(P_{v_0,v_k}) = w_1 + w_2 + \cdots + w_k$$

However, in the path length metric proposed in [3] for optical-wireless FiWi networks [20, 19, 64], the contribution of e_i to the path length computation depends not only on the weight w_i, but also on the weights of the other edges that constitute the path. In the following section, we discuss this metric and a variation of it. We also also formulate the multipath computation problem using this metric.

The *Optimized FiWi Routing Algorithm (OFRA)* proposed in [3] computes the "length" (or weight) of a path P from v_0 to v_k using the following metric

$$w'(P_{v_0,v_k}) = \min_P \left(\sum_{\forall u \in P} (w_u) + \max_{\forall u \in P} (w_u) \right)$$

where w_u represents the traffic intensity at a generic FiWi network node u, which may be an optical node in the fiber backhaul or a wireless node in wireless mesh front-end. In order to compute shortest path using this metric, in our formulation, instead of associating a traffic intensity "weight" (w_u) with nodes, we associate them with edges. This can easily be achieved by replacing the node u with weight w_u with two nodes u_1 and u_2, connecting them with an edge (u_1, u_2) and assigning the weight w_u on this edge. In this scenario, if there is a path P from the node v_0 to v_k in the graph $G = (V, E)$

$$v_0 \overset{w_1}{\rightarrow} v_1 \overset{w_2}{\rightarrow} v_2 \overset{w_3}{\rightarrow} \ldots \overset{w_k}{\rightarrow} v_k$$

then the *path length* between the nodes v_0 and v_k is given by

$$\begin{aligned} w^{+}(P_{v_0,v_k}) &= w_1 + w_2 + \ldots + w_k + \mathtt{max}(w_1, w_2, \ldots w_k) \\ &= \sum_{i=1}^{k} w_i + \mathtt{max}_{i=1}^{k} w_i \end{aligned}$$

In the second metric, the length a path $P_{v_0,v_k} : v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_k$, between the nodes v_0 and v_k is given by

$$\begin{aligned} \bar{w}(P_{v_0,v_k}) &= \sum_{i=1}^{k} w_i + CNT(P_{v_0,v_k}) * \mathtt{max}(w_1, w_2, \ldots w_k) \\ &= \sum_{i=1}^{k} w_i + CNT(P_{v_0,v_k}) * \mathtt{max}_{i=1}^{k} w_i \end{aligned}$$

where $CNT(P_{v_0,v_k})$ is the count of the number of times max $(w_1, w_2, \ldots w_k)$ appears on the path P_{v_0,v_k}. We study the shortest path computation problems in FiWi networks using the above metrics and provide polynomial time algorithms for solution in subsections IV-A and IV-B.

If $w_{max} = \mathtt{max}(w_1, w_2, \ldots w_k)$, we refer to the corresponding edge (link) as e_{max}. If there are multiple edges having the weight of w_{max}, we arbitrarily choose any one of them as e_{max}. It may be noted that both the metrics have an interesting property in that in both cases, the contribution of an edge e on the path length computation depends not only on the edge e but also on every other edge on the path. This is so, because if the edge e happens to be e_{max}, contribution of this edge in computation of $w^{+}(P_{v_0,v_k})$ and $\bar{w}(P_{v_0,v_k})$ will be $2 * w(e)$ and $CNT(P_{v_0,v_k}) * w(e)$ respectively. If e is not e_{max}, then its contribution will be $w(e)$ for both the metrics.

As multipath routing provides an opportunity for higher throughput, lower delay, and better load balancing and resilience, its use have been proposed in fiber networks [50], wireless networks [42] and recently in integrated fiber-wireless networks

[60, 38]. Accordingly, we study the problem of computing a pair of edge disjoint paths between a source-destination node pair s and d, such that *the length of the longer path (path length computation using the first metric) is shortest among all edge disjoint path pairs between the nodes s and d.* In subsection IV-C we prove that this problem is NP-complete, in subsection IV-D, we provide an optimal solution for the problem using integer linear programming, in subsections IV-E and IV-F we provide two approximation algorithms for the problem with a performance bound of 4 and 2 respectively, and in subsection IV-F we provide results of experimental evaluation of the approximation algorithms.

5.4 Path Problems in FiWi Networks

In this section, we present (i) two different algorithms for shortest path computation using two different metrics, (ii) NP-completeness proof for the disjoint path problem, (iii) two approximation algorithms for the disjoint path problem, and (iv) experimental evaluation results of the approximation algorithms.

It may be noted that, in both metrics $w^+(P_{v_0,v_k})$ and $\bar{w}(P_{v_0,v_k})$, we call an edge $e \in P_{v_0,v_k}$ *crucial*, if $w(e) = max_{i=1}^{k} w(e'), \forall e' \in P_{v_0,v_k}$.

5.4.1 Shortest Path Computation using Metric 1

It may be recalled that the path length metric used in this case is the following: $w^+(P_{v_0,v_k}) = \sum_{i=1}^{k} w_i + \texttt{max}_{i=1}^{k} w_i$. If the path length metric was given as $w(P_{v_0,v_k}) = \sum_{i=1}^{k} w_i$, algorithms due to Dijkstra and Bellman-Ford could have been used to compute the shortest path between a source-destination node pair. One important property of the path length metric that is exploited by Dijkstra's algorithm is that "subpath of a shortest path is shortest". However, the new path length metric $\sum_{i=1}^{k} w_i + \texttt{max}_{i=1}^{k} w_i$ does not have this property. We illustrate this with the example below.

Consider two paths P_1 and P_2 from the node v_0 to v_3 in the graph $G = (V, E)$, where $P_1 : v_0 \overset{w_1}{\to} v_1 \overset{w_2}{\to} v_2 \overset{w_3}{\to} v_3$ and $P_2 : v_0 \overset{w_4}{\to} v_4 \overset{w_5}{\to} v_2 \overset{w_3}{\to} v_3$. If $w_1 = 0.25, w_2 = 5, w_3 = 4.75, w_4 = 2, w_5 = 4$, the length of the path P_1, $PL_1 = w_1 + w_2 + w_3 + \texttt{max}(w_1, w_2, w_3) = 0.25 + 5 + 4.75 + \texttt{max}(0.25, 5, 4.75) = 15$ and the length of the path P_2, $PL_2 = w_4 + w_5 + w_6 + \texttt{max}(w_4, w_5.w_6) = 2 + 4 + 4.75 + \texttt{max}(2, 4, 4.75) = 15.5$. Although P_1 is shortest path in this scenario, the length of its subpath $v_0 \overset{w_1}{\to} v_1 \overset{w_2}{\to} v_2$ is 0.25 + 5 + $\texttt{max}$ (0.25, 5) = 10.25, which is greater than the length of a subpath of P_2 $v_0 \overset{w_4}{\to} v_4 \overset{w_5}{\to} v_2$ 2 + 4 + $\texttt{max}$ (2, 4) = 10, demonstrating that the assertion that “subpath of a shortest path is shortest” no longer holds in this path length metric.

As the assertion “subpath of a shortest path is shortest” no longer holds in this path length metric, we cannot use the standard shortest path algorithm due to Dijkstra in this case. However, we show that we can still compute the shortest path between a source-destination node pair in polynomial time by repeated application of the Dijkstra’s algorithm. The algorithm is described next.

For a given graph $G = (V, E)$, w.l.o.g, we assume $|V| = n$ and $|E| = m$. Define G_e as subgraph of G by deleting edges whose weight is greater than $w(e)$.

Also, as Dijkstra’s algorithm does, we need to maintain distance vector. We define $dist_v$ be distance (length of shortest path) from s to v, Π_v be predecessor of v and $maxedge_v$ be weight of the crucial edge from s to v via the shortest path, ans_v be optimal solution (length) from s to v.

Different G_e can be treated as different layers of the G. For any path P, we define the function $e^*(P)$ as the crucial edge along P. It is easy to observe that if P_d is the optimal path from s to node d then $w(P_d) = \sum_{e \in P} w(e)$ and $w^+(P_d) = w(P_d) + w(e^*(P_d))$. It may be noted that henceforth, we shorten $P_{s,d}$ to P_d, because we consider that the source is fixed while the destination d is variable.

Algorithm 7 Modified Dijkstra's Algorithm

```
1: Initialize $ans_v = \infty$ for for all $v \in V$
2: sort all edges according to $w(e)$ in ascending order
3: for $i = 1$ to $m$ do
4:     Initialize $dist_v = \infty, \Pi_v = nil,\ maxedge_v = 0$ for all $v \in V$
5:     $dist_s = 0$
6:     $Q =$ the set of all nodes in graph
7:     while Q is not empty do
8:         $u =$ Extract-Min($Q$)
9:         for each neighbor $v$ of $u$ do
10:            if $e_{u,v} \in E(G_{e_i})$ then
11:                $t =$ MAX $\{maxedge_u\ ,\ w(e_{u,v})\}$
12:                if $dist_u + w(e_{u,v}) < dist_v$ then
13:                    $dist_v = dist_u + w(e_{u,v})$
14:                    $maxedge_v =$ t
15:                    $\Pi_v = u$
16:                else if $dist_u + w(e_{u,v}) == dist_v$ then
17:                    if $maxedge_v > t$ then
18:                        $maxedge_v = t$
19:                        $\Pi_v = u$
20:                    end if
21:                end if
22:            end if
23:        end for
24:    end while
```

Algorithm 8 Modified Dijkstra's Algorithm

```
25:     for each node $v$ do
26:         $ans_v = \min\{ans_v,\ dist_v + maxedge_v\}$
27:     end for
28: end for
```

Lemma 1. *$w(P_d)$ is minimum in $G_{e^*(P_d)}$.*

Proof. It is obvious that P_d still exists in $G_{e^*(P_d)}$, since edges on P_d are not abandoned. Suppose P_d is not shortest, then there must be another path $P_{d'}$ s.t. $w(P'_d) < w(P_d)$. Noting that the crucial edge on P'_d, namely e', is no longer than $e^*(P_d)$ since they both belong to $G_{e^*(P_d)}$. Hence $w^+(P'_d) = w(P'_d) + w(e') < w(P_d) + w(e^*(P_d)) = w^+(P_d)$, contradicting P_d is optimal. □

Lemma 2. *Modified Dijkstra's Algorithm (MDA) computes shortest path while keeping the crucial edge as short as possible in every iteration.*

Proof. Line 4 to 24 works similar to the standard Dijkstra's algorithm does. Besides, when updating distance, MDA also updates the crucial edge to guarantee that it lies on the path and when there is a tie, MDA will choose the edge with the smaller weight. □

Theorem 1. *Modified Dijkstra's Algorithm computes optimal solution for every node v in $O(m(n+m)logn)$ time.*

Proof. Lemma 1 indicates for any node $v \in V$, optimal solution can be obtained by enumerating all possible crucial edges $e^*(P_v)$ and computing shortest path on $G_{e^*(P_v)}$. By sorting all edges in nondecreasing order, every subgraph $G_{e^*(P_v)}$ is considered and it is shown in lemma 2, MDA correctly computes shortest path for every node v in every $G_{e^*(P_v)}$. Then optimal solution is obtained by examining all shortest path using the $w()$ metric plus the corresponding crucial edge. Dijkstra's algorithm runs $O((n+m)logn)$ time when using binary heap, hence MDA runs in $O(m(n+m)logn)$ time when considering all layers. $\square$

5.4.2 Shortest Path Computation using Metric 2

Given a path P, let $e^*(P)$ be the crucial edge along the P and $CNT(P)$ be the number of occurrence of such edge. Now our objective becomes to find a path Q, such that $\bar{w}(P) = \sum_{e \in Q} w(e) + CNT(Q) * w(e^*(Q))$ is minimum.

The layering technique can also be used in this problem. However, shortest path under a ceratin layer may not become a valid candidate for optimal solution. Here, we introduce a dynamic programming algorithm that can solve the problem optimally in $O(n^2m^2)$ time.

Input is a weighted graph $G = (V, E), |V| = n, |E| = m$ with a specified source node s. In this chapter, we only consider nonnegative edge weight. As shown before, we use G_e to represent the residue graph by deleting edges longer than e in G. Different from **MDA1**, in order to consider the number of crucial edges, $dist_v$ is replaced by an array $dist_v^0, dist_v^1, ...dist_v^n$. One can think $dist_v^c$ be the shortest distance from s to v by going through exactly c crucial edges and possibly some shorter edges. Similarly, we replace Π_v by $\Pi_v^c, 0 \le c \le n$. Each Π_v^c records predecessor of v for the path corresponding to $dist_v^c$. Lastly, ans_v is used as optimal solution from s to v.

Lemma 3. *If P_v is the best path from s to v, i.e., $\bar{w}(P_v)$ is minimum among all s-v path, then P_v is computed in $G_{e^*(P_v)}$ and $dist_v^{CNT(P_v)} = w(P_v)$.*

Proof. By definition, P_v exists in $G_{e^*(P_v)}$ and $CNT(P_v) \ge 1$, here we call an edge e *crucial* if $w(e) = w(e^*(P_v))$. Noting $\bar{w}(P_v) = w(P_v) + CNT(P_v) * w(e^*(P_v))$, in one hand if we treat $CNT(P_v)$ as a fixed number, then we need to keep $w(P_v)$ as

Algorithm 9 Maxedge Shortest Path Algorithm

```
1: Initialize $ans_v = \infty$ for for all $v \in V$
2: sort all edges according to $w(e)$ in ascending order, say $e_1, e_2, ..., e_m$ after sorting
3: for $i = 1$ to $m$ do
4:     Initialize $dist_v^c = \infty, \Pi_v^c = nil$ for all $v \in V$ and all $0 \le c \le n$
5:     $dist_s^0 = 0$
6:     for $j = 1$ to $n - 1$ do
7:         for $k = 0$ to $j$ do
8:             for every node $v \in$ V do
9:                 if $dist_v^k = \infty$ then
10:                    continue
11:                end if
12:                for every neighbor $u$ of $v$ do
13:                    if $w(e_{u,v}) > w(e_i)$ then
14:                        continue
15:                    else if $w(e_{u,v}) == w(e^*)$ then
16:                        if $dist_v^k + w(e_{u,v}) < dist_u^{k+1}$ then
17:                            $dist_u^{k+1} = dist_v^k + w(e_{u,v})$
18:                            $\Pi_u^{k+1} = v$
19:                        end if
20:                    else
21:                        if $dist_v^k + w(e_{u,v}) < dist_u^k$ then
22:                            $dist_u^k = dist_v^k + w(e_{u,v})$
23:                            $\Pi_u^k = v$
24:                        end if
25:                    end if
```

Algorithm 10 Maxedge Shortest Path Algorithm

```
26:                 end for
27:             end for
28:         end for
29:     end for
30:     for $i = 1$ to $n - 1$ do
31:         $ans_v = \min\{ans_v,\ dist_v^i + i * w(e_i)\}$
32:     end for
33: end for
```

small as possible. Inspired by idea of bellman-ford algorithm, we can achieve it by enumerating $|P_v|$, i.e., number of edges on P_v. On the other hand, we need to keep tracking number of crucial edges as well. Hence, $dist_v^c$ is adopted to maintain such information, superscript c reflects exact number of crucial edges. From line 12 to line 25, $dist_v^c$ is updated either when it comes from a neighbor who has already witnessed c crucial edges or it comes from a neighbor with $c - 1$ crucial edges and the edge between is crucial. In either case, node v gets a path, say P', with exact c crucial edges on it and $w(P)$ is minimum. At last, P_v can be selected by enumerating number of crucial edges and that is what line 30 to 32 does. □

Lemma 4. *Maxedge Shortest Path Algorithm(MSPA) runs in $O(n^2m)$ time for each G_e.*

Proof. We can apply similar analysis of bellman-ford algorithm. However, we need to update $dist_v^c$ array, it takes extra $O(n)$ time for every node v in every iteration when enumerating $|P_v|$. Hence, total running time is $O(n^2m)$. □

Theorem 2. *MSPA computes optimal path for every $v \in V$ in $O(n^2m^2)$ time.*

Proof. By *Lemma* 3, if P_v is obtained when computing $G_{e^*P_v}$. Then, by considering all possible G_{e^*}, we could get P_v in one of these layering. It takes $O(m)$ to generate all G_{e^*}, by *Lemma* 4, MSPA runs in $v \in V$ in $O(n^2m^2)$ time. $\square$

5.4.3 Computational Complexity of Disjoint Path Problem

In this section, we study edge disjoint path in optical wireless network. By reduction from well known **Min-Max 2-Path Problem**, i.e., min-max 2 edge disjoint path problem under normal length measurement, we show it is also NP-complete if we try to minimize the longer path when w^+ length is applied. Then we give an ILP formulation to solve this problem optimally. At last, we provide two approximation algorithm, one with approximation ratio 4, running time $O((m+n)logn)$, the other one with approximation ratio 2 while running time is $O(m(m+n)logn)$.

Min-Max 2 Disjoint Path Problem (MinMax2PP)
Instance: An undirected graph $G = (V, E)$ with a positive weight $w_(e)$ associated with each edge $e \in E$, a source node $s \in V$, a destination node $t \in V$, and a positive number X.
Question: Does there exist a pair of edge disjoint paths P_1 and P_2 from s to d in G such that $w(P_1) \leq w(P_2) \leq X$?

The **MinMax2PP** problem is shown to be NP-complete in [35]. With a small modification, we show NP-completeness still holds if w^+ length measurement is adopted.

Min-Max 2 Disjoint Path Problem in Optical Wireless Networks (MinMax2OWFN)

Instance: An undirected graph $G = (V, E)$ with a positive weight $w_(e)$ associated with each edge $e \in E$, a source node $s \in V$, a destination node $t \in V$, and a positive number X.

Question: Does there exist a pair of edge disjoint paths P_1' and P_2' from s to t in G such that $w^+(P_1') \leq w^+(P_2') \leq X'$?

Theorem 3. The **MinMax2OWFN** is NP-complete

Proof. Evidently, **MinMax2OWFN** is in NP class, given two edge joint path P_1' and P_2', we can check if $w^+(P_1') \leq w^+(P_2') \leq X'$ in polynomial time.

We then transfer from **MinMax2PP** to **MinMax2OWFN**. Let graph $G = (V, E)$ with source node s , destination t and an integer X be an instance of MinMax2PP, we construct an instance G' of MinMax2OWFN in following way.

1. Create an identical graph G' with same nodes and edges in G.
2. Add one node s_0 to G'.
3. Create two parallel edges e_{01}, e_{02} between s_0 and s, $w(e_{01}) = w(e_{02}) = \max_{e \in G(E)} w(e)$
4. Choose s_0 to the source node in G' and t to be the destination.
5. Set $X' = X + 2w(e_{01})$

Easy to see, the construction takes polynomial time.

Now we need to show a instance of **MinMax2OWFN** have two edge disjoint paths from s_0 to t with length at most X' if and only if the corresponding instance have two edge disjoint paths from s to t with length at most X.

Suppose there are two edge disjoint paths P_1' and P_2' from s_0 to t in G', such that $w^+(P_1') \leq w^+(P_2') \leq X'$. By the way we construct G', P_1' and P_2' must go through e_{01} and e_{02}. W.l.o.g. we say $e_{01} \in P_1'$ and $e_{02} \in P_2'$. Since $w(e_{01}) = w(e_{02}) =$

$\max_{e\in E(G')}\{w(e)\}$, therefore e_{01} and e_{02} are the crucial edge on P_1' and P_2' respectively. Hence, $P_1' - e_{01}$ and $P_2' - e_{02}$ are two edge disjoint path in G, with length no greater than $X' - 2w(e_{01}) = X$.

Conversely, now suppose P_1 and P_2 are two edge joint path in G satisfying $w(P_1) \leq w(P_2) \leq X$. Follow the same argument above, $P_1 + e_{01}$ and $P_2 + e_{02}$ are two desired paths, with length not exceeding $X + 2w(e_{01}) = X'$. □

5.4.4 Optimal Solution for the Disjoint Path Problem

Here, we give an ILP formulation for **MinMax2OWFN**.

ILP for MinMax2OWFN

$$\min \quad MP$$

s.t.

$$\sum_{(i,j)\in E} f_{i,j,1} - \sum_{(j,i)\in E} f_{j,i,1} = \begin{cases} 1 & i = s \\ -1 & i = t \\ 0 & otherwise \end{cases}$$

$$\sum_{(i,j)\in E} f_{i,j,2} - \sum_{(j,i)\in E} f_{j,i,2} = \begin{cases} 1 & i = s \\ -1 & i = t \\ 0 & otherwise \end{cases}$$

$$f_{i,j,1} + f_{i,j,2} \leq 1 \qquad \forall (i,j) \in E$$

$$w_1 \geq f_{i,j,1} * w(i,j) \qquad \forall (i,j) \in E$$

$$w_2 \geq f_{i,j,2} * w(i,j) \qquad \forall (i,j) \in E$$

$$MP \geq w_1 + \sum_{(i,j)\in E} f_{i,j,1} * w(i,j)$$

$$MP \geq w_2 + \sum_{(i,j)\in E} f_{i,j,2} * w(i,j)$$

$$f_{i,j,1} = \{0,1\}, \quad f_{i,j,2} = \{0,1\} \qquad \forall (i,j) \in E$$

The following is a brief description of this ILP formulation. The first two equation represent flow constraint as normal shortest path problem does. $f_{i,j,1} = 1$ indicates path P_1 goes through edge (i,j), and 0 otherwise. So it is with $f_{i,j,2}$ and path P_2. Constraint 3 ensures two edges are disjoint, since $f_{i,j,1}$ and $f_{i,j,2}$ cannot both be 1 at the same time. w_1, w_2 act as the weights of the crucial edges on P_1 and P_2 respectively. Finally, we define MP to be the maximum of $w^+(P_1)$ and $w^+(P_2)$ and therefore try to minimize it.

5.4.5 *Approximation Algorithm for the Disjoint Path Problem, with approximation factor 4*

Next we propose a 4-approximation algorithm which runs in $O((n+m)logn)$ time.

Given $G = (V, E)$ with source s and destination t, the idea of approximation algorithm is to find two disjoint P_1 and P_2 such that $w(P_1) + w(P_2)$ is minimized. Such P_1 and P_2 can be found either using min cost max flow algorithm or the algorithm due to Suurballe presented in [54]. And we need to show both $w^+(P_1)$ and $w^+(P_2)$ are at most four times of the optimal solution.

Algorithm 11 MinMax2OWFN Approximation Algorithm 1 (MAA1)

```
1: Run Suuraballe's algorithm on $G$, denote $P_1$, $P_2$ be two resulting path.
2: Compute $w^+(P_1)$ and $w^+(P_2)$.
3: Output $\max\{w^+(P_1), w^+(P_2)\}$.
```

Lemma 5. *For any path P, $w^+(P) \leq 2w(P)$.*

Proof. By definition, $w^+(P) = w(P) + w(e^*(P))$. Since $w(e^*(P)) \leq w(P)$, then $w^+(P) \leq 2 * w(P)$. □

Lemma 6. *If P_1 and P_2 are two edge joint path from s to t such that $w(P_1)+w(P_2)$ is minimum, then $w^+(P_1)$ and $w^+(P_2)$ are at most four times of the optimal solution.*

Proof. Say *opt* is the optimal value of a $MinMax2OWFN$ instance and Q_1,Q_2 are two $s-t$ edge disjoint path in one optimal solution. W.l.o.g, we may suppose $w^+(P_1) \geq w^+(P_2)$ and $w^+(Q_1) \geq w^+(Q_2)$. Let $w(P_1) + w(P_2) = p$ and $w(Q_1) + w(Q_2) = q$, by assumption, $p \leq q$. Also, we have $w^+(P_1) = w(P_1) + e^*(P_1) \leq 2p$, $opt = w^+(Q_1) = w(Q_1) + e^*(Q_1) > \frac{q}{2}$. Hence, $\frac{w^+(P_2)}{opt} \leq \frac{w^+(P_1)}{opt} < \frac{2p}{q/2} \leq 4$ □

Theorem 4. *MMA1 is a 4-approximation algorithm running in $O((n+m)logn)$ time and 4 is a tight bound.*

Proof. By *Lemma* 5 and 6, MMA1 has approximation ratio at most 4.Then we show MMA1 has approximation at least 4 for certain cases. Consider the following graph.

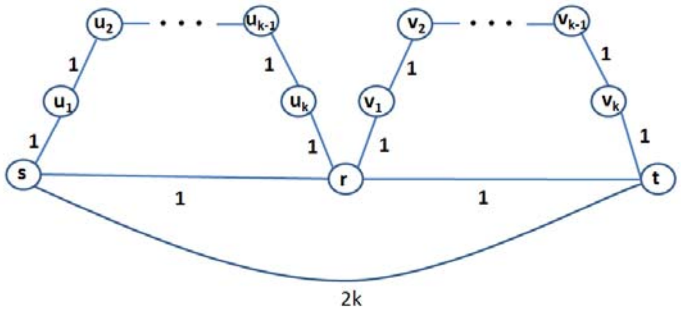


Figure 5.1: Tightness for ratio 4

Easy to check, $P_1 = \{s \to t\}$, $P_2 = \{s \to r \to t\}$ are two edge disjoint path with minimum length $2k + 2$, $w^+(P_1) = 4k > w^+(P_2) = 3$. However, let $Q_1 = \{s \to u_1 \to u_2 \to ... \to u_{k-1} \to u_k \to r \to t\}$, $Q_2 = \{s \to r \to v_1 \to v_2 \to ... \to v_{k-1} \to v_k \to t\}$, then $w(Q_1) + w(Q_2) = 2k + 4$ while $w^+(Q_1) = w^+(Q_2) = k + 3$. $\frac{w^+(P_1)}{w^+Q_1} = \frac{4k}{k+3} \approx 4$ when k is sufficiently large. Hence, 4 is a tight bound for MMA1.

We need $O((n + m)logn)$ time running Suuraballe's algorithm and $O(n)$ time computing $w^+(P_1)$ and $w^+(P_2)$. Therefore total running time is $O((n + m)logn)$.

□

5.4.6 *Approximation Algorithm for the Disjoint Path Problem, with approximation factor 2*

If we do not apply layering technique in MMA1 and instead we consider all G_e of G, we can have a better approximation ratio.

Algorithm 12 MinMax2OWFN Approximation Algorithm 2(MAA2)

```
1: set $ans = \infty$
2: for every $G_e$ of $G$ do
3:     Run Suuraballe's algorithm on $G_e$, denote $P_1$, $P_2$ be two resulting path.
4:     Compute $w^+(P_1)$ and $w^+(P_2)$.
5:     $ans = \min\{ans,\ \max\{w^+(P_1), w^+(P_2)\}\}$.
6: end for
7: Output $ans$.
```

Say Q_1, Q_2 are two disjoint paths in one optimal solution. Let $e' = \max\{e^*(Q_1), e^*(Q_2)\}$ and P_1, P_2 be the resulting paths when computing layer $G_{e'}$; w.l.o.g, we may assume $w(P_1) > w(P_2)$. Also, let $ans_{e'} = \max\{w^+(P_1),\ w^+(P_2)\}$.

Lemma 7. $ans_{e'} < 2\max\{w^+(Q_1),\ w^+(Q_2)\}$.

Proof. Noting that $w(e^*(P_1)) \leq w(e')$ and $w(e^*(P_2)) \leq w(e')$ since they both belong to $G_{e'}$. Then $ans_{e'} \leq w(P_1) + w(e')$. It suffices to show $\frac{w(P_1)+w(e')}{\max\{w^+(Q_1),\ w^+(Q_2)\}} < 2$. We prove it by contradiction. Suppose $\frac{w(P_1)+w(e')}{\max\{w^+(Q_1),\ w^+(Q_2)\}} \geq 2$, then

$$w(P_1) + w(e') \geq w^+(Q_1) + w^+(Q_2)$$

Which follows,

$$w(P_1) + w(e') \geq w(Q_1) + w(e^*(Q_1)) + w(Q_2) + w(e^*(Q_2))$$

By definition, e' is one of $e^*(Q_1)$, $e^*(Q_2)$. Hence,

$$w(P_1) > w(Q_1) + w(Q_2)$$

It is impossible since $w(P_1) + w(P_2)$ is minimum in layer $G_{e'}$. □

Theorem 5. *MMA2 is a 2-approximation algorithm running in $O(m(n+m)logn)$ time and 2 is a tight bound.*

Proof. By *Lemma*7, in one of the layer, we guarantee to have a 2-approximation solution. Since we take minimum outcome among all layers, the final result is no worse than twice of the optimal solution. Now we need to show there exists certain case, such that MMA2 is no good than twice of the optimal solution. Consider the following graph

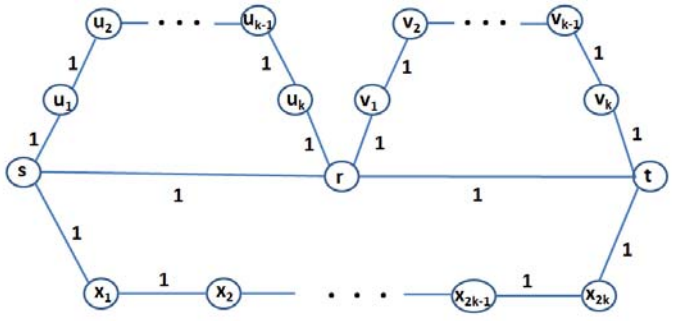


Figure 5.2: Tightness for ratio 2

There is only one layer, and $P_1 = \{s \to x_1 \to x_2 \to ... \to x_{2k-1} \to x_2k \to t\}$, $P_2 = \{s \to r \to t\}$ are two edge disjoint path with minimum length $2k+3$, $w^+(P_1) = 2k+2 > w^+(P_2) = 3$. However, set $Q_1 = \{s \to u_1 \to u_2 \to ... \to u_{k-1} \to u_k \to r \to t\}$, $Q_2 = \{s \to r \to v_1 \to v_2 \to ... \to v_{k-1} \to v_k \to t\}$, then $w(Q_1) + w(Q_2) = 2k+4$ while $w^+(Q_1) = w^+(Q_2) = k+3$. $\frac{w^+(P_1)}{w^+Q_1} = \frac{2k+2}{k+3} \approx 2$ when k is sufficiently large. Hence, 2 is a tight bound for MMA2.

Finally, it is easy to see running time is $O(m(n+m)logn)$. □

5.4.7 Experimental Results for the Disjoint Path Problem

In this section, we present the results of simulations for comparing the performance of our approximation algorithms with the optimal solution. The simulation experiments have been carried out on the ARPANET topology (as shown in Fig 5.3 with nodes and links shown in black) which has twenty nodes and thirty two links. The weights of the links have been randomly generated and lie in the range of two and twelve (as shown in red in Fig 5.3) and we consider the graph to be undirected. The results of the comparison is presented in Table 5.1. We have compared the lengths of the longer of the two edge disjoint paths computed by the optimal and the approximate solutions for seventeen different source-destination pairs. It may be noted that for almost 65% of the cases, the approximate algorithm obtains the optimal solution. In the remaining cases, the approximate solution lies with a factor of 1.2 of the optimal solution. Thus, even though the approximation ratio in the worst case is proven to be 4, in practical cases, it is within 1.2. From these experimental results, we can conclude that the approximation algorithm produces optimal or near optimal solutions in majority of the cases. It may be noted that the two approximation algorithms perform in a similar fashion in the ARPANET graph, however, as proven theoretically, the two approximation algorithms differ in their worst case approximation ratio.

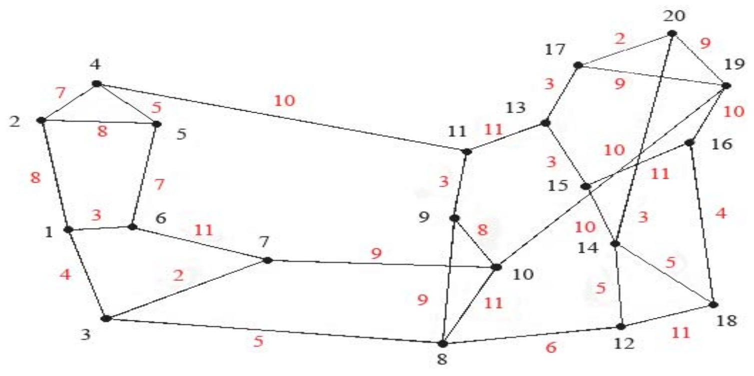


Figure 5.3: The ARPANET graph with 20 nodes and 32 links

S node	*D node*	*Opt Sol*	*Approx Sol* 1	*Approx Ratio* 1	*Approx Sol* 2	*Approx Ratio* 2
14	2	47	55	1.17	55	1.17
18	8	46	46	1	46	1
1	6	28	28	1	28	1
18	4	50	58	1.16	57	1.14
20	3	40	40	1	40	1
10	3	27	27	1	27	1
1	11	35	35	1	35	1
14	6	50	52	1.04	52	1.04
20	7	38	38	1	38	1
10	5	36	38	1.05	38	1.05
18	12	22	22	1	22	1
1	20	46	52	1.13	52	1.13
20	13	26	26	1	26	1
14	19	29	29	1	29	1
10	17	36	36	1	36	1
20	16	29	29	1	29	1
5	11	40	48	1.2	48	1.2

Table 5.1: Comparison of the Approximate solutions with the Optimal solution for the ARPANET graph

In addition, we also conducted experiment on German topology as shown in 5.4. 50 cites(nodes) are included in the topology with given x, y coordinates in 5.2 . Edges weights are computed based on their geo-location and we also treated it as an undirected graph. We choose 20 different source and destination pairs. For each one, we computed optimal solution as well as approximation results. The results of the comparison is presented in Table 5.3. For most of cases, our approximation results achieve the optimal solution.

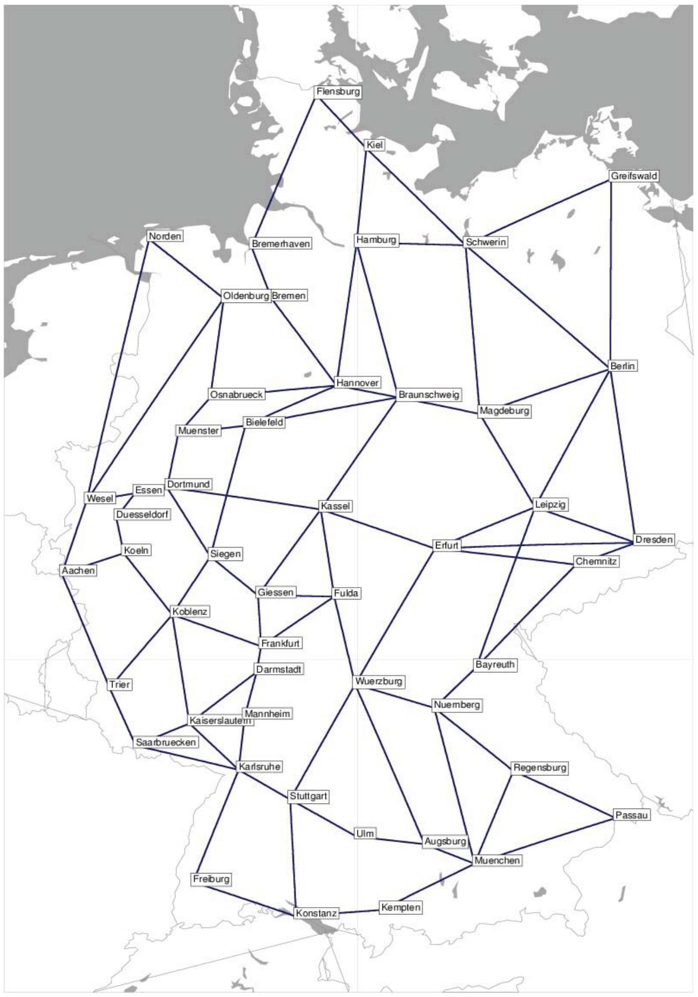


Figure 5.4: The German topology with 50 nodes and 88 links

Name	*ID*	*x*	*y*	*Name*	*ID*	*x*	*y*
Aachen	0	0.0	1279.66	*Kassel*	25	669.42	1107.84
Augsburg	1	937.58	2002.36	*Kempten*	26	825.69	2178.31
Bayreuth	2	1070.69	1530.58	*Kiel*	27	787.1	143.05
Berlin	3	1417.94	732.46	*Koblenz*	28	285.52	1389.03
Bielefeld	4	474.58	883.82	*Koeln*	29	160.12	1224.66
Braunschweig	5	870.06	808.35	*Konstanz*	30	605.76	2195.51
Bremen	6	542.1	544.14	*Leipzig*	31	1223.1	1101.67
Bremerhaven	7	490.01	405.25	*Magdeburg*	32	1080.34	852.42
Chemnitz	8	1329.2	1255.24	*Mannheim*	33	472.65	1661.85
Darmstadt	9	503.51	1542.56	*Muenchen*	34	1066.83	2054.49
Dortmund	10	272.01	1049.07	*Muenster*	35	300.95	905.76
Dresden	11	1483.53	1197.08	*Norden*	36	225.71	385.76
Duesseldorf	12	140.83	1129.43	*Nuremberg*	37	962.66	1638.07
Erfurt	13	964.59	1212.41	*Oldenburg*	38	418.63	544.14
Essen	14	189.06	1064.56	*Osnabruck*	39	383.91	808.35
Flensburg	15	657.85	0.0	*Passau*	40	1431.45	1932.55
Frankfurt	16	515.09	1473.53	*Regensburg*	41	1167.15	1806.65
Freiburg	17	339.53	2103.57	*Saarbrucken*	42	190.99	1738.86
Fulda	18	704.15	1340.53	*Schwerin*	43	1043.68	402.0
Giessen	19	507.37	1337.49	*Siegen*	44	383.91	1233.84
Greifswald	20	1419.87	225.53	*Stuttgart*	45	590.33	1882.9
Hamburg	21	762.02	395.5	*Trier*	46	123.47	1584.42
Hannover	22	709.94	776.78	*Ulm*	47	762.02	1982.03
Kaiserslautern	23	329.89	1679.65	*Wesel*	48	63.66	1086.22
Karlsruhe	24	457.21	1803.71	*Wurzburg*	49	758.17	1575.46

Table 5.2: City info in German topology

S node	*D node*	*Opt Sol*	*Approx Sol* 1	*Approx Ratio* 1	*Approx Sol* 2	*Approx Ratio* 2
9	14	1190	1190	1.0	1190	1
24	44	946	946	1.0	946	1
17	34	1311	1311	1.0	1311	1
19	43	1829	1829	1.0	1829	1
8	9	1966	1966	1.0	1966	1
5	31	1380	1380	1.0	1380	1
46	49	1297	1297	1.0	1297	1
31	32	1274	1274	1.0	1274	1
0	13	1925	1925	1.0	1925	1
37	45	1371	1371	1.0	1371	1
20	48	2783	2783	1.0	2783	1
12	32	1865	1865	1.0	1865	1
3	7	1992	1992	1.0	1992	1
5	24	1820	1842	1.01	1820	1
18	32	1533	1533	1.0	1533	1
2	13	1199	1199	1.0	1199	1
3	45	2336	2336	1.0	2336	1
2	27	2335	2335	1.0	2335	1
17	22	2358	2368	1.004	2358	1
1	36	2937	2937	1.0	2937	1

Table 5.3: Comparison of the Approximate solutions with the Optimal solution for the German topology

Chapter 6

READER SCHEDULING FOR TAG ACCESS IN RFID SYSTEMS

6.1 Introduction

Radio Frequency Identification (RFID) systems, comprising of *readers* and *tags*, are used extensively for identification of objects with unique identifiers. In order to support complex needs of RFID dependent business sectors, such as Supply Chain Management and Transportation, a RFID system is expected to allow readers fast and accurate access to tags available in the environment. However, simultaneous transmissions by multiple readers and tags in close proximity may cause signal interference and hinder accurate reading. Such interference can be divided into three classes : *tag-to-tag*, *reader-to-tag*, and *reader-to-reader*. To overcome these hindrances and achieve interference-free operation, development of conflict resolution techniques are essential. Accordingly, several conflict resolution techniques have been developed. However, most of these techniques are developed for resolving tag-to-tag collision, instead of reader-to-reader collision.

In this chapter we study the optimal schedule construction problem to avoid reader-to-reader collision. We formalize the optimal schedule construction problem for RFID readers as computation of *interval chromatic number* [52] of a RFID-conflict graph (a generalized version of Unit Disk Graphs), and prove that the problem is NP-complete. We provide a centralized and a distributed approximation algorithm for the problem with a guaranteed performance bound.

The rest of the chapter is organized as follows: In Section 6.2 we outline the related work in this domain. In Section 6.3 we formally define the problem and in Section 6.4 we show that the scheduling problem is NP-complete and provide a heuristic algorithms for the problem. Finally, in Section 6.6 we present the results of our experiments.

6.2 Related Work

Reader-reader anti-collision protocols from the literature are of several kinds. Some, like HiQ-learning [24] use a hierarchical architecture to provide an online learning of collision patterns of readers and assign frequencies to the readers over time. Others, such as [26, 28] apply a carrier sense multiple access (CSMA) based algorithms to detect collisions. Centralized solutions [49] use an iterative procedure. A reader is allocated a color in an order determined by the number of neighboring readers already colored, choosing at each step, the color with the smallest index. In Distributed Color Selection (DCS) [58], each reader randomly selects a time slot in a frame for transmission. The Variable-Maximum DCS (VDCS, or colorwave) [57] allows adjustment of the maximum number of colors. In [14], mobile readers communicate with a centralized server which grants service to readers for tag identification on a first-come-first-served basis. [22] proposed an Adaptive Color based Reader Anti-collision Scheduling algorithm for 13.56 MHz RFID technology where every reader is assigned a set of colors that allows it to read tags during a specific time slot within a time frame. [55] also studied the slotted access model to improve the read throughput of a multi-reader RFID system by extending a centralized algorithm to a distributed one that operates without location information of other readers.

Our model differs from the slotted access model as we assume a reader can read only one tag per time unit. Also, a reader fails to read any nearby tags if there is a reader collision, i.e. a tag is present in the sensing range of two active readers.

6.3 Problem Formulation

The inputs for the RFID scheduling problem are the locations of the readers and the tags in the deployment area. Suppose that there are n readers located in points $\{p_1, \ldots, p_n\}$ and m tags located in points $\{q_1, \ldots, q_m\}$. If the deployment area is a two dimensional space, then each point p_i (or q_i) is specified by its x, y co-ordinates (x_i, y_i). We formulate the RFID scheduling problem as computation of the *Interval Chromatic Number* of a *RFID Graph.*

Definition: *Interval Chromatic Number (ICN):* An interval coloring of a weighted graph maps each node v to an interval of size $w(v)$ such that intervals of adjacent nodes do not intersect. The size of a coloring is the size of the union of these intervals. The minimum possible size of an interval coloring of a given weighted graph is its interval chromatic number [52].

We draw a circle of radius r, with each point $p_i, 1 \leq i \leq n$ as the center, where r is the sensing range of the readers. Corresponding to every point p_i in the problem instance, we create a node v_i in the graph $G = (V, E)$, and two nodes share an edge if the intersection area of the sensing circles of the corresponding points p_i and p_j covers at least one tag. Since G is constructed from an instance of the RFID problem, we will refer to it as a *RFID Graph (RFIDG).* It can be seen that RFIDGs are a generalization of the UDG (when $r = 1$).

As a reader needs to be turned on for $t \times N_{tag}$ *consecutive* time units to ensure that all tags in its sensing range is read, for each node $v_i \in V$ we assign a *weight* $w_i = t \times N_i$, where N_i is the number of tags available in the reader's sensing range. This way of assigning weights may be considered somewhat inefficient as the tags that belong to the intersection area of the sensing range of multiple readers will be read by multiple readers. However, such duplication can be avoided if the readers have sophisticated electronics to determine which tag is being read by which reader, this is currently unavailable in today's commodity RFID readers. Accordingly, we use this weight assignment rule to ensure that no tag is left unread.

We can now view the optimal schedule construction problem for the RFID problem as the Interval Coloring problem of the corresponding RFID graph. We associate colors to readers as communication tokens. A single color stands for a unit of time. To ensure that every reader i successfully accesses all tags in its sensing range, i has to be allocated N_i colors.

Incompatibility rule: Two readers are *incompatible* if there is a tag in the intersection area of their sensing range.

Optimal Schedule Construction Problem (OSCP):

GIVEN: Two sets of points $P = \{p_1, \dots, p_n\}$ (locations of readers) and $Q = \{q_1, \dots, q_m\}$ (locations of tags), the sensing range r, from which the RFID Graph $G = (V, E)$ can be constructed, where V is the set of nodes corresponding to the set of readers, and for every pair of nodes $\{u, v\} \in V$ there exists an edge $(u, v) \in E$ if the readers represented by the nodes (u, v) are *incompatible*.

QUESTION: Is it possible to assign an interval $I(v_i)$ of size $|I(v_i)| = w_i$ to each node v_i in V, such that the total span of all the intervals does not exceed some predefined value B ($\cup_i |I_i| < B$), and $\forall i, j I_i \cap I_j = \emptyset$? A schedule is said to be "optimal" if the total span all the intervals is the smallest.

From our discussion, Optimal Schedule Construction Problem (OSCP) for a RFID system is equivalent to the ICN computation problem of the corresponding RFID graph.

6.4 Heuristic Algorithms and Analysis

We firstly prove that the OSCP is NP-complete. We then provide a heuristic algorithm and analyze it to establish a performance bound. Finally, we provide a distributed implementation of our algorithm. As the solution to the OSCP is equivalent to the computation of ICN of a RFID graph, our algorithm essentially computes the ICN of a RFIDG. Note that UDG is a special case of RFIDG. We prove that the result produced by this algorithm will be bounded by a factor of $\mathtt{max}(3, 2+k)$ of the optimal solution for UDG, and $\mathtt{max}(3\alpha, 2+k)$ of the optimal solution for RFIDG, where k and α are parameters determined by reader and tag density respectively. For our application we expect $k \leq 5$ and $\alpha \leq 5$, and later in this section we explain why we expect the two parameters to satisfy these two bounds.

Theorem 6. *The OSCP is NP-complete.*

Proof. If we consider a case of the OSCP where weight $w(v_i)$ of every node v_i is 1, and the intersection area of every pair of circles associated with the readers has a tag, OSCP becomes equivalent to the computation of the Chromatic Number of a Unit Disk Graph, a known NP-complete problem [46]. □

Notations:

$N(v_k)$: Set of v_k's neighbors, ie nodes sharing an edge in G with v_k.

$N^l(v_k)$ (resp. $N^r(v_k)$) : Left (resp. right) neighbors of v_k : $v_i \in N(v_k)$ s.t. $x_i < x_k$ (resp ($x_i > x_k$)).

$|I(v_i)| = w_i$: Length of $I(v_i)$, where w_i is the weight of v_i

$L(I(v_i))$: Left end point of $I(v_i)$ on the Interval Line.

$R(I(v_i))$: Right end point of $I(v_i)$ on the Interval Line.

Definition: *Lexicographic Ordering:* The Lexicographic Ordering of a set of points in a plane is the ordering induced by their (x, y) coordinates. The points are ordered by the increasing values of their x coordinates and in case of a tie, are ordered by the increasing values of their y coordinates [46].

Definition: *Least Indexed Coloring (LIC):* LIC scheme assigns an interval $I(v_i)$ to each node v_i, such that $L(I(v_i))$ is as small as possible, without violating any stated constraint.

Our interval coloring uses the LIC scheme on lexicographic ordering of the nodes. Centralized and distributed algorithms for computation of ICN are summed up in Algo. 13 and 14 resp.

Algorithm 13 Centralized ICN Algorithm for RFIDG/UDG

```
1: Arrange the nodes (readers) in Lexicographic Ordering
2: Sequentially apply LIC on the nodes till each node is assigned an interval of size
   equal to its weight with no overlap with intervals of adjacent nodes.
```

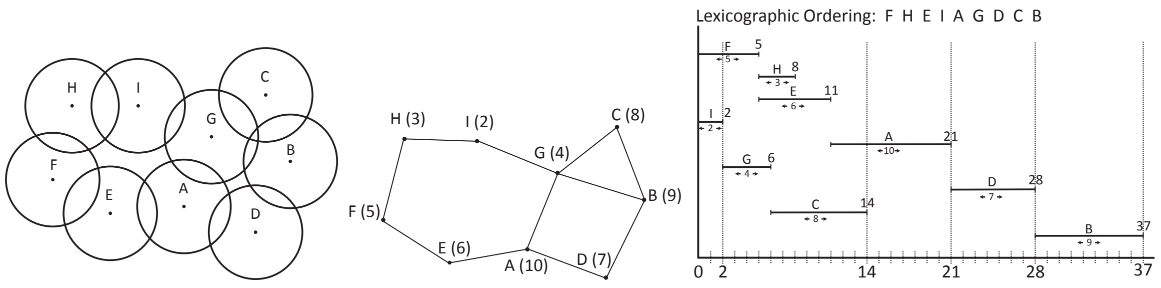
Algorithm 14 Distributed ICN Algorithm for RFIDG/UDG
(executed independently by each node v_i in the graph)

```
1: $v_i$ broadcasts $x_i, y_i$ and $N_i$
2: while $v_i$ is not assigned color do
3:     if Every node in $N^l(v_i)$ has assigned colors then
4:         $v_i$ chooses $I(v_i)$ such that $|I(v_i)| = N_i$, $\forall v_j \in N^l(v_i)$, $I(v_i) \cap I(v_j) = \emptyset$ and
           $L(I(v_i))$ is the smallest.
5:         $v_i$ broadcasts $I(v_i)$
6:     end if
7: end while
```

For the distributed version of the coloring algorithm we assume that the readers are aware of their own position as well as the positions of its neighbors and the location of the tags in its sensing range. As discussed earlier, each reader v_i must be assigned a set of w_i consecutive colors to ensure that every tag is read. To do so, we rely on [46] which proposes a 3-approximated lexicographic order coloring. To illustrate it, let us consider the conflict graph represented on Fig. 6.1b. Nodes represent readers with the number of colors they should have. Each reader v_i collects the position of its neighbors and their colors. As readers are aware of their own position and positions of their neighbors in the deployment area, they can determine the "lexicographic order" (i.e. F, H, E, I, A, G, D, C and B in Fig. 6.1b). Each reader will assign a set of colors to itself, only after all its *left* neighbors have assigned colors to themselves. In Fig. 6.1b, reader G will wait till readers I and A are colored. Reader B waits for D, G and C, but H and E chooses independently as soon as F is colored as they do not share an edge. F has no left neighbors, thus it chooses first and takes the smallest set of 5 colors, i.e. colors 1, 2, 3, 4, 5 (as shown on Fig.6.1c). Readers H and E follow by respectively assigning colors 6-8 and 6-11, and so on. B cannot take colors between 2-14, nor 21-28 as they have already been selected by readers G, C and D. Although colors 14-21 are available in the left neighborhood of reader B, it cannot utilize these colors as it requires 9 consecutive colors. It thus assigns colors 28-37. Finally, G assigns itself colors 2-6 as reader I is already using colors 0-2. It may be noted that in this example, the centralized and distributed algorithms provide the same solution.

Analysis of Algorithm

The input of both algorithms, are the locations of the readers (nodes) and the weights assigned to the nodes. In the following sections, we first analyze the performance of our algorithms for a special case of RFID graphs, known as Unit Disk Graphs (UDG). In UDGs every pair of readers has at least one tag present in the intersection area of



(a) Deployment area (b) Unit Disk Graph (c) Interval assignment

Figure 6.1: (a) Readers (points) in the deployment area and their sensing range, (b) UDG graph constructed from the problem instance in Fig. 6.1a, (c) Interval assignment for the problem instance in Fig. 6.1a

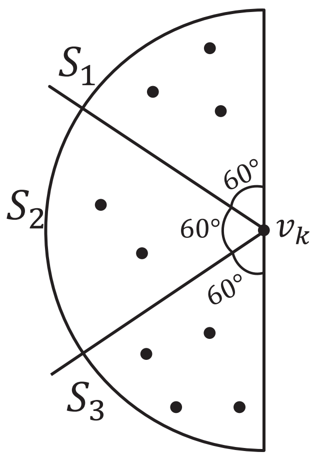
the circles associated with the nodes.

Part I: Analysis of Algorithm for Unit Disk Graphs

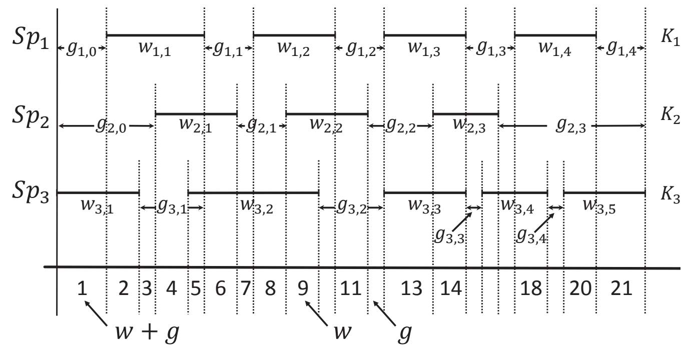
Our interval coloring algorithm uses the LIC scheme on lexicographic ordering of the nodes.If $R(I(v_i)) \geq R(I(v_j)) \forall 1 \leq j \leq n$ then the node v_i is called a *critical node.* Suppose that v_k is a critical node when algorithm $\mathcal{A}$ is applied on an instance of the RFID scheduling problem. In this case, $R_{\mathcal{A}}(I(v_k))$ is the solution to the instance of the interval coloring problem using algorithm $\mathcal{A}$. We will refer to $R_{\mathcal{A}}(I(v_k))$ as *Approximate Interval Chromatic Number* and denote it by $AICN$. As per the interval layout shown in Fig. 6.1c, the critical node is $v_k = B$.

Intervals $I(v_1), I(v_2), \ldots, I(v_{k-1})$ associated with nodes $v_1, v_2, \ldots, v_{k-1}$ are mapped on the Interval Line before the interval $I(v_k)$. For this reason, $L_{\mathcal{A}}(I(v_k))$ may be greater than zero as some of the nodes in the set $\{v_1, v_2, \ldots, v_{k-1}\}$ may be adjacent to v_k in G and the intervals associated with these sets of nodes cannot overlap with the interval $I(v_k)$.

If v_l and v_r are two nodes in $N(v_k)$, such that $L_{\mathcal{A}}(I(v_l)) \leq L_{\mathcal{A}}(I(v_j))$, and



(a) Sections S_1, S_2, S_3



(b) Segment Span

Figure 6.2: (a) Left side neighbors of the critical node v_k, (b) Span of the three segments associated with the critical node v_k

$R_{\mathcal{A}}(I(v_r)) \geq R_{\mathcal{A}}(I(v_j))$, , the interval between $L_{\mathcal{A}}(I(v_l))$ and $R_{\mathcal{A}}(I(v_r))$ will be referred to as the *span* of $N(v_k)$ with algorithm $\mathcal{A}$, and will be denoted by $Sp_{\mathcal{A}}(N(v_k))$. The *length* of the span is the difference between $L_{\mathcal{A}}(I(v_l))$ and $R_{\mathcal{A}}(I(v_r))$ and is denoted by $|Sp_{\mathcal{A}}(N(v_k))|$. In Fig. 6.1c $Sp_{\mathcal{A}}(N(v_k))$ is from 2 to 28, of length of the span $|Sp_{\mathcal{A}}(N(v_k))| = 26$.

It may be recalled that as our graph is a UDG, the nodes $v_1, \ldots, v_n$ correspond to points $p_i, \ldots, p_n$ on a two dimensional plane. Suppose that we draw a semi-circle of unit radius around the point p_k corresponding to the node v_k (as shown in Fig. 6.2a), and divide the semi-circle into three 60 degree segments, $S_1, \ldots, S_3$, as shown in Fig. 6.2a. We denote by $N^i(v_k)$ the subset of $N(v_k)$, comprised of nodes corresponding to points in $S_i, \forall 1 \leq i \leq 3$. Due to the construction rule of a UDG, the nodes corresponding to the points that belong to segment $S_i, 1 \leq i \leq 3$ form a clique with the node v_k in $G = (V, E)$. As such, $AICN \geq |Sp(N^i(v_k), \mathcal{A})|, \forall i, 1 \leq i \leq 3$. Although $Sp(N^i(v_k), \mathcal{A})$ and $Sp(N^j(v_k), \mathcal{A}), j \neq i$, need not be non-overlapping, in the worst case scenario $|Sp_{\mathcal{A}}(N(v_k))|$ may be as large as $\sum_{i=1}^{3} |Sp_{\mathcal{A}}(N^i(v_k))|$. The maximum value of $|Sp_{\mathcal{A}}(N(v_k))|$, denoted by $Max\_Sp_{\mathcal{A}}(N(v_k))$, can be $\sum_{i=1}^{3} |Sp_{\mathcal{A}}(N^i(v_k))|$. As points (nodes) that belong to segment $S_i, 1 \leq i \leq 3$ form a clique with the node v_k in G, the minimum value of $|Sp_{\mathcal{A}}(N(v_k))|$, denoted by $Min\_Sp_{\mathcal{A}}(N(v_k))$, has to be at least $\max_{1 \leq i \leq 3} \left[\sum_{u \in N^i(v_k)} |I(u)|\right]$.

It may be recalled that $\{v_1, v_2, \ldots, v_{k-1}\}$ were assigned intervals on the Interval Line before the critical node v_k. We will denote the set $\{v_1, v_2, \ldots, v_{k-1}\}$ by V_{k-1}. Thus, the set of nodes in V_{k-1} that are not adjacent to the node v_k is given by (called *non-neighbors of* v_k, $NN(v_k)$), $NN(v_k) = V_{k-1} \setminus N(v_k)$. In the example of Fig. 6.1a, the set $NN(v_k) = \{A, E, F, H, I\}$, where $v_k = B$. When the nodes in the set $NN(v_k)$ are assigned intervals on the Interval Line by the algorithm $\mathcal{A}$, some of these intervals may have overlap with the interval span of the neighbors of v_k, i.e., $Sp(N(v_k), \mathcal{A})$. However, there may be some nodes in $NN(v_k)$ whose assigned intervals may not have any overlap with the $Sp(N(v_k), \mathcal{A})$. We will refer to this subset of $NN(v_k)$ as *non-overlapping non-neighbor* of v_k and denote it by $NO\_NN(v_k)$. In the example of Fig. 6.1a, the set $NO\_NN(v_k) = \{I\}$. The span of non-overlapping non-neighbors of v_k, $Sp(NO\_NN(v_k), \mathcal{A})$ can be at most $|I(v_k)|$, as otherwise $I(v_k)$ can be assigned space in the interval line covered by the $Sp(NO\_NN(v_k), \mathcal{A})$, as such an assignment will not violate the *non-overlapping requirement* for adjacent nodes. The ICN will have three non-overlapping intervals on the Interval Line corresponding to $Sp(NO\_NN(v_k), \mathcal{A})$, $Sp(N(v_k), \mathcal{A})$ and $I(v_k)$. As such, we can conclude that $AICN \leq Max\_Sp(NO\_NN(v_k), \mathcal{A}) + Max\_Sp(N(v_k), \mathcal{A}) + |I(v_k)|$. As the maximum value of $Sp(N(v_k), \mathcal{A})$ is $\sum_{i=1}^{3} |Sp(N^i(v_k), \mathcal{A})|$ and the maximum value of $Sp(NO\_NN(v_k), \mathcal{A})$ is $|I(v_k)|$, it follows that:

$$AICN \leq \sum_{i=1}^{3} |Sp_{\mathcal{A}}(N^i(v_k))| + 2|I(v_k)| \tag{6.1}$$

As nodes in $N^i(v_k)$ form a clique with v_k, any optimal solution to the Interval Chromatic Number of UDG ($OICN$) must be at least as large as:

$$OICN \geq \max_{1 \leq i \leq 3} \left[\sum_{u \in N^i(v_k)} |I(u)| \right] + |I(v_k)| \tag{6.2}$$

Next, we examine the relationship between $\sum_{i=1}^{3} |Sp(N^i(v_k), \mathcal{A})|$ and $\max_{1 \leq i \leq 3} \left[\sum_{u \in N^i(v_k)} |I(u)| \right]$.

For any instance of the problem, the diagram of $Sp(N(v_k))$ will have the form shown in Fig. 6.2b. Suppose the number of nodes in the segments S_1, S_2, S_3 are K_1, K_2, K_3. In that case Sp_1, Sp_2, Sp_3 will have K_1, K_2, K_3 intervals with possibly *gaps* between them as shown in Fig. 6.2b. Suppose that the weights of nodes in $S_i, 1 \leq i \leq 3$ are $w_{i,1}, w_{i,2}, \ldots, w_{i,K_i}$. If we draw vertical lines through the left and right end points of every interval, the lines will intersect the Interval Line at most at $2\sum_{j=1}^{3} K_j$ points and divide the Interval Line into at most $2\sum_{j=1}^{3} K_i + 1$ *sub-intervals*. The sub-intervals are divided into three disjoint groups – *w-type*, *g-type* and *(w+g)-type*, depending on whether they include only w, g, or w and g type of parts from the spans Sp_i. Examples of *w-type*, *g-type* and *(w+g)-type* are shown in Fig. 6.2b. The total space occupied on the Interval Line by w and *(w+g)* type sub-intervals is at most: $\sum_{i=1}^{3} \sum_{j=1}^{K_i} w_{i,j} = \sum_{i=1}^{3} \left[\sum_{u \in N^i(v_k)} |I(u)| \right]$. The size of g-type sub-interval can be at most $|I(v_k)|$ as otherwise, the critical node interval $I(v_k)$ could have been inserted in the gap. The number of g-type sub-intervals can be at most half of the total number of sub-intervals on the Interval Line. As noted earlier, there could be at most $2K + 1$ sub-intervals and as such, the number of g-type sub-intervals can be at most $K' = \lceil (2K + 1)/2 \rceil = K + 1$. Consequently:

$$\sum_{i=1}^{3} |Sp_{\mathcal{A}}(N^i(v_k))| \leq \sum_{i=1}^{3} \left[\sum_{u \in N^i(v_k)} |I(u)| \right] + (K + 1)|I(v_k)|$$

Thus, from Equation (6.1), we have:

$$AICN \leq \sum_{i=1}^{3} \left[\sum_{u \in N^i(v_k)} |I(u)| \right] + (K+1)|I(v_k)| + 2|I(v_k)|, \text{ or}$$

$$AICN \leq \sum_{i=1}^{3} \left[\sum_{u \in N^i(v_k)} |I(u)| \right] + (K+3)|I(v_k)| \tag{6.3}$$

Using Equations (6.2) and (6.3), we have:

$$\frac{AICN}{OICN} \leq \frac{\sum_{i=1}^{3} \left[\sum_{u \in N^i(v_k)} |I(u)| \right] + (K+3)|I(v_k)|}{\max_{1 \leq i \leq 3} \left[\sum_{u \in N^i(v_k)} |I(u)| \right] + |I(v_k)|}, \text{ or}$$

$$\frac{AICN}{OICN} \leq \max(3\alpha, (K+3)) \tag{6.4}$$

In the RFID application domain, it is unlikely that reader density will be high. If there are no more than five readers within a semi-circle of radius equal to the sensing range of a reader, then $K \leq 5$. In this case $\frac{AICN}{OICN} \leq 8$.

Part II: Analysis of Algorithm for RFID Graphs

In Part I, we established that $\frac{AICN}{OICN} \leq \max(3, (K+3))$ for Unit Disk graphs. In Part II, we extend that result to RFID graphs, a generalized version of the UDG. In a UDG, if the circles corresponding to points p_i and p_j intersect, then corresponding nodes v_i and v_j share an edge in G. In RFIDG, if the circles corresponding to points p_i and p_j intersect, then corresponding nodes v_i and v_j share an edge if and only if their sensing intersection area contains at least one tag. Unlike the UDG case, the nodes that belong to each $N^i(v_k)$, $1 \leq i \leq 3$ may no longer form a *clique* and we can no longer claim that Equation (6.2) holds true. However, if we assume that at least a fraction of the nodes in each $N^i(v_k)$, $1 \leq i \leq 3$ form a clique and the sum of the weights of the nodes in the clique is at least a fraction $(\frac{1}{\alpha}, \alpha > 1)$, of the sum of the weights of all nodes in each segment, we have:

$$OICN \geq \max_{1 \leq i \leq 3} \left[\sum_{u \in N^i(v_k)} \frac{|I(u)|}{\alpha} \right] + |I(v_k)| \tag{6.5}$$

Thus, from Equations (6.3) and (6.5), it follows: $\frac{AICN}{OICN} \leq \max(3\alpha, (K+3))$.

In case of UDG we assume reader density determines that $K \leq 5$. In case of RFIDG, if we assume that the tag density is such that the sum of the weights of the nodes in segments S_1, S_2 and S_3 is at least 20% of the sum of the weights of all nodes in segments S_1, S_2 and S_3 respectively, (i.e. $\alpha \leq 5$), we have $\frac{AICN}{OICN} \leq 15$.

6.5 Approximation Algorithms and Analysis

In this chapter, we propose an approximation algorithm with constant approximation ratio. In order to achieve it, we need to firstly divide the network into small grids.

Let G be the layout of one instance of RFIDG. W.l.o.g, we assume there m readers and n tags. For every reader r_i, denote its coordinate as (r_x^i, r_y^i). Similarly, for every tag $t_i \in G$, denote its coordinate as (t_x^i, t_y^i). Let $x_l = \min_{\forall x_i} x_i$, i.e., the leftmost $x-$coordinate. Similarly, let $y_b = \min_{\forall y_i} y_i$, i.e., the undermost $y - coordinate$. We then divide the whole layout into 1×1 grids according to the following algorithm:

Let $w(grid_{i,j}) = \sum_{v_k \in grid_{i,j}} w(v_k)$, we have the following lemma.

Lemma 6.

$$OICN \geq w(grid_{i,j})$$

for all grids obtained from 15

Proof. By the rule we divide G, each grid is a square of unit side length. Hence if v_i and v_j are within the same grid, their distance is at most $\sqrt{2}$. Reader-reader collision will occur if $I(v_i)$ and $I(v_j)$ intersect. All nodes Since each grid is a square of unit side length, nodes in the same grid are within each other's sensing range and thus forming a clique. □

Next, we define a *cell* as a collection of nearby *grids*. More specifically, $grid_{i,j}$

Algorithm 15 Grids Dividing Algorithm for RFIDG/UDG

```
1: compute $x_l, y_b$
2: $GRIDS = \emptyset$
3: for all $v_i \in G$ do
4:     $i = \lfloor x_i - x_l \rfloor$
5:     $j = \lfloor y_i - y_b \rfloor$
6:     if $grid_{i,j} \notin GRIDS$ then
7:         $grid_{i,j} = \emptyset$
8:         $GRIDS = GRIDS \cup grid_{i,j}$
9:     end if
10:    $grid_{i,j} = grid_{i,j} \cup v_i$
11: end for
```

and $grid_{i',j'}$ belong to the same *cell* if and only if $\lfloor i/3 \rfloor = \lfloor i'/3 \rfloor$ and $\lfloor j/3 \rfloor = \lfloor j'/3 \rfloor$. For instance, $grid_{5,7}$ and $grid_{3,6}$ belong to the same cell while $grid_{5,7}$ and $grid_{6,7}$ are not. Easy to see,

6.6 Experimental Results

As primarily experiments, we implemented the distributed and centralized coloring algorithms and compared them to the optimal solution using WSNet [21], an event-driven simulator for large scale Wireless Sensor Networks. As to fairly evaluate the performance under various network scenarios, we considered a dense RFID system where 10 readers were randomly deployed with uniform distribution on a square network of dimension 100m $\times$ 100m. We set the reader-to-tag communication and sensing range to 10m, and set the reader-to-reader communication range to 20m. For each of the 100 simulations per scenario, we computed the optimal solution and recorded the difference between the optimal and our algorithm's result.

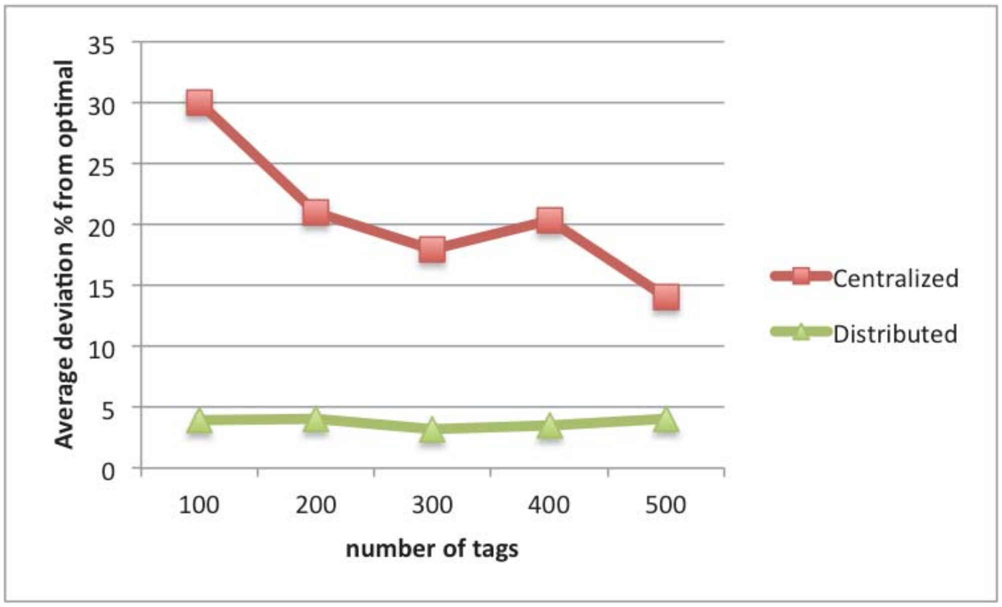


Figure 6.3: Avg. % deviation of approx. schedule from the optimal.

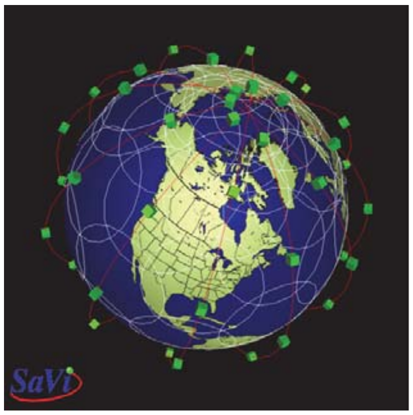
Fig. 6.3 presents the average percentage deviation from the optimal solution for each scenario in terms of efficiency. As we can observe, the centralized protocol achieves performances close to the optimal and the more readers, the closer to the centralized approach our distributed solution is. These first results let us expect interesting behavior in terms of throughput and fairness (evaluation left for future work), especially in presence of high tag mobility.

Chapter 7

ON UPPER AND LOWER BOUNDS OF IDENTIFYING CODE SET FOR SOCCER BALL GRAPH WITH APPLICATION TO SATELLITE DEPLOYMENT

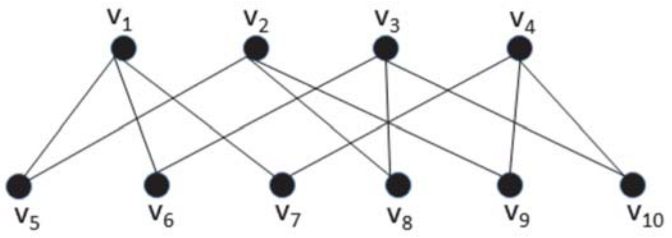
7.1 Introduction

In this chapter, we study an event monitoring problem with satellites as sensors. The events that we focus on may be environmental (drought/famine), social/political (social unrest/war) or extreme events (earthquakes/tsunamis). Such events take place in *regions* on the surface of the earth, where a region may be a continent, a country, or a set of neighboring countries. The sensors that we envisage for monitoring such events are satellites placed in orbits surrounding the earth. A satellite constellation that can be deployed for such monitoring purposes is shown in Fig. 7.1a. Examples of such constellations include the Global Positioning System (GPS) for navigation, the Iridium and Globalstar satellite telephony, and the Disaster Monitoring Constellation (DMC) for remote sensing. In particular, DMC is designed to provide earth imaging for disaster relief and was used extensively to monitor the impact of the Indian Ocean Tsunami in December 2004, Hurricane Katrina in August 2005, and several other floods, fires and disasters. The problem that we address in this chapter is directly relevant to the services being provided by organizations such as the DMC.



(a) Satellite Constellation Covering Earth

(b) A Truncated Icosahedron



(c) Graph with Identifying Code Set $\{v_1, v_2, v_3, v_4\}$

Figure 7.1: Satellites as sensors and soccer ball as a model of planet earth

For Earth's spherical structure, we use a soccer ball as a model of the planet Earth. In technical terms, a standard soccer ball is a *truncated icosahedron* with 12 pentagonal and 20 hexagonal patches [31] (shown in Fig. 7.1b as black and white patches). We associate a patch on the surface of the ball with a region on the surface of the Earth. Accordingly, in our model, the surface of the Earth is partitioned into 32 regions. We assume that the coverage area of a satellite corresponds to a patch (region) and events are confined to a region. With such a framework, it is clear that with 32 satellites (one per region), all the 32 regions can be effectively monitored. However, if we assume that the *impact* of an event in one region will *spill* into its neighboring regions, and as such there will be indicators of such events in neighboring regions, then a significantly lower number of satellites may be sufficient for effective monitoring of all the regions. As an example of impact of an event spilling over to neighboring regions, one can think of a situation where war breaking out in one region can trigger an exodus of refugees to the neighboring regions. As these sensors are expensive, one would like to deploy as few sensors as possible, subject to the constraint that all the regions can be effectively monitored. In the following, we discuss *Identifying Codes* [29] that can be utilized for this purpose. In particular, we will show that ten satellites are *sufficient* to *effectively monitor* 32 regions in the sense that, if an event breaks out in a region, that region can be *uniquely identified*. In fact there exists 26 different ways of deploying ten satellites that will achieve the effective monitoring task. Moreover, we will establish that the effective monitoring task cannot be accomplished by deployment of *fewer than* nine satellites.

In this chapter we have assumed that the regions have only two different regular shapes - hexagons and pentagons. This assumption may appear to be too restrictive, in the sense that in reality, regions on the surface of the earth may have irregular shapes. As our analysis is based on the structure of the graph (the SBG graph construction rule from the regions is described in section 7.2), as long as the graph structure arising out of irregular shaped regions remains the same as the structure of the SBG, the shape of the regions (whether regular or irregular, only hexagonal/pentagonal or some other contours) are irrelevant.

The notion of *Identifying Codes* [29] has been established as a useful concept for optimizing sensor deployment in multiple domains. In this chapter, we use Identifying Code of the *simplest form* and define it as follows. *A vertex set V' of a graph $G = (V, E)$ is defined as the Identifying Code Set (ICS) for the vertex set V, if for all $v \in V$, $N^+[v] \cap V'$ is unique where, $N^+[v] = v \cup N(v)$ and $N(v)$ represents the set of nodes adjacent to v in $G = (V, E)$.* The *Minimum Identifying Code Set* (MICS) problem is to find the Identifying Code Set of *smallest cardinality.* The vertices of the set V' may be viewed as *alphabets* of the code, and the *string* made up with the alphabets of $N^+[v]$ may be viewed as the unique "code" for the node v. For instance, Consider the graph $G = (V, E)$ shown in Fig. 7.1c. In this graph $V' = \{v_1, v_2, v_3, v_4\}$ is an ICS as it can be seen from Table 7.1 that $N^+[v] \cap V'$ is *unique* for all $v_i \in V$.

$N^+[v_1] \cap V' = \{v_1\}$	$N^+[v_2] \cap V' = \{v_2\}$
$N^+[v_3] \cap V' = \{v_3\}$	$N^+[v_4] \cap V' = \{v_4\}$
$N^+[v_5] \cap V' = \{v_1, v_2\}$	$N^+[v_6] \cap V' = \{v_1, v_3\}$
$N^+[v_7] \cap V' = \{v_1, v_4\}$	$N^+[v_8] \cap V' = \{v_2, v_3\}$
$N^+[v_9] \cap V' = \{v_2, v_4\}$	$N^+[v_{10}] \cap V' = \{v_3, v_4\}$

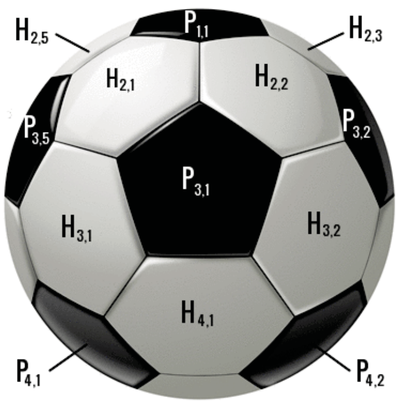
Table 7.1: $N^+[v] \cap V'$ results for all $v \in V$ for the graph in Fig. 7.1c

From the soccer ball, we construct a graph (referred to as a Soccer Ball Graph, SBG) where each of the 32 regions is represented as a node and two nodes have an edge between them if the corresponding regions share a boundary. The construction rules for the SBG are given in section 7.2 and a two dimension layout of the SBG is shown in Fig. 7.2. We establish that the upper and lower bounds of the MICS problem for the SBG are ten and nine respectively. Furthermore, we also establish that there exist at least 26 different Identifying Code Sets of size ten in the SBG.

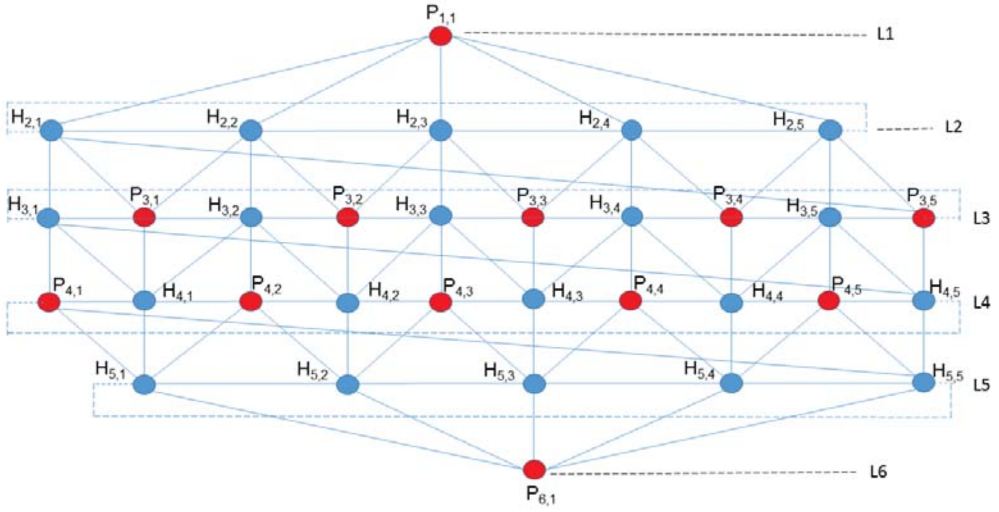
In the last few years a number of researchers have studied Identifying Codes and its applications in sensor network domains. Karpovsky *et. al.* [29] introduced the concept of Identifying Codes in [29] and provided results for Identifying Codes for graphs with specific topologies, such as binary cubes and trees. Using Identifying Codes, Laifenfeld *et. al.* studied covering problems in [33] and joint monitoring and routing in wireless sensor networks in [34]. Ray *et. al.* in [48] generalized the concept of Identifying Codes, to incorporate robustness properties to deal with faults in sensor networks. Charon *et. al.* in [8, 9], studied complexity issues related to computation of minimum Identifying Codes for graph and showed that in several types of graph, the problem is NP-hard. Approximation algorithms for computation of Identifying Codes for some special types of graphs are presented in [61, 53]. Auger in [2] show that the problem can be solved in linear time if the graph happend to be a tree, but even for a planar graph the problem remains NP-complete. Topological and combinatorial properties of soccer balls have been studied extensively in [31].

7.2 Problem Formulation

A Soccer Ball Graph (SBG) $G = (V, E)$ is defined in the following way. The graph comprises of 32 nodes and 90 edges. The 32 nodes correspond to 32 patches (20 hexagonal and 12 pentagonal) of the soccer ball and two nodes in the graph have an edge between them if the corresponding patches share a boundary. A graph can have different layouts on a two dimensional plane. We show one layout of the SBG in Fig. 7.2 where the nodes are labeled using a set of rules. The soccer ball, placed on a two dimensional plane (as shown in Fig. 7.2 (a)), has a pentagonal patch on top. There are five hexagonal patches adjacent to this pentagonal patch. We consider a *layering scheme*, where the node corresponding to pentagonal patch on top is in Layer 1 (L1), the six nodes corresponding to six hexagonal patches adjacent to the pentagonal patch on top are in Layer 2 (L2) and so on. Following this layering scheme, all 32 nodes can be assigned to six layers, L1 through L6, as shown in Fig.



(a) Nodes corresponding to the patches on the soccer


$P_{1,1}$
L1
$H_{2,1}$ $H_{2,2}$ $H_{2,3}$ $H_{2,4}$ $H_{2,5}$
L2
$H_{3,1}$ $P_{3,1}$ $H_{3,2}$ $P_{3,2}$ $H_{3,3}$ $P_{3,3}$ $H_{3,4}$ $P_{3,4}$ $H_{3,5}$ $P_{3,5}$
L3
$P_{4,1}$ $H_{4,1}$ $P_{4,2}$ $H_{4,2}$ $P_{4,3}$ $H_{4,3}$ $P_{4,4}$ $H_{4,4}$ $P_{4,5}$ $H_{4,5}$
L4
$H_{5,1}$ $H_{5,2}$ $H_{5,3}$ $H_{5,4}$ $H_{5,5}$
L5
$P_{6,1}$
L6

(b) Soccer Ball Graph

Figure 7.2: Soccer Ball and the corresponding graph

7.2 (b). In this scheme, one node is assigned to L1, five nodes to L2, ten nodes to L3, ten nodes to L4, five nodes to L5, and one node to L6. There is only one pentagonal node in layers L1 and L6 and we refer to these two nodes as $P_{1,1}$ and $P_{6,1}$ respectively. There are five hexagonal nodes in layers L2 and L5 and we refer to these nodes as $H_{2,i}, 1 \leq i \leq 5$ and $H_{5,i}, 1 \leq i \leq 6$ respectively. There are five hexagonal and five pentagonal nodes in layers L3 and L4 and we refer to these nodes as $H_{i,j}, i = 4, 5, 1 \leq j \leq 5$ and $P_{i,j}, i = 4, 5, 1 \leq j \leq 5$ respectively. The vertex set V of the SBG, is divided into two subsets, P (for Pentagon) and H (for Hexagon), with 12 and 20 members respectively.

It may be noted from Fig. 7.2, that P-type nodes appear only on layers 1, 3, 4 and 6 and H-type nodes appear only on layers 2, 3, 4 and 5. The SBG $G = (V, E) = ((P \cup H), E)$ is formally defined as follows:

$P = \{P_{1,1}\} \cup \{P_{i,j}, 3 \leq i \leq 4, 1 \leq j \leq 5\} \cup \{P_{6,1}\}$ and

$H = \{H_{i,j}, 2 \leq i \leq 5, 1 \leq j \leq 5\}$

The edge set E is divided into 17 subsets, i.e., $E = \cup_{i=1}^{17} E_i$. Each subset is defined in Table 7.2.

With the formal definition of the SBG complete, it may be observed that the problem of determining the fewest number of satellites necessary to uniquely identify the region (among 32 regions) where a significant event has taken place is equivalent to computation of the Minimum Identifying Code Set problem for the SBG.

`Graph Coloring with Seepage (GCS) Problem`: The MICS computation problem can be viewed as a novel variation of the classical Graph Coloring problem. We will refer to this version as the *Graph Coloring with Seepage (GCS)* problem. In the classical graph coloring problem, when a color is *assigned* (or injected) to a node, only that node is colored. The goal of the classical graph coloring problem to use as few distinct colors as possible such that (i) every node receives a color, and (ii) no two adjacent nodes of the graph have the same color. In the GCS problem, when a color is assigned (or injected) to a node, not only that node receives the color, but also the color *seeps* into all the adjoining nodes. As a node v_i may be adjacent to two other nodes v_j and v_k in the graph, if the color red is injected to v_j, not only v_j will become red, but also v_i will become red as it is adjacent to v_j. Now if the color blue

SBG Edge Construction
$E_1 = \{(P_{1,1}, H_{2,j}), 1 \leq j \leq 5\}$
$E_2 = \{(P_{6,1}, H_{5,j}), 1 \leq j \leq 5\}$
$E_3 = \{(H_{i,j}, H_{i,(j+1) mod\ 5}), i = 2, i = 5, 1 \leq j \leq 5\}$
$E_4 = \{(H_{i,j}, P_{i,j}), i = 3, 1 \leq j \leq 5\}$
$E_5 = \{(P_{i,j}, H_{i,(j+1) mod\ 5}), i = 3, 1 \leq j \leq 5\}$
$E_6 = \{(H_{i,j}, P_{i,(j+1) mod\ 5}), i = 4, 1 \leq j \leq 5\}$
$E_7 = \{(P_{i,j}, H_{i,j}), i = 4, 1 \leq j \leq 5\}$
$E_8 = \{(H_{2,j}, H_{3,j}), 1 \leq j \leq 5\}$
$E_9 = \{(H_{2,j}, P_{3,(j-1) mod\ 5}), 1 \leq j \leq 5\}$
$E_{10} = \{(H_{2,j}, P_{3,j}), 1 \leq j \leq 5\}$
$E_{11} = \{(H_{3,j}, P_{4,j}), 1 \leq j \leq 5\}$
$E_{12} = \{(H_{3,j}, H_{4,(j-1) mod\ 5}), 1 \leq j \leq 5\}$
$E_{13} = \{(H_{3,j}, H_{4,j}), 1 \leq j \leq 5\}$
$E_{14} = \{(P_{3,j}, H_{4,j}), 1 \leq j \leq 5\}$
$E_{15} = \{(H_{4,j}, H_{5,j}), 1 \leq j \leq 5\}$
$E_{16} = \{(P_{4,j}, H_{5,j}), 1 \leq j \leq 5\}$
$E_{17} = \{(P_{4,j}, H_{5,(j-1) mod\ 5}), 1 \leq j \leq 5\}$

Table 7.2: 17 subsets of the edge set E of the SBG graph $G = (V, E) = ((P \cup H), E)$

is injected to v_k, not only v_k will become blue, but also the color blue will seep in to v_i as it is adjacent to v_k. Since v_i was already colored red (due to seepage from v_j), after color seepage from v_k, it's color will be a *combination of red and blue.* At this point all three nodes v_i, v_j and v_k have a color and all of them have distinct colors (red, blue and the combination of the two). *The color assigned to a node may be due to, (i) only injection to that node, (ii) only seepage from other adjoining nodes and (iii) a combination of injection and seepage.* The colors injected at the nodes will be referred to as *atomic* colors. The colors formed by the combination of two or more atomic colors are referred to as *composite* colors. The colors injected at the nodes (atomic colors) are all *unique.* The goal of the GCS problem is to inject colors to as few nodes as possible, such that (i) every node receives a color, and (ii) no two nodes of the graph have the same color.

Suppose that the node set V' is an ICS of a graph $G = (V, E)$ and $|V'| = p$. In this case if p distinct colors are injected to V' (one distinct atomic color to one node of V'), then as by the definition of ICS for all $v \in V$, if $N^+(v) \cap V'$ is unique, all nodes of $G = (V, E)$ will have a unique color (either atomic or composite). Thus computation of MICS is equivalent to solving the GCS problem.

7.3 Upper Bound of MICS of SBG

In this section, we first show that MICS of the SBG is at most 10 and there exists at least 26 ICS of size ten.

Theorem 7. *The MICS of SBG is at most ten.*

Proof. Inject colors A, B, C, D, E to the nodes $H_{2,j}, 1 \leq j \leq 5$ and colors E, F, G, H, I, J to the nodes $H_{5,j}, 1 \leq j \leq 5$. Injection of 10 different colors at these 10 nodes, will cause color seepage to all other nodes of SBG. The color seepage will be constrained

Node: Color	Node: Color	Node: Color	Node: Color
$P_{1,1}$: $ABCDE$	$H_{2,1}$: A^*BE	$H_{2,2}$: AB^*C	$H_{2,3}$: BC^*D
$H_{2,4}$: CD^*E	$H_{2,5}$: DE^*A	$H_{3,1}$: A	$P_{3,1}$: AB
$H_{3,2}$: B	$P_{3,2}$: BC	$H_{3,3}$: C	$P_{3,3}$: CD
$H_{3,4}$: D	$P_{3,4}$: DE	$H_{3,5}$: E	$P_{3,5}$: AE
$P_{4,1}$: JF	$H_{4,1}$: F	$P_{4,2}$: FG	$H_{4,2}$: G
$P_{4,3}$: GH	$H_{4,3}$: H	$P_{4,4}$: HI	$H_{4,4}$: I
$P_{4,5}$: IJ	$H_{4,5}$: J	$H_{5,1}$: JF^*G	$H_{5,2}$: FG^*H
$H_{5,3}$: GH^*I	$H_{5,4}$: HI^*J	$H_{5,5}$: IJ^*F	$P_{6,1}$: $FGHIJ$

Table 7.3: Color assignment at nodes after seepage in the SBG

by the topological structure of the SBG. It may be verified that because of the constraint imposed by the SBG structure, and the fact that seepage takes place only to the neighbors of the node where a color is injected, the 32 nodes of the SBG will have the color assignment shown in Table 7.3. In the entries of Table 7.3, $H_{2,1}$: A^*BE implies that the color A was *injected* at the node $H_{2,1}$ and the colors B and E *seeped* into the node $H_{2,1}$, from the adjacent nodes $H_{2,2}$ and $H_{2,5}$ respectively, where the colors B and E were injected. In general, if an alphabet A through E (representing distinct colors), appears *with* a * as a part of a string attached to a node (such as $H_{2,1}$), it implies that the color was *injected* at that node. On the other hand, if an alphabet appears *without* a * as a part of a string attached to a node, it implies that the color *seeped* into that node from one of the neighboring nodes. It may be verified that the color assignment to the nodes, as shown in Table 7.3 is *unique* (i.e, no two nodes have the same color or *strings* assigned to them). $\square$

Theorem 8. *At least 26 distinct ICSs of size ten exist for SBG.*

Proof. The 26 different ways in which ten colors can be injected into ten nodes of the SBG such that every node of the SBG receives a unique color can be divided into four classes.

- *Class I:* Inject colors A, B, C, D, E to the nodes $H_{2,j}, 1 \leq j \leq 5$ and colors E, F, G, H, I, J to the nodes $H_{5,j}, 1 \leq j \leq 5$. As shown in Table 7.3, such an injection ensures that each of the 32 nodes of the SBG receives a unique color. It may be observed that the node set where the colors are injected in this Class all have degree six, corresponding to hexagonal patches on the surface of the soccer ball. Only one ICS of the 26, belongs to Class I. It may be noted that as the nodes where colors are being injected correspond to the regions where satellites are being deployed, a permutation of the color set A, B, C, D, E is not important here because if either color A or B is injected at node $H_{2,1}$, it implies that one satellite is deployed to monitor the region represented by node $H_{2,1}$.

- *Class II:* The node set where the colors injected are in this Class is made up of six nodes of degree five (corresponding to the pentagonal patches of the soccer ball) and four nodes of degree six (corresponding to the hexagonal patches of the soccer ball). This Class can be subdivided into two sub-classes and we will refer to them as *Class II-A* and *Class II-B* respectively. As seen in Fig. 7.2, the SBG graph is somewhat symmetric in the sense that the layers 4, 5 and 6 are close to being mirror images of layers 1, 2 and 3. Because of this symmetry, the Class II-A color injections are mirror images of the Class II-B color injection. Accordingly, in this section we will focus our discussion primarily on Class II-A, as color injection for class II-B be can be obtained easily from color injection in Class IIA. We introduce the notion of a *motif*, and by motif we imply a set of either P-type (degree five) or H-type (degree six) nodes. It will be clear from further discussion that the Class II-A solutions comprise of one P-type motif and one H-type motif. These two motifs *complement* each other to produce a solution together. The motif-pairs can be *slid* along the structure of the SBG to produce a set of five solutions that make up the Class II-A. The five solutions that make up the Class II-B can be constructed in a similar fashion.

Node	Color
$P_{1,1}$	$P^c_{1,1}$
$H_{2,j}$	$P^c_{1,1}P^c_{3,j}$
$H_{2,(j+1)mod\ 5}$	$P^c_{1,1}P^c_{3,j}P^c_{3,(j+1)mod\ 5}$
$H_{2,(j+2)mod\ 5}$	$P^c_{1,1}P^c_{3,(j+1)mod\ 5}$
$H_{2,(j+3)mod\ 5}$	$P^c_{1,1}H^c_{3,(j+3)mod\ 5}$
$H_{2,(j+4)mod\ 5}$	$P^c_{1,1}H^c_{3,(j+4)mod\ 5}$
$H_{3,j}$	$P^c_{3,j}P^c_{4,j}$
$P_{3,j}$	$P^c_{3,j}$
$H_{3,(j+1)mod\ 5}$	$P^c_{3,j}P^c_{3,(j+1)mod\ 5}P^c_{4,(j+1)mod\ 5}$
$P_{3,(j+1)mod\ 5}$	$P^c_{3,(j+1)mod\ 5}$
$H_{3,(j+2)mod\ 5}$	$P^c_{3,(j+1)mod\ 5}P^c_{4,(j+2)mod\ 5}$
$P_{3,(j+2)mod\ 5}$	$H^c_{3,(j+3)mod\ 5}$
$H_{3,(j+3)mod\ 5}$	$H^c_{3,(j+3)mod\ 5}H^c_{4,(j+3)mod\ 5}$
$P_{3,(j+3)mod\ 5}$	$H^c_{3,(j+3)mod\ 5}H^c_{4,(j+3)mod\ 5}H^c_{3,(j+4)mod\ 5}H_{5,(j+3)mod\ 5}$
$H_{3,(j+4)mod\ 5}$	$H^c_{3,(j+4)mod\ 5}H^c_{4,(j+3)mod\ 5}$
$P_{3,(j+4)mod\ 5}$	$H^c_{4,(j+4)mod\ 5}$
$P_{4,j}$	$P^c_{4,j}$
$H_{4,j}$	$P^c_{3,j}P^c_{4,j}P^c_{4,(j+1)mod\ 5}$
$P_{4,(j+1)mod\ 5}$	$P^c_{4,(j+1)mod\ 5}$
$H_{4,(j+1)mod\ 5}$	$P^c_{3,(j+1)mod\ 5}P^c_{4,(j+1)mod\ 5}P^c_{4,(j+2)mod\ 5}$

$P_{4,(j+2)mod\ 5}$	$P^c_{4,(j+2)mod\ 5}$
$H_{4,(j+2)mod\ 5}$	$P^c_{4,(j+2)mod\ 5}H^c_{3,(j+3)mod\ 5}$
$P_{4,(j+3)mod\ 5}$	$H^c_{3,(j+3)mod\ 5}H^c_{4,(j+3)mod\ 5}H^c_{5,(j+3)mod\ 5}$
$H_{4,(j+3)mod\ 5}$	$H^c_{4,(j+3)mod\ 5}H^c_{3,(j+3)mod\ 5}H^c_{3,(j+4)mod\ 5}H^c_{5,(j+3)mod\ 5}$
$P_{4,(j+4)mod\ 5}$	$H^c_{4,(j+4)mod\ 5}H^c_{3,(j+3)mod\ 5}H^c_{5,(j+3)mod\ 5}$
$H_{4,(j+4)mod\ 5}$	$P^c_{4,j}H^c_{4,(j+4)mod\ 5}$
$H_{5,j}$	$P^c_{4,j}P^c_{4,(j+1)mod\ 5}$
$H_{5,(j+1)mod\ 5}$	$P^c_{4,(j+1)mod\ 5}P^c_{4,(j+2)mod\ 5}$
$H_{5,(j+2)mod\ 5}$	$P^c_{4,(j+2)mod\ 5}H^c_{5,(j+3)mod\ 5}$
$H^c_{5,(j+3)mod\ 5}$	$H^c_{4,(j+3)mod\ 5}$
$H_{5,(j+4)mod\ 5}$	$P^c_{4,j}H^c_{5,(j+3)mod\ 5}$
$P_{6,1}$	$H^c_{5,(j+3)mod\ 5}$

Table 7.4: Node versus Color assignment for Class II-A ICS

Color	Node
$H^c_{3,(j+3)mod\ 5}$	$P_{3,(j+2)mod\ 5}$
$H^c_{4,(j+4)mod\ 5}$	$P_{3,(j+4)mod\ 5}$
$H^c_{5,(j+3)mod\ 5}$	$P_{6,1}$
$H^c_{3,(j+3)mod\ 5}H^c_{4,(j+3)mod\ 5}$	$H_{3,(j+3)mod\ 5}$
$H^c_{3,(j+4)mod\ 5}H^c_{4,(j+3)mod\ 5}$	$H_{3,(j+4)mod\ 5}$
$H^c_{5,(j+3)mod\ 5}H^c_{4,(j+3)mod\ 5}$	$H_{5,(j+3)mod\ 5}$
$H^c_{3,(j+3)mod\ 5}H^c_{4,(j+3)mod\ 5}H^c_{3,(j+4)mod\ 5}$	$P_{3,(j+3)mod\ 5}$
$H^c_{3,(j+3)mod\ 5}H^c_{4,(j+3)mod\ 5}H^c_{5,(j+3)mod\ 5}$	$P_{4,(j+3)mod\ 5}$
$H^c_{4,(j+4)mod\ 5}H^c_{3,(j+3)mod\ 5}H^c_{5,(j+3)mod\ 5}$	$P_{4,(j+4)mod\ 5}$
$H^c_{4,(j+3)mod\ 5}H^c_{3,(j+3)mod\ 5}H^c_{3,(j+4)mod\ 5}H^c_{5,(j+3)mod\ 5}$	$H_{4,(j+3)mod\ 5}$
$H^c_{3,(j+3)mod\ 5}P^c_{1,1}$	$H_{2,(j+3)mod\ 5}$

$H^c_{3,(j+4)mod\ 5}P^c_{1,1}$	$H_{2,(j+4)mod\ 5}$
$H^c_{3,(j+3)mod\ 5}P^c_{4,(j+2)mod\ 5}$	$H_{4,(j+2)mod\ 5}$
$H^c_{4,(j+4)mod\ 5}P^c_{4,j}$	$H_{4,(j+4)mod\ 5}$
$H^c_{5,(j+3)mod\ 5}P^c_{4,(j+2)mod\ 5}$	$H_{5,(j+2)mod\ 5}$
$H^c_{5,(j+3)mod\ 5}P^c_{4,j}$	$H_{5,(j+4)mod\ 5}$
$P^c_{1,1}$	$P_{1,1}$
$P^c_{3,j}$	$P_{3,j}$
$P^c_{3,(j+1)mod\ 5}$	$P_{3,(j+1)mod\ 5}$
$P^c_{4,(j+1)mod\ 5}$	$P_{4,(j+1)mod\ 5}$
$P^c_{4,(j+2)mod\ 5}$	$P_{4,(j+2)mod\ 5}$
$P^c_{1,1}P^c_{3,j}$	$H_{2,j}$
$P^c_{1,1}P^c_{3,(j+1)mod\ 5}$	$H_{2,(j+2)mod\ 5}$
$P^c_{3,j}P^c_{4,j}$	$H_{3,j}$
$P^c_{3,(j+1)mod\ 5}P^c_{4,(j+2)mod\ 5}$	$H_{3,(j+2)mod\ 5}$
$P^c_{4,j}P^c_{4,(j+1)mod\ 5}$	$H_{5,j}$
$P^c_{4,(j+1)mod\ 5}P^c_{4,(j+2)mod\ 5}$	$H_{5,(j+1)mod\ 5}$
$P^c_{1,1}P^c_{3,j}P^c_{3,(j+1)mod\ 5}$	$H_{2,(j+1)mod\ 5}$
$P^c_{3,j}P^c_{3,(j+1)mod\ 5}P^c_{4,(j+1)mod\ 5}$	$H_{3,(j+1)mod\ 5}$
$P^c_{3,j}P^c_{4,j}P^c_{4,(j+1)mod\ 5}$	$H_{4,j}$
$P^c_{3,(j+1)mod\ 5}P^c_{4,(j+1)mod\ 5}P^c_{4,(j+2)mod\ 5}$	$H_{4,(j+1)mod\ 5}$

Table 7.5: Color versus Node assignment for Class II-A ICS

For the ICS that belong to Class II, the P-type motif is made up of the set of six nodes $\{P_{1,1}, P_{3,j}, P_{3,(j+1)mod\ 5}, P_{4,j}, P_{4,(j+1)mod\ 5}, P_{4,(j+2)mod\ 5}\}$. The H-type motif that complements the P-type motif is made up of the set of four nodes $\{H_{3,(j+3)mod\ 5}, H_{3,(j+4)mod\ 5}, H_{4,(j+3)mod\ 5}, H_{5,(j+3)mod\ 5}\}$. One complete solution (i.e., ICS) is obtained by choosing a value of $j, 1 \leq j \leq 5$. The Fig. 7.3 (a) and (b) shows the solutions with $j = 1$ and $j = 2$ respectively. As shown in Fig. 7.3, changing the index j from 1 to 2, has the effect of sliding the motif along the structure of the SBG. By changing j from 1 through 5 (i.e., sliding the motif 5 times), 5 different ICS can be computed. The colors that will be associated with the nodes of the SBG, if they are injected at the motif nodes, are shown in Table 7.3. The first column of the table indicates the node and the second column provides the color assigned to that node. For example, in row 3 of Table 7.3, the node $H_{2,(j+1)mod\ 5}$ receives the colors injected at motif nodes $P_{1,1}, P_{3,j}, P_{3,(j+1)mod\ 5}$ and is denoted by $P^c_{1,1}P^c_{3,j}P^c_{3,(j+1)mod\ 5}$. It may be verified that every node of the SBG has a *color associated with it and no two nodes have the same color assignment*. For ease of verification, we have presented another Table 7.3 which may be viewed as an "inverse" of Table 7.3, in the sense that, the first column provides the color assigned to a node and the second column represents the corresponding node. As the color associated with a node may be viewed as a string $P^c_{1,1}P^c_{3,j}P^c_{3,(j+1)mod\ 5}$, we have presented them in the *lexicographic* order. The verification of the fact that every node of SBG receives a unique color is much simpler now, as one has to verify only among *similar* strings (made up of H only, P only or combination of H and P) of *same length*.

• *Class III:* As in Class II, the Class III ICS is made up of six nodes of degree five and four nodes of degree six. Moreover, this Class also can be subdivided into two sub-classes and we will refer to them as Class III-A and Class III-B respectively. In this section we will restrict our discussion on Class III-A, as color injection for Class III-B can be obtained as a mirror image of Class III-A. It will be clear from further discussion that, as in Class II, the Class III solutions also comprise of one P-type motif and one H-type motif and they complement each other to produce a solution together. As in Class II, the motif-pairs can be slid along the structure of the SBG to produce a set of five solutions that make up the Class III-A. The five solutions that make up the Class III-B can be constructed in a similar fashion.

Color	Node
$P_{1,1}$	$P^c_{1,1}$
$H_{2,j}$	$P^c_{1,1}P^c_{3,j}P^c_{3,(j+4)mod\ 5}$
$H_{2,(j+1)mod\ 5}$	$P^c_{1,1}P^c_{3,j}P^c_{3,(j+1)mod\ 5}$
$H_{2,(j+2)mod\ 5}$	$P^c_{1,1}P^c_{3,(j+1)mod\ 5}P^c_{3,(j+2)mod\ 5}$
$H_{2,(j+3)mod\ 5}$	$P^c_{1,1}P^c_{3,(j+2)mod\ 5}$
$H_{2,(j+4)mod\ 5}$	$P^c_{1,1}P^c_{3,(j+4)mod\ 5}$
$H_{3,j}$	$P^c_{3,j}P^c_{3,(j+4)mod\ 5}$
$P_{3,j}$	$P^c_{3,j}$
$H_{3,(j+1)mod\ 5}$	$P^c_{3,j}P^c_{3,(j+1)mod\ 5}P^c_{4,(j+1)mod\ 5}$
$P_{3,(j+1)mod\ 5}$	$P^c_{3,(j+1)mod\ 5}$
$H_{3,(j+2)mod\ 5}$	$P^c_{3,(j+1)mod\ 5}P^c_{3,(j+2)mod\ 5}$
$P_{3,(j+2)mod\ 5}$	$P^c_{3,(j+2)mod\ 5}$
$H_{3,(j+3)mod\ 5}$	$P^c_{3,(j+2)mod\ 5}H^c_{4,(j+3)mod\ 5}$
$P_{3,(j+3)mod\ 5}$	$H^c_{4,(j+3)mod\ 5}$
$H_{3,(j+4)mod\ 5}$	$H^c_{4,(j+3)mod\ 5}P^c_{3,(j+4)mod\ 5}$
$P_{3,(j+4)mod\ 5}$	$P^c_{3,(j+4)mod\ 5}$
$P_{4,j}$	$H^c_{5,(j+4)mod\ 5}$
$H_{4,j}$	$P^c_{4,(j+1)mod\ 5}P^c_{3,j}$
$H_{4,(j+1)mod\ 5}$	$P^c_{4,(j+1)mod\ 5}P^c_{3,(j+1)mod\ 5}$
$P_{4,(j+1)mod\ 5}$	$P^c_{4,(j+1)mod\ 5}$
$P_{4,(j+2)mod\ 5}$	$H^c_{5,(j+2)mod\ 5}$
$H_{4,(j+2)mod\ 5}$	$P^c_{3,(j+2)mod\ 5}H^c_{5,(j+2)mod\ 5}$
$P_{4,(j+3)mod\ 5}$	$H^c_{4,(j+3)mod\ 5}H^c_{5,(j+2)mod\ 5}H^c_{5,(j+3)mod\ 5}$

$H_{4,(j+3) mod\ 5}$	$H^c_{4,(j+3) mod\ 5} H^c_{5,(j+3) mod\ 5}$
$P_{4,(j+4) mod\ 5}$	$H^c_{4,(j+3) mod\ 5} H^c_{5,(j+3) mod\ 5} H^c_{5,(j+4) mod\ 5}$
$H_{4,(j+4) mod\ 5}$	$P^c_{3,(j+4) mod\ 5} H^c_{5,(j+4) mod\ 5}$
$H_{5,j}$	$H^c_{5,(j+4) mod\ 5} P^c_{4,(j+1) mod\ 5}$
$H_{5,(j+1) mod\ 5}$	$H^c_{5,(j+2) mod\ 5} P^c_{4,(j+1) mod\ 5}$
$H_{5,(j+2) mod\ 5}$	$H^c_{5,(j+2) mod\ 5} H^c_{5,(j+3) mod\ 5}$
$H_{5,(j+3) mod\ 5}$	$H^c_{5,(j+3) mod\ 5} H^c_{5,(j+4) mod\ 5} H^c_{5,(j+2) mod\ 5} H^c_{4,(j+3) mod\ 5}$
$H_{5,(j+4) mod\ 5}$	$H^c_{5,(j+4) mod\ 5} H^c_{5,(j+3) mod\ 5}$
$P_{6,1}$	$H^c_{5,(j+2) mod\ 5} H^c_{5,(j+3) mod\ 5} H^c_{5,(j+4) mod\ 5}$

Table 7.6: Node versus Color assignment for Class III-A ICS

Hehe, The P-type motif is made up of the set of six nodes $\{P_{1,1}, P_{3,j}, P_{3,(j+1) mod\ 5}, P_{3,(j+2) mod\ 5}, P_{3,(j+4) mod\ 5}, P_{4,(j+1) mod\ 5}\}$. As shown in Fig. 7.4a, changing the index j from 1 to 5, has the effect of sliding the motif along the structure of the SBG. The H-type motif that complements the P-type motif is made up of the set of four nodes $\{H_{4,(j+3) mod\ 5}, H_{5,(j+2) mod\ 5}, H_{5,(j+3) mod\ 5}, H_{5,(j+4) mod\ 5}\}$.
One complete solution is obtained by choosing a value of $j, 1 \leq j \leq 5$. By moving the P-type and H-type motifs in tandem by changing the value of the index from 1 to 5, five different solutions can be obtained. The colors that will be associated with the nodes of the SBG, if the they are injected at the motif nodes, are shown in Table 7.6. As in the case of Class II, we present an "inverse" of Table 7.6 (Table 7.7), for the purpose of verification that every node of the SBG has a *unique color*.

Color	Node
$P_{1,1}$	$P^c_{1,1}$
$H_{2,j}$	$P^c_{1,1} P^c_{3,j} P^c_{3,(j+4) mod\ 5}$

$H_{2,(j+1)mod\ 5}$	$P^c_{1,1}P^c_{3,j}P^c_{3,(j+1)mod\ 5}$
$H_{2,(j+2)mod\ 5}$	$P^c_{1,1}P^c_{3,(j+1)mod\ 5}P^c_{3,(j+2)mod\ 5}$
$H_{2,(j+3)mod\ 5}$	$P^c_{1,1}P^c_{3,(j+2)mod\ 5}$
$H_{2,(j+4)mod\ 5}$	$P^c_{1,1}P^c_{3,(j+4)mod\ 5}$
$H_{3,j}$	$P^c_{3,j}P^c_{3,(j+4)mod\ 5}$
$P_{3,j}$	$P^c_{3,j}$
$H_{3,(j+1)mod\ 5}$	$P^c_{3,j}P^c_{3,(j+1)mod\ 5}P^c_{4,(j+1)mod\ 5}$
$P_{3,(j+1)mod\ 5}$	$P^c_{3,(j+1)mod\ 5}$
$H_{3,(j+2)mod\ 5}$	$P^c_{3,(j+1)mod\ 5}P^c_{3,(j+2)mod\ 5}$
$P_{3,(j+2)mod\ 5}$	$P^c_{3,(j+2)mod\ 5}$
$H_{3,(j+3)mod\ 5}$	$P^c_{3,(j+2)mod\ 5}H^c_{4,(j+3)mod\ 5}$
$P_{3,(j+3)mod\ 5}$	$H^c_{4,(j+3)mod\ 5}$
$H_{3,(j+4)mod\ 5}$	$H^c_{4,(j+3)mod\ 5}P^c_{3,(j+4)mod\ 5}$
$P_{3,(j+4)mod\ 5}$	$P^c_{3,(j+4)mod\ 5}$
$P_{4,j}$	$H^c_{5,(j+4)mod\ 5}$
$H_{4,j}$	$P^c_{4,(j+1)mod\ 5}P^c_{3,j}$
$P_{4,(j+1)mod\ 5}$	$P^c_{4,(j+1)mod\ 5}$
$H_{4,(j+1)mod\ 5}$	$P^c_{4,(j+1)mod\ 5}P^c_{3,(j+1)mod\ 5}$
$P_{4,(j+2)mod\ 5}$	$H^c_{5,(j+2)mod\ 5}$
$H_{4,(j+2)mod\ 5}$	$P^c_{3,(j+2)mod\ 5}H^c_{5,(j+2)mod\ 5}$
$P_{4,(j+3)mod\ 5}$	$H^c_{4,(j+3)mod\ 5}H^c_{5,(j+2)mod\ 5}H^c_{5,(j+3)mod\ 5}$
$H_{4,(j+3)mod\ 5}$	$H^c_{4,(j+3)mod\ 5}H^c_{5,(j+3)mod\ 5}$
$P_{4,(j+4)mod\ 5}$	$H^c_{4,(j+3)mod\ 5}H^c_{5,(j+3)mod\ 5}H^c_{5,(j+4)mod\ 5}$
$H_{4,(j+4)mod\ 5}$	$P^c_{3,(j+4)mod\ 5}H^c_{5,(j+4)mod\ 5}$
$H_{5,j}$	$H^c_{5,(j+4)mod\ 5}P^c_{4,(j+1)mod\ 5}$
$H_{5,(j+1)mod\ 5}$	$H^c_{5,(j+2)mod\ 5}P^c_{4,(j+1)mod\ 5}$

$H_{5,(j+2)mod\ 5}$	$H^c_{5,(j+2)mod\ 5}H^c_{5,(j+3)mod\ 5}$
$H_{5,(j+3)mod\ 5}$	$H^c_{5,(j+3)mod\ 5}H^c_{5,(j+4)mod\ 5}H^c_{5,(j+2)mod\ 5}H^c_{4,(j+3)mod\ 5}$
$H_{5,(j+4)mod\ 5}$	$H^c_{5,(j+4)mod\ 5}H^c_{5,(j+3)mod\ 5}$
$P_{6,1}$	$H^c_{5,(j+2)mod\ 5}H^c_{5,(j+3)mod\ 5}H^c_{5,(j+4)mod\ 5}$

Table 7.7: Color versus Node assignment for Class III-A ICS

- *Class IV:* As in Class I, the Class IV ICS are made up of 10 nodes of degree six (i.e., the nodes corresponding to hexagonal patches). This Class comprises of five ICS and cannot be subdivided like in Classes II and III.

This class comprises of two H-type motifs made up of five hexagonal nodes each. The first motif comprises of $\{H_{2,(j+1)mod\ 5}, H_{2,(j+2)mod\ 5}, H_{3,(j+1)mod\ 5}, H_{3,(j+2)mod\ 5}, H_{4,(j+1)mod\ 5}\}$. The other motif comprises of $\{H_{3,(j+4)mod\ 5}, H_{4,(j+3)mod\ 5}, H_{4,(j+4)mod\ 5}, H_{5,(j+3)mod\ 5}, H_{5,(j+4)mod\ 5}\}$. As shown in Fig. 7.4b, changing the index j from 1 to 5, has the effect of sliding the motif along the structure of the SBG. One complete solution is obtained by choosing a value of $j, 1 \leq j \leq 5$. By moving two H-type motifs in tandem, changing the value of the index from 1 to 5, five different solutions can be obtained. The colors that will be associated with the nodes of the SBG, if the they are injected at the motif nodes, are shown in Table 7.8. As in the case of Classes II and III, we present an "inverse" of Table 7.8 (Table 7.9) for the purpose of verification that every node of the SBG has a *unique color*.

Color	Node
$P_{1,1}$	$H^c_{2,(j+1)mod\ 5}H^c_{2,(j+2)mod\ 5}$
$H_{2,j}$	$H^c_{2,(j+1)mod\ 5}$
$H_{2,(j+1)mod\ 5}$	$H^c_{2,(j+1)mod\ 5}H^c_{3,(j+1)mod\ 5}H^c_{2,(j+2)mod\ 5}$
$H_{2,(j+2)mod\ 5}$	$H^c_{2,(j+1)mod\ 5}H^c_{2,(j+2)mod\ 5}H^c_{3,(j+2)mod\ 5}$

$H_{2,(j+3)mod\ 5}$	$H^c_{2,(j+2)mod\ 5}$
$H_{2,(j+4)mod\ 5}$	$H^c_{3,(j+4)mod\ 5}$
$H_{3,j}$	$H^c_{4,(j+4)mod\ 5}$
$P_{3,j}$	$H^c_{2,(j+1)mod\ 5}H^c_{3,(j+1)mod\ 5}$
$H_{3,(j+1)mod\ 5}$	$H^c_{3,(j+1)mod\ 5}H^c_{2,(j+1)mod\ 5}H^c_{4,(j+1)mod\ 5}$
$P_{3,(j+1)mod\ 5}$	$H^c_{2,(j+1)mod\ 5}H^c_{2,(j+2)mod\ 5}H^c_{3,(j+1)mod\ 5}$
$H_{3,(j+2)mod\ 5}$	$H^c_{2,(j+2)mod\ 5}H^c_{3,(j+2)mod\ 5}H^c_{4,(j+1)mod\ 5}$
$P_{3,(j+2)mod\ 5}$	$H^c_{2,(j+2)mod\ 5}H^c_{3,(j+2)mod\ 5}$
$H_{3,(j+3)mod\ 5}$	$H^c_{4,(j+3)mod\ 5}$
$P_{3,(j+3)mod\ 5}$	$H^c_{3,(j+4)mod\ 5}H^c_{4,(j+3)mod\ 5}$
$H_{3,(j+4)mod\ 5}$	$H^c_{4,(j+3)mod\ 5}H^c_{3,(j+4)mod\ 5}P^c_{4,(j+4)mod\ 5}$
$P_{3,(j+4)mod\ 5}$	$P^c_{3,(j+4)mod\ 5}$
$P_{4,j}$	$P^c_{4,j}$
$H_{4,j}$	$H^c_{3,(j+1)mod\ 5}P^c_{4,j}$
$P_{4,(j+1)mod\ 5}$	$H^c_{3,(j+1)mod\ 5}H^c_{4,(j+1)mod\ 5}H^c_{5,(j+1)mod\ 5}$
$H_{4,(j+1)mod\ 5}$	$H^c_{3,(j+1)mod\ 5}H^c_{3,(j+2)mod\ 5}H^c_{4,(j+1)mod\ 5}H^c_{5,(j+1)mod\ 5}$
$P_{4,(j+2)mod\ 5}$	$H^c_{3,(j+2)mod\ 5}H^c_{4,(j+1)mod\ 5}H^c_{5,(j+1)mod\ 5}$
$H_{4,(j+2)mod\ 5}$	$H^c_{3,(j+2)mod\ 5}P^c_{4,(j+3)mod\ 5}$
$P_{4,(j+4)mod\ 5}$	$P^c_{4,(j+4)mod\ 5}$
$P_{4,(j+3)mod\ 5}$	$P^c_{4,(j+3)mod\ 5}$
$H_{4,(j+3)mod\ 5}$	$P^c_{3,(j+3)mod\ 5}P^c_{4,(j+3)mod\ 5}P^c_{4,(j+4)mod\ 5}$
$H_{4,(j+4)mod\ 5}$	$P^c_{3,(j+4)mod\ 5}P^c_{4,(j+4)mod\ 5}P^c_{4,j}$
$H_{5,j}$	$H^*_{5,(j+1)mod\ 5}P^c_{4,j}$
$H_{5,(j+1)mod\ 5}$	$H^c_{4,(j+1)mod\ 5}H^c_{5,(j+1)mod\ 5}$
$H_{5,(j+2)mod\ 5}$	$H^c_{5,(j+1)mod\ 5}P^c_{4,(j+3)mod\ 5}$
$H_{5,(j+3)mod\ 5}$	$P^c_{4,(j+3)mod\ 5}P^c_{4,(j+4)mod\ 5}$

$H_{5,(j+4)mod\ 5}$	$P^c_{4,(j+4)mod\ 5}P^c_{4,j}$
$P_{6,1}$	$H^c_{5,(j+1)mod\ 5}$

Table 7.8: Node versus Color assignment for Class IV-A ICS

Color	Node
$H^c_{2,(j+1)mod\ 5}$	$H_{2,j}$
$H^c_{2,(j+2)mod\ 5}$	$H_{2,(j+3)mod\ 5}$
$H^c_{3,(j+4)mod\ 5}$	$H_{2,(j+4)mod\ 5}$
$H^c_{4,(j+4)mod\ 5}$	$H_{3,j}$
$H^c_{4,(j+3)mod\ 5}$	$H_{3,(j+3)mod\ 5}$
$H^c_{5,(j+1)mod\ 5}$	$P_{6,1}$
$H^c_{2,(j+1)mod\ 5}H^c_{2,(j+2)mod\ 5}$	$P_{1,1}$
$H^c_{2,(j+1)mod\ 5}H^c_{3,(j+1)mod\ 5}$	$P_{3,j}$
$H^c_{2,(j+2)mod\ 5}H^c_{3,(j+2)mod\ 5}$	$P_{3,(j+2)mod\ 5}$
$H^c_{3,(j+4)mod\ 5}H^c_{4,(j+3)mod\ 5}$	$P_{3,(j+3)mod\ 5}$
$H^c_{4,(j+1)mod\ 5}H^c_{5,(j+1)mod\ 5}$	$H_{5,(j+1)mod\ 5}$
$H^c_{2,(j+1)mod\ 5}H^c_{3,(j+1)mod\ 5}H^c_{2,(j+2)mod\ 5}$	$H_{2,(j+1)mod\ 5}$
$H^c_{2,(j+1)mod\ 5}H^c_{2,(j+2)mod\ 5}H^c_{3,(j+2)mod\ 5}$	$H_{2,(j+2)mod\ 5}$
$H^c_{3,(j+1)mod\ 5}H^c_{2,(j+1)mod\ 5}H^c_{4,(j+1)mod\ 5}$	$H_{3,(j+1)mod\ 5}$
$H^c_{2,(j+2)mod\ 5}H^c_{3,(j+2)mod\ 5}H^c_{4,(j+1)mod\ 5}$	$H_{3,(j+2)mod\ 5}$
$H^c_{3,(j+1)mod\ 5}H^c_{4,(j+1)mod\ 5}H^c_{5,(j+1)mod\ 5}$	$P_{4,(j+1)mod\ 5}$
$H^c_{3,(j+2)mod\ 5}H^c_{4,(j+1)mod\ 5}H^c_{5,(j+1)mod\ 5}$	$P_{4,(j+2)mod\ 5}$
$H^c_{3,(j+1)mod\ 5}H^c_{3,(j+2)mod\ 5}H^c_{4,(j+1)mod\ 5}$	$H_{4,(j+1)mod\ 5}$
$H^c_{2,(j+1)mod\ 5}H^c_{2,(j+2)mod\ 5}H^c_{3,(j+1)mod\ 5}H^c_{3,(j+2)mod\ 5}$	$P_{3,(j+1)mod\ 5}$
$H^c_{3,(j+1)mod\ 5}P^c_{4,j}$	$H_{4,j}$
$H^c_{3,(j+2)mod\ 5}P^c_{4,(j+3)mod\ 5}$	$H_{4,(j+2)mod\ 5}$

$H^c_{5,(j+1)mod\ 5}P^c_{4,j}$	$H_{5,j}$
$H^c_{5,(j+1)mod\ 5}P^c_{4,(j+3)mod\ 5}$	$H_{5,(j+2)mod\ 5}$
$H^c_{4,(j+3)mod\ 5}H^c_{3,(j+4)mod\ 5}P^c_{4,(j+4)mod\ 5}$	$H_{3,(j+4)mod\ 5}$
$P^c_{3,(j+4)mod\ 5}$	$P_{3,(j+4)mod\ 5}$
$P^c_{4,j}$	$P_{4,j}$
$P^c_{4,(j+3)mod\ 5}$	$P_{4,(j+3)mod\ 5}$
$P^c_{4,(j+4)mod\ 5}$	$P_{4,(j+4)mod\ 5}$
$P^c_{4,(j+3)mod\ 5}P^c_{4,(j+4)mod\ 5}$	$H_{5,(j+3)mod\ 5}$
$P^c_{4,(j+4)mod\ 5}P^c_{4,j}$	$H_{5,(j+4)mod\ 5}$
$P^c_{3,(j+3)mod\ 5}P^c_{4,(j+3)mod\ 5}P^c_{4,(j+4)mod\ 5}$	$H_{4,(j+3)mod\ 5}$
$P^c_{3,(j+4)mod\ 5}P^c_{4,(j+4)mod\ 5}P^c_{4,j}$	$H_{4,(j+4)mod\ 5}$

Table 7.9: Color versus Node assignment for Class IV-A ICS

This concludes proof of Theorem 2.

□

7.4 Lower Bound of MICS of SBG

In Fig. 7.2, we have provided a *layered* representation of the SBG, where 32 nodes of the SBG is placed in six layers, indicated by L1 through L6. The layers L1 through L3 constitute the *top half* of the SBG and the layers L4 through L6 constitute the *bottom half*. As the two halves are symmetric, similar argument can be applied to both of them.

Lemma 7. *A MICS must select at least four nodes from each half. In other words, at least four distinct colors need to be injected in each half.*

Proof. We provide arguments for the top half of the SBG, and we first show that *at* *least* three distinct colors are necessary to ensure that each node in top half receives a distinct color (either through injection, seepage or combination of the two). Let's consider layers L1 and L2. No matter which nodes in bottom half are injected with colors, these colors will not seep into the nodes in L1 and L2 (colors seep only to adjacent nodes), coloring in bottom half nodes will not affect the colors associated with nodes in L1 or L2. Since L1 and L2 have six nodes, six distinct colors need to be associated with them. It can be easily verified that in the SBG, in order to ensure distinct colors to each one of the six nodes in L1 and L2, at least three colors must be injected to three nodes in the top half. Clearly at least three nodes must be selected from each half so that nodes in L1, L2, L5 and L6 receive distinct colors. With injection of three colors, up to $2^3 - 1 = 7$ colors (excluding an empty combination) can be generated.

Next we show that three colors are not sufficient to color L1 and L2. WLOG, we use alphabets $\{A, B, C\}$ to represent three colors. As mentioned above, seven distinct colors can be generated with these three colors (three primary and four composite), $\{A, B, C, AB, AC, BC, ABC\}$. Simple counting shows that each alphabet (color) appears exactly 4 times. Suppose there is a proper injection using A, B, C that ensures all nodes in L1 and L2 received distinct colors. Since seven distinct colors can be generated with three primary colors, and L1 and L2 has only six nodes, it implies that one of the seven colors (primary or composite) is not used while coloring the nodes of L1 and L2. This implies that at least one of the alphabets A, B, C is appearing three times instead of four in the alphabet strings (representing colors) associated with the nodes of L1 and L2. WLOG *we assume that color A is appearing three times.* There are four possible locations for injection of color A in the top half of the SBG,

1. A is injected on L1, i.e., at $P_{1,1}$. A would then appear at all nodes in L1 and L2, making its appearance six times, contradicting the assumption.

2. A is injected on L2, i.e., one of $H_{2,i} (1 \leq i \leq 5)$ nodes. Thus, A appears four times (three nodes in L2 and one node in L1) contradicting the assumption.

3. A is injected on one of the hexagonal nodes L3, i.e., one of the $H_{3,i}, 1 \leq i \leq 5$. Since $H_{3,i}$ has only one neighbor in on L1 and L2 ($H_{2,i}$), in this case A will appear only on one node in L2, making its appearance one time, contradicting the assumption.

4. A is injected on one of the pentagonal nodes L3, i.e., one of the $P_{3,i}, 1 \leq i \leq 5$. Since $P_{3,i}$ has only two neighbors in on L1 and L2 ($H_{2,(i-1) mod\ 5}$ and $H_{2,i}$), in this case A will appear only on two nodes in L2, making its appearance two times, contradicting the assumption.

As there is no location for injection of A, we can conclude that three colors are inadequate to ensure that all nodes in L1 and L2 receive a unique color. Similar arguments can be made for coloring of nodes in L5 and L6. Therefore the lower bound of MICS for the SBG must be at least $4 + 4 = 8$. □

Lemma 8. *MICS of the SBG is at least nine.*

Proof. In the GCS problem, each node is assigned a color, which may be a primary or a composite color. A primary color is indicated by one alphabet and a composite color by a *string* of alphabets. The number of alphabets that appear in a string determines the *length* of that string. We establish the lemma by providing arguments based on the sum of the length of strings associated with all 32 nodes of the SBG. We will refer to the sum of the length of strings associated with each one of the 32 nodes of the SBG as “total string length”.

We use the term “valid injection” to imply an injection of colors to the nodes that ensures that all 32 nodes of the SBG receive a distinct color. Suppose, there exists more than one valid injection using eight colors. Among the set of all valid injections, we consider the one whose total string length is minimum. The lower bound of the total string length for a valid injection with eight colors is 56. This is true, as with eight injected colors, at most eight nodes of the SBG can have associated strings of length one, and the remaining 24 nodes must have strings of length at least two. Thus the lower bound on the string length must be $8 \times 1 + 24 \times 2 = 56$. The upper bound on the total string length with injection of at most eight colors is also 56. This is true for the following reason. If a color is injected on a hexagonal node, then it will appear seven times (six neighbors and the node itself). Similarly for a pentagonal node, the color will appear six times. Therefore, the upper bound of total string length is $7 \times 8 = 56$. It may be noted that the total string length is 56 if and only if all colors are injected on hexagonal nodes. However, it is impossible to achieve a valid injection by injecting eight colors only on hexagonal nodes. Consider the top half of SBG. In order to color nodes on L1, at least one color, say A, must be injected on one node on L2. WLOG, we assume that A is injected on $H_{2,i}, 1 \leq i \leq 5$. We consider two scenarios:

1. No other color is injected at the nodes on L2. In this case, the other colors are injected at three hexagonal nodes on L3. Because of injection of A at $H_{2,i}$, after seepage, all six adjacent nodes, $P_{1,1}, H_{2,(i-1) mod\ 5}, H_{2,(i+1) mod\ 5}$, $H_{3,(i-1) mod\ 5}, P_{3,i}$, $H_{3,i}$, will have color A. In order to ensure that all these nodes have distinct colors, colors must be injected on $H_{3,(i-1) mod\ 5}, H_{3,i}$, $H_{3,(i+1) mod\ 5}$. However, in such an injection, the nodes $H_{2,(i+2) mod\ 5}$ and $H_{2,(i+3) mod\ 5}$ will not receive any color, making such an injection invalid.

2. One or more colors are injected at the nodes on L2. Suppose a different color B

is injected at a node different from $H_{2,i}$. Due to the SBG topology, no matter which node on L2 is injected with B, one node on L2 and the node on L1 must have color AB after seepage. In order to ensure distinct colors on these two nodes, a third color C must be injected on another node. After injection of C, one of the two nodes that had the color AB before injection of C, will have the color AB and the other will have ABC. However, if one node has a string of length three, the lower bound of the total string length can longer be 56. It has to be at least 57, thus exceeding the upper bound (56), that is possible with injection of at most eight colors.

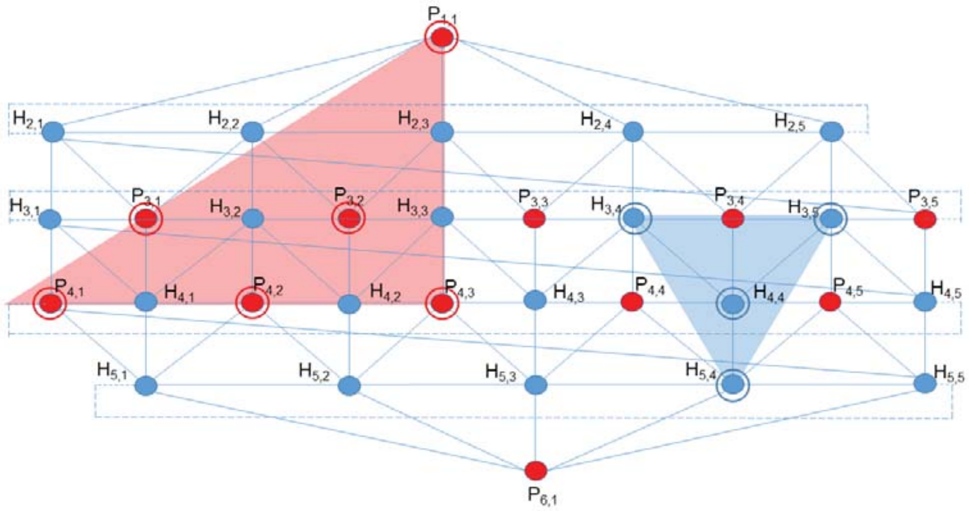
□

Theorem 9. *The lower bound of MICS of the SBG is at least nine i.e., eight colors are insufficient to ensure that all nodes of the SBG receive a distinct color.*

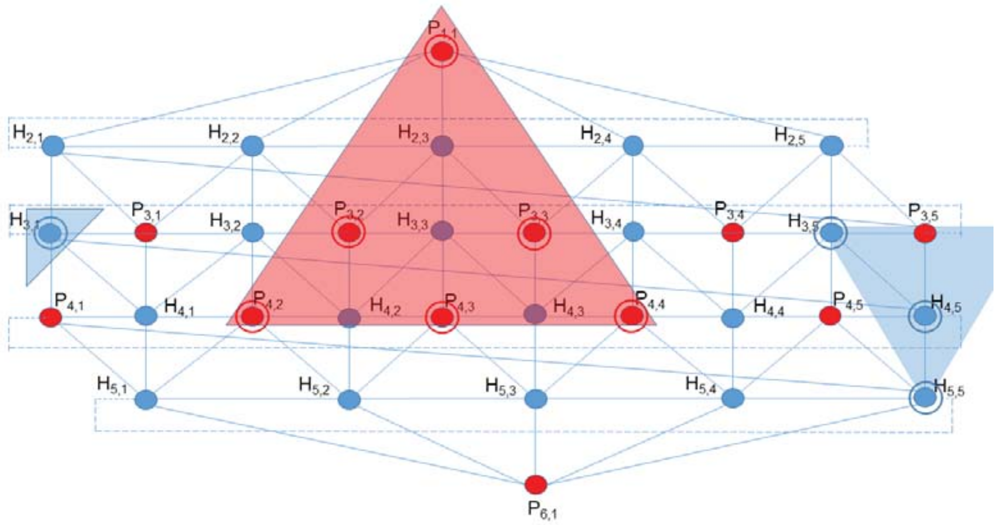
Proof. Follows from Lemmas 1 and 2. □

7.5 Conclusion

We have studied an event monitoring problem with satellites as sensors and a soccer ball as a model of the planet Earth. We have provided upper and lower bound of the MICS problem where the difference between the bounds is just one, implying that our solution is close to being optimal.

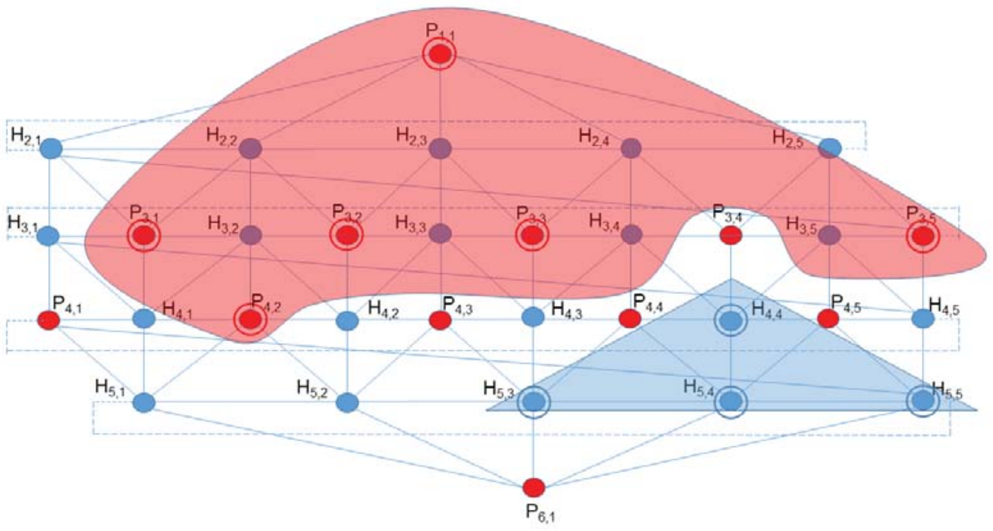


(a) Class IIA Motif Assignment I

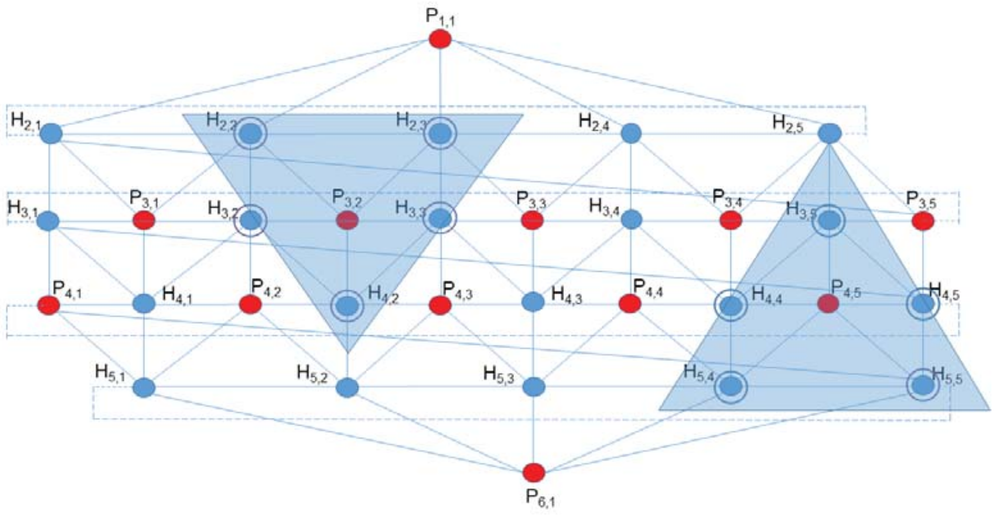


(b) Class IIA Motif Assignment II (Assignment I shifted one position to the right)

Figure 7.3: Examples of Color Assignments using Motif IIA



(a) Class IIIA Motif Assignment



(b) Class IV Motif Assignment

Figure 7.4: Examples of Color Assignments using Motifs IIIA and IV

Chapter 8

CONCLUSION AND FUTURE WORK

Firstly, in this dissertation, an overview of the existing literature on wireless relay node placement was presented. The dissertation also pointed out that few research considered budget limitation while it is very common in real world and practical applications. Hence, the dissertation presented three new problems: (i)*Budget Constrained Relay node Placement for Maximizing the size of the Largest Connected Component* (BCRP-MLCC) problem, (ii) *Budget Constrained Relay node Placement for Maximizing the size of the Smallest Connected Component* (BCRP-MSCC) problem and (iii) *Budget Constrained Relay node Placement for Minimizing the Number of Connected Components* (BCRP-MNCC) problem. The dissertation gave an $\frac{1}{10} - approximation$ algorithm for (i) and showed inapproximatebility for (ii) and (iii). Linearized MILP was included and experiments were conducted on both approximation algorithm and MILP.

Secondly, this dissertation considers a generalized $d - degree$ minimum arrangement problem in graph embedding. An asymptotically optimal $(d-1)$-ary tree tree construction is given for uniform graphs and two greedy approaches are presented for general graphs. Experiments and simulations are implemented to compare heuristic result again optimal solution obtained by ILP.

Thirdly, this dissertation introduced an new concept of "candidate trees" in Elastic Optical Network. An computational scheme for counting the number of different candidate trees on complete graphs is given. In detail, two closed form expressions that provide the number of Steiner Trees of a complete graph with two or three terminal nodes respectively were presented. For the cases, where the number of terminal nodes is at least four, an efficient $O(np)$ algorithm that computes the *number* of Steiner Trees of a complete graph with n nodes and p terminal nodes was given. Moreover, a candidate tree generation algorithm was also attached.

Fourthly, he dissertation studied path computation problem in FiWi networks. Based on the new framework focusing on "heaviest edge", an polynomial algorithm which solved single path computation was given. Moreover, mult-ipath problem was also considered. NP-compleness proof as well approximation analysis were presented in this dissertation. The efficacy and guaranteed performance of approximation algorithm was evaluated using the power and communication network infrastructure data of ARPANET network.

Moreover, except for analysis of wireless sensor network and FiWi network, the dissertation also studied scheduling problems in cyber-physical systems. Specifically, the *Optimal Schedule Construction Problem* (OSCP) in a RFID system was formulated and solution techniques were proposed. Though a heuristic is proposed in this dissertation, there is still some improvement that can be done in in extension to this work:

Last but not the least, this dissertation uses concept of identification code to better assistant event and disaster supervision. By modeling earth as a soccer ball surface, this dissertation provides schemes to place at most 10 satellites to cover the whole planet. Theoretical lower bound is also established by analysis.

Apart from the preliminary work discussed in this dissertation, some of the future work that can be performed in extension to this work is outlined below: *Gender assignment problem in wireless network*

The frequency-division duplex (FDD) nodes use two separate frequency bands (separated by a guard band) for transmission and reception, thus enabling the full-duplex

(FD) communication. On the other hand, the use of directional FDD nodes in multihop wireless network offers the advantages of larger transmission range, better link reliability, and spatial reuse, resulting in a much higher throughput and superior interference mitigation. However, the multihop FDD communication partitions the nodes in two classes (or genders) wherein the nodes of the same class (or gender) in a neighborhood cannot communicate with each other. This can seriously impact the availability of neighboring nodes for communication, and lead to disconnected nodes (or regions) in the network. In a simple way, we can think this scenario as assigning gender to each node and two nodes can communicate with each other if and only if they are within transmission range and have different genders. So, as a new direction of research in this dissertation, I propose the following two variations of gender assignment problems in wireless networks: (i) minimize the number of edges disabled due to gender assignment, (ii) maximize the number of nodes such after assigning gender there is no conflict. The goal of the study would be to analyze the computational complexity for both variations of the problem. If the variations can be solved in polynomial time, the goal would be to provide polynomial optimal algorithms. Otherwise, the goal would be to prove that the variations are NP-complete, and thereafter to provide approximation algorithms or proof of hardness of approximation.

Upper and Lower Bounds of Choice Number for Successful Channel Assignment in Cellular Networks

A cellular network is often modeled as a graph and the channel assignment problem is formulated as a coloring problem of the graph. Cellular graphs are used to model hexagonal cell structure of a cellular network. Assuming a $2-band$ buffering system where the interference does not extend beyond two cells away from the call originating cell, we study a version of the channel assignment problem in cellular graphs that been studied only minimally. In this version, each node has a fixed set of frequency channels where only a subset of which may be available at a given time for communication (as other channels may be busy). Assuming that only a subset of frequency channels are available for communication at each node, we try to determine the size of the smallest set of free channels in a node that will guarantee that each node of the cellular graph can be assigned a channel (from its own set of free channels) that will be interference free in a two band buffering system. The mathematical abstraction of this problem is known as the Choice Number computation problem and is closely related to List Coloring problem in Graph Theory. In previous research, we establish lower and upper bound -$(8, 10)$, on the choice number of a $2 - band$ buffering system. One may notice that, the bound is not tight. Thus it is of our interest to have a better understanding on cellular network and achieve a tight bound, i.e., the exact choice number for $2 - band$ buffering system.

REFERENCES

[1] Abhishek Kashyap, M. S. a., Samir Khuller b, "Relay placement for fault tolerance in wireless networks in higher dimensions", Computational Geometry **44**, 206–215 (2011).

[2] Auger, D., "Minimal identifying codes in trees and planar graphs with large girth", European Journal of Combinatorics **31**, 5, 1372 – 1384, URL `http://www.sciencedirect.com/science/article/pii/S0195669809002339` (2010).

[3] Aurzada, F., M. Lvesque, M. Maier and M. Reisslein, "Fiwi access networks based on next-generation pon and gigabit-class wlan technologies: A capacity and delay analysis", IEEE/ACM Transactions on Networking **22**, 4, 1176–1189 (2014).

[4] Camino, J.-T., C. Artigues, L. Houssin and S. Mourgues, "Linearization of euclidean norm dependent inequalities applied to multibeam satellites design", (2016).

[5] Chang, J.-Y. and Y.-W. Chen, "A cluster-based relay station deployment scheme for multi-hop relay networks", Communications and Networks, Journal of **17**, 1, 84–92 (2015).

[6] Charikar, M., M. T. Hajiaghayi, H. J. Karloff and S. Rao, "$l^2{}_2$ spreading metrics for vertex ordering problems", in "Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22-26, 2006", pp. 1018–1027 (2006), URL `http://dl.acm.org/citation.cfm?id=1109557.1109670`.

[7] Charikar, M., K. Makarychev and Y. Makarychev, "A divide and conquer algorithm for d-dimensional arrangement", in "Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007", pp. 541–546 (2007), URL `http://dl.acm.org/citation.cfm?id=1283383.1283441`.

[8] Charon, I., O. Hudry and A. Lobstein, "Identifying and locating-dominating codes: Np-completeness results for directed graphs", IEEE Transactions on Information Theory **48**, 8, 2192–2200 (2002).

[9] Charon, I., O. Hudry and A. Lobstein, "Minimizing the size of an identifying or locating-dominating code in a graph is np-hard", Theoretical Computer Science **290**, 3, 2109 – 2120, URL `http://www.sciencedirect.com/science/article/pii/S0304397502005364` (2003).

[10] Chen, D., D.-Z. Du, X.-D. Hu, G.-H. Lin, L. Wang and G. Xue, "Approximations for steiner trees with minimum number of steiner points", Journal of Global Optimization **18**, 1, 17–33 (2000).

[11] Cheng, X., D.-Z. Du, L. Wang and B. Xu, "Relay sensor placement in wireless sensor networks", Wireless Networks **14**, 3, 347–355 (2008).

[12] Cormen, T. H., C. E. Leiserson, R. L. Rivest and C. Stein, *Introduction to algorithms*, vol. 6 (MIT press Cambridge, 2001).

[13] Dourado, M. C., R. A. de Oliveira and F. Protti, "Generating all the steiner trees and computing steiner intervals for a fixed number of terminal-s", Electronic Notes in Discrete Mathematics **35**, Supplement C, 323 – 328, URL `http://www.sciencedirect.com/science/article/pii/S1571065309002066`, lAGOS'09 V Latin-American Algorithms, Graphs and Optimization Symposium (2009).

[14] Eom, J. and T.-J. Lee, "RFID reader anti-collision algorithm using a server and mobile readers based on conflict-free multiple access", in "Proc. of IEEE Int. Performance, Computing, and Communications Conference (IPCCC)", (2008).

[15] Eppstein, D., "Finding the k smallest spanning trees", in "Proceedings of the Second Scandinavian Workshop on Algorithm Theory", SWAT '90, pp. 38–47 (Springer-Verlag New York, Inc., New York, NY, USA, 1990), URL `http://dl.acm.org/citation.cfm?id=88723.88736`.

[16] Even, G., J. Naor, S. Rao and B. Schieber, "Divide-and-conquer approximation algorithms via spreading metrics", J. ACM **47**, 4, 585–616, URL `http://doi.acm.org/10.1145/347476.347478` (2000).

[17] Feige, U. and J. R. Lee, "An improved approximation ratio for the minimum linear arrangement problem", Inf. Process. Lett. **101**, 1, 26–29, URL `http://dx.doi.org/10.1016/j.ipl.2006.07.009` (2007).

[18] Garg, N., "Saving an epsilon: a 2-approximation for the k-mst problem in graph-s", in "Proceedings of the thirty-seventh annual ACM symposium on Theory of computing", pp. 396–402 (ACM, 2005).

[19] Ghazisaidi, N. and M. Maier, "Fiber-wireless (fiwi) access networks: Challenges and opportunities", IEEE Network **25**, 1, 36–42 (2011).

[20] Ghazisaidi, N., M. Maier and C. M. Assi, "Fiber-wireless (fiwi) access networks: A survey", IEEE Communications Magazine **47**, 2, 160–167 (2009).

[21] Hamida, E. B., "Wsnet - an event-driven simulator for large scale wireless sensor networks", URL `http://wsnet.gforge.inria.fr/` (2007).

[22] Hamouda, E., N. Mitton and D. Simplot-Ryl, "Reader Anti-Collision in Dense RFID Networks With Mobile Tags", in "Proc. of IEEE International Conference on RFID-Technologies and Applications (RFID-TA", (Barcelona, Spain, 2011).

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

[24] J. Ho, S. S., D.W. Engels, “Hiq: a hierarchical q-learning algorithm to solve the reader collision problem”, in “Proc. of Int. Symposium on Applications and the Internet Workshops (SAINT)”, (USA, 2006).

[25] J. S. Provan, M. K. C., “Counting problems associated with steiner trees in graphs”, SIAM Journal on Discrete Mathematics **10**, 3, 436–446 (1997).

[26] Jain, S. and S. Das, “Collision avoidance in a dense RFID network”, in “Proc. of ACM International Workshop on Wireless Network Testbeds, Experimentation and CHaracterization (WiNTECH)”, (USA, 2006).

[27] Jonathan L. Bredin, M. T. H., Erik D. Demaine and D. Rus, “Deploying sensor networks with guaranteed fault tolerance”, TRANSACTIONS ON NETWORKING **18**, 1, 216–228 (2010).

[28] Joshi, G. P., K. M. A. Mamun and S. W. Kim, “A reader anti-collision mac protocol for dense reader RFID system”, in “Proc. of Int. Conference on Communications and Mobile Computing (CMC)”, (USA, 2009).

[29] Karpovsky, M. G., K. Chakrabarty and L. B. Levitin, “On a new class of codes for identifying vertices in graphs”, IEEE Transactions on Information Theory **44**, 2, 599–611 (1998).

[30] Khuller, S. and B. Raghavachari, “Improved approximation algorithms for uniform connectivity problems”, in “Proceedings of the twenty-seventh annual ACM symposium on Theory of computing”, pp. 1–10 (ACM, 1995).

[31] Kotschick, D., “The topology and combinatorics of soccer balls: When mathematicians think about soccer balls, the number of possible designs quickly multiplies”, American Scientist **94**, 4, 350–357, URL `http://www.jstor.org/stable/27858804` (2006).

[32] Krause, A., C. Guestrin, A. Gupta and J. Kleinberg, “Near-optimal sensor placements: Maximizing information while minimizing communication cost”, in “Proceedings of the 5th international conference on Information processing in sensor networks”, pp. 2–10 (ACM, 2006).

[33] Laifenfeld, M. and A. Trachtenberg, “Identifying codes and covering problems”, IEEE Transactions on Information Theory **54**, 9, 3929–3950 (2008).

[34] Laifenfeld, M., A. Trachtenberg, R. Cohen and D. Starobinski, “Joint monitoring and routing in wireless sensor networks using robust identifying codes”, in “2007 Fourth International Conference on Broadband Communications, Networks and Systems (BROADNETS ’07)”, pp. 197–206 (2007).

[35] Li, C.-L., T. S. McCormick and D. Simich-Levi, “The complexity of finding two disjoint paths with min-max objective function”, Discrete Appl. Math. **26**, 1, 105–115, URL `http://dx.doi.org/10.1016/0166-218X(90)90024-7` (1989).

[36] Li, J., L. L. Andrew, C. H. Foh, M. Zukerman and H.-H. Chen, "Connectivity, coverage and placement in wireless sensor networks", Sensors **9**, 10, 7664–7693 (2009).

[37] Li, P., C. Huang and Q. Liu, "Bcdp: Budget constrained and delay-bounded placement for hybrid roadside units in vehicular ad hoc networks", Sensors **14**, 12, 22564–22594 (2014).

[38] Li, S., J. Wang, C. Qiao and Y. Xu, "Mitigating packet reordering in fiwi networks", IEEE/OSA Journal of Optical Communications and Networking **3**, 2, 134–144 (2011).

[39] Lin, G.-H. and G. Xue, "Steiner tree problem with minimum number of steiner points and bounded edge-length", Information Processing Letters **69**, 2, 53–57 (1999).

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

[41] Lu, H.-C. and W. Liao, "Joint base station and relay station placement for ieee 802.16 j networks", in "Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE", pp. 1–5 (IEEE, 2009).

[42] M. Mosko, J. G.-L.-A., "Multipath routing in wireless mesh networks", in "IEEE Workshop on Wireless Mesh Networks", (2005).

[43] Mazumder, A., C. Zhou, A. Das and A. Sen, "Budget constrained relay node placement problem for maximal "connectedness"", in "IEEE International Conference for Military Communications (MILCOM)", (2016).

[44] Misra, S., S. D. Hong, G. Xue and J. Tang, "Constrained relay node placement in wireless sensor networks: Formulation and approximations", IEEE/ACM Transactions on Networking (TON) **18**, 2, 434–447 (2010).

[45] Pan, J., Y. T. Hou, L. Cai, Y. Shi and S. X. Shen, "Topology control for wireless sensor networks", in "Proceedings of the 9th Annual International Conference on Mobile Computing and Networking", MobiCom '03, pp. 286–299 (ACM, New York, NY, USA, 2003), URL `http://doi.acm.org/10.1145/938985.939015`.

[46] Peeters, M., "On coloring j-unit sphere graphs", Research Memorandum FEW 512, Tilburg University, School of Economics and Management (1991).

[47] Rao, S. and A. W. Richa, "New approximation techniques for some linear ordering problems", SIAM J. Comput. **34**, 2, 388–404, URL `http://dx.doi.org/10.1137/S0097539702413197` (2004).

[48] Ray, S., R. Ungrangsi, F. De Pellegrini, A. Trachtenberg and D. Starobinski, "Robust location detection in emergency sensor networks", pp. 1044 – 1053 vol.2 (2003).

[49] Riihijärvi, J., M. Petrova and P. Mähönen, “Frequency allocation for wlans using graph colouring techniques”, Ad Hoc & Sensor Wireless Networks **3**, 2-3, 121–139 (2007).

[50] Sen, A., B. Hao, B. H. Shen, L. Zhou and S. Ganguly, “On maximum available bandwidth through disjoint paths”, in “HPSR. 2005 Workshop on High Performance Switching and Routing, 2005.”, pp. 34–38 (2005).

[51] Senel, F. and M. Younis, “Relay node placement in structurally damaged wireless sensor networks via triangular steiner tree approximation”, Computer Communications **34**, 16, 1932–1941 (2011).

[52] Shalom, M., “On the interval chromatic number of proper interval graphs”, Discrete Mathematics **338**, 11, 1907–1916 (2015).

[53] Suomela, J., “Approximability of identifying codes and locatingcdominating codes”, Information Processing Letters **103**, 1, 28 – 33, URL `http://www.sciencedirect.com/science/article/pii/S0020019007000385` (2007).

[54] Suurballe, J. W., “Disjoint paths in a network”, Networks **4**, 2, 125–145, URL `http://dx.doi.org/10.1002/net.3230040204` (1974).

[55] Tang, S., C. Wang, X.-Y. Li and C. Jiang, “Reader activation scheduling in multi-reader rfid systems: a study of general case”, in “Proc. of IEEE International Parallel & Distributed Processing Symposium (IPDPS)”, pp. 1147–1155 (IEEE, 2011).

[56] ufer, H. P., “Neuer beweis eines satzes uber permutationen”, pp. 742–744 (1918).

[57] Waldrop, J., D. W. Engles and S. E. Sarma, “Colorwave : A mac for RFID reader networks”, Proc. of IEEE Conference on Wireless Communication and Networking (WCNC) (2003).

[58] Waldrop, J., D. W. Engles and S. E. Sarma, “Colorwave: An anticollision algorithm for the reader collision problem”, in “Proc. of IEEE International Conference on Communications (ICC)”, (2003).

[59] Walkowiak, K., R. Gocie, M. Klinkowski and M. Woniak, “Optimization of multicast traffic in elastic optical networks with distance-adaptive transmission”, IEEE Communications Letters **18**, 12, 2117–2120 (2014).

[60] Wang, J., K. Wu, S. Li and C. Qiao, “Performance modeling and analysis of multi-path routing in integrated fiber-wireless networks”, in “2010 Proceedings IEEE INFOCOM”, pp. 1–5 (2010).

[61] Xiao, Y., C. Hadjicostis and K. Thulasiraman, “The d-identifying codes problem for vertex identification in graphs: Probabilistic analysis and an approximation algorithm”, in “Computing and Combinatorics”, edited by D. Z. Chen and D. T. Lee, pp. 284–298 (Springer Berlin Heidelberg, Berlin, Heidelberg, 2006).

[62] Zhang, W., G. Xue and S. Misra, "Fault-tolerant relay node placement in wireless sensor networks: Problems and algorithms", in "INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE", pp. 1649–1657 (IEEE, 2007).

[63] Zheng, Z. and J. Wang, "A study of network throughput gain in optical-wireless (fiwi) networks subject to peer-to-peer communications", in "2009 IEEE International Conference on Communications", pp. 1–6 (2009).

[64] Zheng, Z., J. Wang and X. Wang, "Onu placement in fiber-wireless (fiwi) networks considering peer-to-peer communications", in "GLOBECOM 2009 - 2009 IEEE Global Telecommunications Conference", pp. 1–7 (2009).