ABSTRACT

Title of Document: ROBUST DESIGN OF WIRELESS NETWORKS

Abhishek Kashyap, Ph.D., 2006

Directed By: Professor Mark Shayman

Electrical and Computer Engineering

We consider the problem of robust topology control, routing and power control in wireless networks. We consider two aspects of robustness: topology control for robustness against device and link failures; routing and power control for robustness against traffic variations. The first problem is more specific to wireless sensor networks.

Sensors typically use wireless transmitters to communicate with each other. However, sensors may be located in a way that they cannot even form a connected network (e.g, due to failures of some sensors, or loss of battery power). Using power control to induce a connected topology may not be feasible as the sensors may be placed in clusters far apart. We consider the problem of adding the smallest number of relay nodes so that the induced communication graph is k-connected. We consider both edge and vertex connectivity. The problem is NP-hard. We develop approximation algorithms that find close to optimal solutions. We consider extension to higher dimensions, and provide approximation guarantees for the algorithms. In addition, our methods extend with the same approximation guarantees to a generalization when the locations of relays are required to avoid certain polygonal

obstacles. We also consider extension to networks with non-uniform transmission range, and provide approximation algorithms.

The second problem we consider is of joint routing and transmission power assignment in multi-hop wireless networks with unknown traffic. We assume the traffic matrix, which specifies the traffic load between every source-destination pair in the network, is unknown, but always lies inside a polytope. Our goal is to find a fixed routing and power assignment that minimizes the maximum total transmission power in the network over all traffic matrices in a given polytope. In our approach, we do not need to monitor and update paths to adapt to traffic variations. We formulate this problem as a non-convex semi-infinite programming problem. We propose an efficient algorithm that computes a routing and power assignment that is schedulable for all traffic matrices in the given polytope. We perform extensive simulations to show that the proposed algorithm performs close to algorithms that adaptively optimize their solution to the traffic variations.

ROBUST DESIGN OF WIRELESS NETWORKS

by

Abhishek Kashyap

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2006

Advisory Committee:
Professor Mark Shayman, Chair/Advisor
Professor Alexander Barg
Professor Samir Khuller
Professor Richard La
Professor Aravind Srinivasan

Dedication

To my parents for their love, encouragement and support

Acknowledgments

I would like to thank my advisor, Prof. Mark Shayman, and Prof. Samir Khuller for their guidance and help through my PhD. Their guidance and insight has helped me accomplish the work being published in this dissertation, along with other research projects. I would also like to thank them for other professional advice, which has been really helpful in shaping my PhD.

I would like to thank Prof. Richard La and Prof. Bobby Bhattacharjee for their guidance on the last chapter of this dissertation, and on other research projects.

I would like to thank Prof. Mark Shayman, Prof. Alexander Barg, Prof. Richard La, Prof. Samir Khuller and Prof. Aravind Srinivasan for serving on the dissertation committee.

I would like to thank my colleagues, Vahid Tabatabaee, Anuj Rawat, Mehdi Kalantari, Tuna Güven and Fangting Sun for their suggestions and help with the research projects I worked on during my PhD.

I would like to thank my friends at University of Maryland, specifically Amrit Bandyopadhyay, Ravi Tandon, Rahul Ratan, Manish Shukla, Vishal Khandelwal, Rajat Ahuja; for helpful discussions on my research problems, and related logistics.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Robustness is a desired feature of wireless networks due to their dynamic characteristics. We consider the problem of robust design of wireless networks. We work with two aspects of robustness: robustness against wireless node and link failures, which is critical for networks in harsh conditions; and robustness against traffic variations in the network, which is important for providing desired Quality of Service to the end-users. The first aspect is more relevant to wireless sensor networks [1] since they may exist in harsh network conditions. The second problem is applicable to wireless mesh networks, where a high volume of changing traffic is expected, and the application is to provide Internet access [2], that requires good Quality of Service. We introduce the problems in the following sections.

1.1 Robustness Against Failures

A wireless sensor network is a group of sensor nodes with sensing, processing and communication capabilities, deployed to achieve a certain objective (Akyildiz et al. [1]). Typical applications of sensor networks are habitat monitoring, environmental monitoring, object tracking etc. Sensor networks may exist in harsh network conditions, thus the network must be designed so that failure of some sensor nodes or some communication links between them does not disrupt the network.

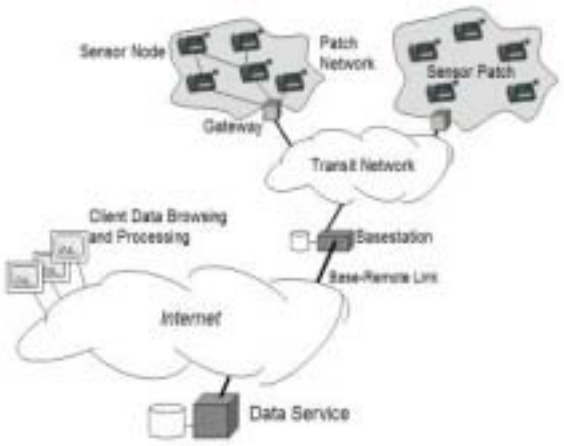


Figure 1.1: Example application - Storm Petrel monitoring

We consider the problem of forming a fault-tolerant network topology.

Figure 1.1 shows an example sensor network (Mainwaring et al. [55]) our work targets. The network was deployed for monitoring of birds (Storm Petrel), with patches of sensor nodes deployed around and inside burrows, each patch having a gateway (or multiple so it is not a single point of failure). The gateways are connected to a base station through a transit network. A user (sink) can get the sensed data by connecting to the base station (static sink) through the Internet or by physically going to a gateway and connecting directly to it (mobile sink). In this application, our work concerns design of a fault-tolerant transit network between the gateways. We will call the nodes on which a fault-tolerant topology is desired as terminal nodes, which are the gateways in this example. This example demonstrates the typical application of our work: the design of fault-tolerant sensor network topology, where sensors are deployed in distant areas of interest. Sensor nodes have very limited energy. Thus, they transmit at low power levels, and have

a limited transmission range. We assume a fixed transmission range for each sensor node (gateway in the application of Figure 1.1). It may not be feasible to construct even a connected topology among the sensor nodes due to their short transmission range and deployment in far-away regions. We propose the use of additional relay nodes, whose position we can control, to achieve the desired level of fault-tolerance. The relay nodes are cheaper than sensor nodes as they do not have any sensing capabilities.

We define fault-tolerance as the existence of multiple internally vertex-disjoint (or edge-disjoint) paths between each pair of terminal nodes. If k vertex (edge) disjoint paths exist between each pair of nodes, the network is said to be k-vertex (edge) connected. A k-vertex (edge) connected graph has the property that the failure of any set of $(k-1)$ nodes (edges) cannot disconnect the network. We also consider the problem where fault-tolerance is desired between both terminal and relay nodes. We call this objective as **full k-connectivity**, and the objective of achieving k-connectivity among terminal nodes as **partial k-connectivity**.

We consider two models: one where we assume all relay and sensor nodes have the same communication capabilities (homogeneous network); and the other where we assume the communication capabilities of sensor nodes are different, and the relay nodes have the same communication capabilities, which can be different from those of sensor nodes (heterogeneous network).

The contributions of this work are the following:

1. We provide algorithms for placement of relays to achieve partial k-vertex and

k-edge connectivity among terminals in a homogeneous network.

2. We prove the partial k-vertex connectivity algorithm to be a $c(3\lceil k/2\rceil(\lceil k/2\rceil + 1)-1)$-approximation[1] with respect to the optimal, for terminals distributed in the Euclidean plane. Here, c is the approximation ratio of the best algorithm for computing a minimum-weight k-vertex connected spanning subgraph of a k-vertex connected graph.

3. We prove the partial k-edge connectivity algorithm to be a $10\lceil k/2\rceil$-approximation with respect to the optimal, for terminals distributed in the Euclidean plane.

4. We prove the full k-vertex connectivity algorithm of Bredin et al. [14] is a $3c(3\lceil k/2\rceil(\lceil k/2\rceil + 1) - 1)k$-approximation.

5. We provide a full k-edge connectivity algorithm, and prove the algorithm to be a $30\lceil k/2\rceil^2$-approximation.

6. We consider extensions to other metric spaces[2]. We prove the partial k-vertex connectivity algorithm is a $2M$-approximation for $k = 2$, and the full k-vertex

[1]Approximation ratio of an algorithm is defined as the worst case ratio of the performance of the algorithm to the performance of an optimal algorithm. An algorithm with approximation ratio α is called an α-approximation.

[2]We consider metric spaces which have a known MST number. MST number is defined as the maximum node degree in a minimum-degree Minimum Spanning Tree (MST) spanning points from the space. MST number for the Euclidean plane is 5 (Monma and Suri [58]), three-dimensional Euclidean space is 12, and rectilinear plane (two-dimensional space with metric defined by L_1 norm) is 4 (Robins and Salowe [65]).

connectivity algorithm is a $12M$-approximation for $k = 2$, where M is the MST number of the metric space.

7. We prove the partial k-edge connectivity algorithm is a $2M\lceil k/2\rceil$-approximation and full k-edge connectivity algorithm is a $6M\lceil k/2\rceil$-approximation for terminals distributed in other metric spaces.

8. We prove the approximation analysis is tight for partial 2-edge connectivity, i.e., the lower bound on the worst case performance relative to the optimal is equal to the upper bound provided by our analysis.

9. We extend our algorithms to the generalization where the relays cannot be placed in certain polygonal regions (obstacles) and show the same approximation ratios hold for this generalization as well, for terminals distributed in the Euclidean plane.

10. We consider heterogeneous networks with non-uniform transmission range of terminal and relay nodes. We provide algorithms for partial and full k-vertex and k-edge connectivity. We prove the partial vertex connectivity algorithm to be a $2(5+11\lceil\log_{\sqrt{3}}(\min\{\alpha,\gamma\})\rceil+5\mathcal{I}_{\lceil\log_{\sqrt{3}}\gamma\rceil>\lceil\log_{\sqrt{3}}\alpha\rceil})$-approximation for $k = 2$, and full vertex connectivity algorithm to be a $12(5+11\lceil\log_{\sqrt{3}}(\min\{\alpha,\gamma\})\rceil+5\mathcal{I}_{\lceil\log_{\sqrt{3}}\gamma\rceil>\lceil\log_{\sqrt{3}}\alpha\rceil})$-approximation for $k = 2$, for terminals distributed in Euclidean plane. Here, $\mathcal{I}_x$ is the indicator function, which is 1 if condition x is true, else it is 0. We prove the edge connectivity algorithms to be $2(5+11\lceil\log_{\sqrt{3}}(\min\{\alpha,\gamma\})\rceil+5\mathcal{I}_{\lceil\log_{\sqrt{3}}\gamma\rceil>\lceil\log_{\sqrt{3}}\alpha\rceil})\lceil k/2\rceil$-approximation for par-

tial k- connectivity, and $6(5+11\lceil\log_{\sqrt{3}}(\min\{\alpha,\gamma\})\rceil+5\mathcal{I}_{\lceil\log_{\sqrt{3}}\gamma\rceil>\lceil\log_{\sqrt{3}}\alpha\rceil})\lceil k/2\rceil^2$ - approximation for full k-connectivity.

11. We consider networks in the three dimensional space using Euclidean metrics. We prove the partial vertex connectivity algorithm to be a $2(12 + 37\lceil\log_{1.5}(\min\{\alpha,\gamma\})\rceil + 13\mathcal{I}_{\lceil\log_{1.5}\gamma\rceil>\lceil\log_{1.5}\alpha\rceil})$-approximation for $k = 2$, and full vertex connectivity algorithm to be a $12(12 + 37\lceil\log_{1.5}(\min\{\alpha,\gamma\})\rceil + 13\mathcal{I}_{\lceil\log_{1.5}\gamma\rceil>\lceil\log_{1.5}\alpha\rceil})$-approximation for $k = 2$, for terminals distributed in the Euclidean plane. We prove the edge connectivity algorithms to be $2(12 + 37\lceil\log_{1.5}(\min\{\alpha,\gamma\})\rceil + 13\mathcal{I}_{\lceil\log_{1.5}\gamma\rceil>\lceil\log_{1.5}\alpha\rceil})\lceil k/2\rceil$-approximation for partial k-connectivity, and $6(12 + 37\lceil\log_{1.5}(\min\{\alpha,\gamma\})\rceil + 13\mathcal{I}_{\lceil\log_{1.5}\gamma\rceil>\lceil\log_{1.5}\alpha\rceil})\lceil k/2\rceil^2$ - approximation for full k-connectivity.

We also consider application to networks with node mobility. In such networks, we can use the proposed algorithms to add relays to establish a fault-tolerant topology between network nodes. But as the network nodes move, the desired fault-tolerance might be lost due to their limited transmission range. We propose algorithms to reconfigure the topology by moving the existing relay nodes a minimum amount and adding the minimum number of additional relay nodes.

The main application of these algorithms is in battlefield communication networks, which are networks of mobile nodes, communicating with each other using wireless links. The nodes in the battlefield refer to soldiers, army vehicles, UAVs, robots, etc. The network may also be a team of nodes engaged together to perform a large scale reconnaissance mission. The capability of a device to communicate

with all nodes (using single or multiple hops) is very critical for the collaborative missions to succeed. In these applications, the nodes are in hostile conditions, and the nodes or communication links between nodes may fail. Thus, it is critical for the communication network between the nodes to be connected even after a few failures. Since the nodes are mobile, it is necessary to re-configure the communication topology as it changes substantially.

We assume the network nodes move at a slow time scale. Thus, once their locations have changed significantly, we would like to re-establish the desired topology using minimum number of relays. Since some relays already exist in the network (used in the topology on previous terminal locations), the secondary objective is to move the existing relay nodes a minimum distance to the new relay positions so that the topology is constructed quickly. We propose algorithms to achieve the above stated objectives.

1.2 Robustness Against Traffic Variation

A wireless mesh network [2] is a multi-hop wireless network, consisting of both static and mobile devices. In these networks, it is crucial to control and minimize the transmission power because, (1) nodes in wireless networks generally have limited power and bandwidth, and (2) transmissions can interfere with other wireless devices in the area and degrade the performance.

Joint routing, power control and scheduling schemes are effective mechanisms to control power consumption in wireless multi-hop networks. The common ap-

proach is to *assume* the traffic load in the network is known or can be measured, and to provide provable optimal solutions (Cruz and Santhanam [22], Lin and Cruz [51], Bhatia and Kodialam [12], Neely et al. [60]). However, in practice, since traffic demands in the network are hard to measure, and they change continuously, deployment of these algorithms result in sub-optimal routing that has to be updated continuously. Frequent path updates can cause further problems, e.g., communication overhead which consumes power and bandwidth, temporal route instability due to asynchronous information exchange, and disruption in traffic flow.

In this dissertation, we propose a joint *fixed* routing and power control algorithm that does not assume exact knowledge of the traffic matrix, but gives a performance guarantee for an ensemble of traffic matrices that are inside a polytope. Since we fix the routing we do not need to update the paths once they are established in the network. We also ensure that the obtained transmission rates satisfy the sufficient conditions introduced by Kodialam and Nandagopal [43] for schedulability. Therefore, using local traffic information, a distributed online algorithm can be used for scheduling. Similar to Cruz and Santhanam [22], Lin and Cruz [51], Bhatia and Kodialam [12], our goal is to minimize the total transmission power in the network. However, in our framework, since we are not dealing with a single traffic matrix, we minimize the maximum total transmission power over all traffic matrices in the traffic region.

For the traffic demand region, we assume some restrictions on the network traffic do exist. The first set of restrictions is on the total traffic originating either at a source node, or sinking at a destination node. These constraints are called

hose model constraints (Duffield et al. [25]), and usually represent the capacity constraints of the ingress/egress nodes. The second set of constraints is on the individual commodities (pipe model constraints), and can normally be inferred from the traffic history, Service Level Agreements (SLAs), etc. These two sets of constraints constitute a polytope in which the traffic matrices lie. Our objective is to minimize the worst case total transmission power over the traffic matrix polytope, subject to the constraint that the computed routing and link rates (inferred from link transmission power) are schedulable for all traffic matrices in the polytope.

There has been recent work in wireline networks for computing a static routing with good performance for a traffic matrix polyhedron, Azar et al. [6], Applegate and Cohen [3], Kodialam et al. [42], Tabatabaee et al. [70]. The problem in wireline networks is different since the link capacities are fixed. In wireless networks, the link transmission power specifies the achievable capacity (rate) of each link.

The dependence of link capacities on transmitting powers in wireless networks gives an additional degree of freedom (power control) that does not exist in wireline networks. Therefore, the optimal routing solutions proposed for wireline systems are generally not directly applicable for wireless systems. In fact, the non-linear relation between capacity and transmission power in wireless systems results in a non-convex optimization problem, which is fundamentally harder than the linear optimization problem of wireline systems. There has been recent work on oblivious routing in wireless networks, Li et al. [49]. However, the authors assume fixed link capacity, neglecting the non-linear relation between capacity and transmission power of wireless networks. To summarize, the contributions of our work are as follows:

1. We formulate the problem of joint routing and power control (without the knowledge of exact end-to-end traffic matrix) as a non-convex semi-infinite programming problem.

2. We propose an efficient algorithm that yields a *fixed* set of paths and transmission powers, and is schedulable for all traffic matrices within a specified polytope.

3. We propose centralized and distributed extensions of the basic algorithm that trade off complexity for better performance. The extended algorithms still use fixed routing, but adapt the transmission powers to the traffic matrix variations in the network.

4. We perform extensive simulations to show that the performance of our static algorithm is comparable to an existing more complex dynamic algorithm, Bhatia and Kodialam [12], that continuously optimizes both routes and power assignment for the given traffic matrix.

1.3 Organization

The dissertation is organized as follows: Chapter 2 discusses the related work. Chapter 3 presents the algorithms and analysis for partial k-vertex (edge) connectivity. Chapter 4 discusses generalizations of the k-connectivity problems. The generalizations considered are: providing full k-vertex/edge connectivity; algorithms and analysis for providing k-connectivity in presence of obstacles in the network; analysis

for terminal nodes distributed in other metric spaces; and algorithms and analysis for providing k-connectivity in heterogeneous networks. Chapter 5 presents algorithms for movement of relay nodes for topology reconfiguration. Chapter 6 presents algorithms for joint routing and power control in wireless networks with unknown traffic. Chapter 7 concludes the dissertation, and presents some future research directions.

Chapter 2

Related Work

We first present the related work on topology control for design of fault-tolerant wireless networks. We later present the related work on traffic-oblivious joint routing and power control.

2.1 Topology Control for Connectivity

There has been a considerable amount of work on relay placement in sensor networks for topology control. Corke et al. [19, 20] present algorithms for deployment and connectivity repair of sensor networks using helicopters and flying robots. They use flying robots to automatically deploy sensor nodes at target locations, and repair connectivity by deploying additional nodes when the network becomes disconnected. Bredin et al. [14] propose $O(k^4)$-approximation algorithms for achieving full k-vertex connectivity using minimum number of relays. Their analysis is for nodes distributed in the Euclidean plane. Their algorithms use a similar approach as ours and their analysis for partial k-vertex connectivity gives an approximation ratio of $O(k^3)$, while our analysis proves the approximation ratio to be $O(k^2)$. We prove the approximation ratio of their full vertex connectivity algorithm to be $O(k^3)$. We also provide an $O(k)$-approximation algorithm for partial k-edge connectivity, and $O(k^2)$-approximation algorithm for full k-edge connectivity. We also analyze our

algorithms for terminal nodes distributed in other metric spaces.

Han et al. [33] consider the problem of placing minimum number of relay nodes for achieving partial and full k-vertex connectivity in a heterogeneous wireless sensor network. They assume the transmission range of the sensor nodes is in the range $[T_{min}, T_{max}]$, and the transmission range of relay nodes is T_{relay}. They provide algorithms for two models: only uni-directional links are allowed between nodes; only bi-directional links are allowed between nodes. In this second model, a link exists between two nodes only if both are within each other's transmission range. They provide algorithms with approximation ratio of $c((8\beta^2 + \frac{1}{4})k^2 + \frac{3}{2}k + 2)$ for partial k-vertex connectivity for the bi-directional link model. Here, c is the approximation ratio of the best algorithm for computing a k-vertex connected subgraph of a graph, and $\beta = \lceil T_{relay}/T_{min} \rceil$. The authors also provide approximation analysis for nodes distributed in metric spaces with dimension d. For homogeneous networks, with the transmission range being the same for all nodes, their algorithms are identical to our algorithms for k-vertex connectivity. For homogeneous networks, their approximation ratios are of the same order as our analysis provides, but the constants are very large. As an example, for partial 2-vertex connectivity, $c = 2$, and their analysis yields an approximation ratio of 76, while our analysis yields a ratio of 10. We also analyze their full and partial vertex connectivity algorithms for 2-vertex connectivity in heterogeneous networks in the Euclidean plane and three dimensional Euclidean space, and our analysis is much tighter. For partial 2-vertex connectivity, our analysis gives an approximation ratio of $O(\log(\min\{\alpha, \gamma\}))$, where $\alpha = T_{max}/T_{min}, \gamma = T_{relay}/T_{min}$, while their algorithm gives an approximation ratio

of $O(\beta^2)$. As an example, for the Euclidean plane, if $T_{max} = 2T_{min}$ and $T_{relay} = T_{max}$, the approximation ratio for partial 2-vertex connectivity proved by their analysis is 274, while our analysis yields an approximation ratio of 37.76. We also propose and analyze algorithms for partial and full k-edge connectivity.

Hao et al. [34] consider the problem of placing the minimum number of backbone nodes (relays) among a set of candidate locations such that each sensor node has paths to at least two backbone nodes, and the backbone nodes have at least two vertex-disjoint paths between them. They provide an approximation algorithm having an $O(D \log n)$ approximation ratio, where D depends on the diameter of the network and n is the number of sensor nodes in the network. Liu et al. [52] consider the problem of placing relays in a network of sensor nodes so that they cover the sensor nodes (i.e., each sensor node is connected to at least one relay) and the network is 2-vertex connected. They provide a $(6 + \epsilon)$-approximation algorithm for connectivity and two approximation algorithms for 2-vertex connectivity with ratios $(24 + \epsilon)$ and $(6/T + 12 + \epsilon)$, where T is the ratio of relays needed for connectivity to the number of sensor nodes. Tang et al. [71] consider the problem of placement of relay nodes to cover a set of sensor nodes such that each sensor node can reach two relay nodes and the topology is 2-vertex connected. They provide 6 and 4.5-approximation algorithms for the problem. They also consider the problem of covering the sensor nodes and providing a connected topology, and provide 8 and 4.5-approximation algorithms. Their problem is different from ours as they want the set of relays to be a dominating set among the sensor nodes, i.e., each sensor node should be directly connected to at least one relay node.

Recall that we defined terminal nodes as nodes on which a fault-tolerant topology is desired. The problem of constructing a connected network on terminal nodes using minimum number of relay nodes has been considered in Lin et al. [50], Măndoui and Zelikovsky [56], and Chen et al. [15]. Lin et al. [50] prove the problem to be NP-Hard, and propose an approximation algorithm for constructing a tree using relay nodes. They prove the algorithm to be a 5-approximation for nodes distributed in the Euclidean plane. The algorithm restricts the placement of relay nodes on lines joining pairs of terminal nodes. It then assigns a weight function to each pair of terminal nodes according to the number of relay nodes needed to connect them directly. They find a minimum spanning tree (MST) on this graph. Măndoui and Zelikovsky [56] and Chen et al. [15] prove the algorithm to be a 4-approximation, and the bound is proved to be tight. Măndoui and Zelikovsky [56] prove the approximation ratio to be $M-1$ for nodes distributed in metric spaces with MST number M. Chen et al. [15] also provide a 3-approximation algorithm for the problem. Cheng et al. [16] provide a 2.5-approximation randomized algorithm for placement of relay nodes to connect a given set of terminal nodes.

There has been work on probabilistic analysis of number of nodes required and their transmission range required for achieving k-connectivity among randomly distributed nodes. Bettstetter [11] studies the node degree properties of nodes distributed randomly in a network, and the connectivity and k-connectivity properties of the network. The author gives probabilities of having isolated nodes and approximates the probability of having a connected network to be equal to the probability of having no isolated node. Similar approximations are done for k-connectivity as

well. Li et. al. [48] prove that for sufficiently large number of nodes in the network, there is a critical power level after which the network is k-connected with a certain non-zero probability. None of the frameworks seems to extend to the case of adding additional nodes in the network for achieving connectivity, as achieving connectivity by increasing power level of all nodes is not similar in spirit to achieving connectivity by adding relay nodes. Also, if terminal nodes are not located close to each other, addition of relay nodes is a more practical solution to achieving desired connectivity levels.

Basu and Redi [9] consider the problem of moving the terminal nodes (robots in their setting) to bi-connect the topology on terminal nodes. They assume the position of all terminal nodes is controllable. Their objective is to minimize the total distance travelled by the terminal nodes. They provide optimal polynomial time solutions using linear programming for terminals distributed in a one-dimensional space, and provide heuristic algorithms for terminals distributed in a two-dimensional space. Shields et al. [68] consider the problem of placement of relay nodes to keep the topology on base stations in a wireless network connected. They assume the movement trajectory of base stations is given. They provide upper and lower bounds on the number of relays needed, and propose heuristic algorithms to place them.

Another area of recent work has been on power control algorithms for achieving connectivity and k-vertex connectivity in wireless networks. Ramanathan and Rosales-Hain [63] consider the problem of power control to achieve connectivity and bi-connectivity in multi-hop wireless networks. They consider the problem of minimizing the maximum power used in the network, and provide greedy opti-

mal algorithms for static networks. For mobile networks, they provide heuristics that adaptively adjust power to achieve connectivity in the network. Rodoplu and Meng [66] propose distributed power control algorithms for minimizing the total transmission energy used in the network while achieving connectivity. Wattenhofer et. al. [74] propose a distributed cone based topology control (CBTC) algorithm for power control at each node to achieve connectivity in the network. The algorithm is local at each node, and works by having each node increase its transmission power enough to have at least one neighbor in every cone of angle α around it. Wattenhofer and Zollinger [75] propose a topology control algorithm called XTC. XTC computes a connected topology without using any node location information; it uses link quality indicators to establish the topology using only two-hop information. Lloyd et al. [53] propose an 8-approximation algorithm for computing a minimum total power 2-connected topology among wireless nodes. Li et al. [47] propose localized algorithms for construction of a bounded-degree connected topology among wireless nodes with non-uniform transmission range.

Hajiaghayi et al. [32] propose approximation algorithms for achieving k-vertex connectivity in wireless networks. The objective is to minimize the sum of the maximum power levels of all nodes. The authors propose $O(k)$-approximation centralized algorithms for general graphs, and $O(k)$-approximation distributed algorithms for achieving k-vertex connectivity in geometric graphs, i.e., graphs where edge lengths satisfy triangle inequality. Li and Hou [46] propose algorithms for computing a k-vertex connected topology that minimizes the maximum node power in the network. They provide an optimal centralized algorithm, and a localized algorithm

that is proved to be optimal among all algorithms that use only local information. Thallner and Moser [73] propose clustering-based algorithms for localized computation of a k-vertex connected topology on wireless nodes. They recursively construct clusters of size k, such that total power used in the network is low. Bahramgiri et al. [7] extend the Cone Based Topology Control (CBTC) algorithm of Wattenhofer et al. [74] to provide k-vertex connectivity in two and three dimensional Euclidean spaces.

2.2 Traffic-Oblivious Joint Routing and Power Control

Distributed algorithms for transmission power assignment have been proposed for cellular networks, Zander [78], Foschini and Miljanic [28], Yates [77], where the data transfers are between mobile nodes and a base station. The objective is to transmit at a power such that the signal to interference and noise ratio is above the reception threshold for all transmitting links (in different cells) which may interfere with each other.

Based on these algorithms, Elbatt and Ehpremides [26] proposed an algorithm for joint scheduling and power control in ad-hoc networks. The authors provide a set of medium access rules that makes the power control problem similar to the problem of power control in cellular networks. They propose a greedy centralized scheme to construct sets of links which can transmit together and schedule each set in one slot at the start of each TDMA frame. Power control is done using the algorithms proposed for cellular networks. The authors assume that the routing is

given and do not consider routing optimization, which has a critical effect on the performance.

The problem of joint routing and scheduling of a given traffic matrix for fixed transmission power has been studied in Hajek and Sasaki [31], Wieselthier et al. [76], Kodialam and Nandagopal [43, 44]. Hajek and Sasaki [31] provide polynomial time algorithms to compute a minimum-length schedule and routing for traffic in a network. They use ellipsoid algorithm with a separation oracle, thus the running time, as the authors mention, is not practical. Wieselthier et al. [76] propose neural networks based heuristics that compute a single path per traffic demand, with a cost function that prefers a smaller length schedule. Kodialam and Nandagopal [43, 44] study the problem of schedulability of a traffic matrix according to the scheduling constraints, and propose an approximation algorithm to compute a routing and schedule for a given traffic matrix.

There has been recent work on the problem of joint routing, scheduling and power control for minimizing total transmission power for a given traffic demand. Cruz and Santhanam [22], Lin and Cruz [51] propose optimal algorithms for the problem, but the algorithms consider an exponential number of transmission scenarios. Thus, the algorithms take exponential time to run, and cannot be executed each time the traffic in the network changes. Bhatia and Kodialam [12] propose a polynomial-time 3-approximation algorithm for the problem. Their work is different from ours, since they use the exact end-to-end traffic information, whereas we assume that traffic demand lies inside a region.

Neely et al. [60] consider the problem of dynamic routing and power assign-

ment for wireless networks of power constrained nodes with time varying channels. The authors provide a region of traffic matrices a network can support and a centralized algorithm for computing an adaptive routing and power assignment in each TDMA slot that guarantees system stability for all traffic in that region. The authors prove that the algorithm has bounded delays for certain channel and traffic processes. In their algorithm they have to solve a non-convex optimization problem. Therefore, the authors also propose a distributed heuristic algorithm for solving the non-convex optimization problem. The problem we consider is finding a fixed solution that minimizes worst case total transmission power consumption for traffic matrices inside a polytope. The worst case occurs at a vertex of the polytope. However, the number of vertices is exponential, and there are infinite number of non-convex constraints. We propose algorithms that solve a few convex quadratic optimization problems (there are very fast algorithms for solving convex quadratic programs). We show via simulations that the algorithms have performance comparable to more complex algorithms that continuously optimize and update routing as the traffic matrix varies. Also, we compute routing and power assignment just once, rather than once every time slot.

There has been recent work in wireline networks for computing a static routing with good performance for a traffic matrix polyhedron. Oblivious routing (Azar et al. [6], Applegate and Cohen [3]) is a routing that minimizes maximum ratio of the maximum link load to the maximum link load of the optimal routing for any arbitrary traffic matrix in the polyhedron (bounded or unbounded)[1]. The problem

[1]Considering bounds on the polyhedron actually makes their problem more complicated.

of providing absolute bounds on the performance to all traffic in a polytope (as we do), and not relative bounds (as oblivious routing does), has been considered in Kodialam et al. [42], Tabatabaee et al. [70]. The problem in wireline networks is different since the link capacities are fixed. In wireless networks, the link transmission power specifies the achievable capacity (rate) of each link. The dependence of link capacities on transmitting powers in wireless networks gives an additional degree of freedom (power control) that does not exist in wireline networks. Therefore, the optimal routing solutions proposed for wireline systems are generally not directly applicable for wireless systems. In fact, the non-linear relation between capacity and transmission power in wireless systems results in a non-convex optimization problem, which is fundamentally harder than the linear optimization problem of wireline systems.

Li et al. [49] consider the problem of oblivious routing for wireless networks. Their objective is to minimize the maximum node energy consumption relative to the maximum per node energy consumption of the optimal policy for each traffic matrix in a polyhedron. They assume static link capacities in the network (thus uncontrollable transmission power at each link). Their model is very similar to a wireline network and does not reflect the relation between transmission power and link capacity in wireless networks. Also, the scheduling constraints they have in their formulation are on the optimal routing (that is used for normalizing in the objective function) for each traffic demand, but not on the oblivious routing that they compute. In fact, they consider all traffic matrices that are schedulable using some routing. But, their computed routing may not be schedulable for all those

traffic matrices, and thus the performance guarantees will not hold. As an example, let there be two traffic matrices $T1$ and $T2$ for which we need to compute single routing for given capacity (if each can be scheduled by some routing, that may be different). The algorithm checks for the existence of a routing that is schedulable for $T1$, and another routing that is schedulable for $T2$. However, the algorithm does not constrain its output routing to be schedulable for any of $T1$ and $T2$. Thus, the output routing might not be able to schedule any of them under the given capacity constraints, while a routing that can schedule both might exist. They provide linear programs similar to those of wireline oblivious routing for computing the optimal routing. Incorporating scheduling constraints in their output routing would make the problem non-convex, and thus harder to solve. In our formulation, we consider the relation between power and rate in wireless networks, and guarantee that the output of our algorithm is schedulable for all traffic matrices in the given traffic matrix polytope. Due to these additions, the problem becomes much more complicated. Another difference is that we minimize the total power value rather than relative value to the optimal for each traffic demand.

Chapter 3

Robustness Against Node and Link Failures

In this chapter, we consider the problem of designing a network that is robust to node and link failures. We first describe the network model and problem statement. We then present algorithms and analysis for network topology design that is tolerant against node and link failures. We present some simulation results to conclude the chapter.

3.1 Network Model and Problem Statement

We model the network as a graph $G = (V, E)$, where V is the set of sensor nodes, which we call terminal nodes, and E is the set of links between them. We assume each node has a limited transmission range, which we normalize to one. It is assumed that a node can connect to all nodes within its transmission range. A link $e = (x, y)$ belongs to E if nodes x and y are within unit distance of each other. The links can be either omnidirectional RF, directional RF or Free Space Optical (without obscuration).

Our objective is to construct a network topology that is k-vertex or k-edge connected on the terminals. A network is said to be k-vertex (edge) connected if k vertex (edge) disjoint paths exist between each pair of nodes. A k-vertex (edge) connected graph has the property that the failure of any set of $(k-1)$ nodes (edges)

does not disconnect the network.

Due to the limited transmission range of terminal nodes, it may not even be possible to form a connected topology. We assume we have relay nodes and we have control over their location. We place the relay nodes in the network so that the desired level of connectivity is achieved. We assume the relay nodes are identical to the terminal nodes in terms of their transmission range and type of links. The problem can be stated as follows:

Partial k-connectivity: Given a graph $G = (V, E)$, find the minimum number of relay nodes (denoted by set R) needed (and their locations) such that the set of nodes V is k-vertex (edge) connected ($k \geq 2$) in the resulting graph $G' = (V + R, E'), E \subseteq E'$. The objective is to construct a graph such that $\forall\, u, v \in V$, $\lambda(u, v) \geq k$; where $\lambda(u, v)$ is the number of internally vertex-disjoint (or edge-disjoint) paths between u and v in G'.

3.2 Vertex Connectivity

We first consider the problem of providing k-vertex connectivity. We follow the relay placement framework of the connectivity algorithm of [50]. To connect two terminal nodes outside each other's transmission range, the relay nodes are placed on the straight line connecting the two nodes. The algorithm proceeds by forming a complete graph G_c on the terminal nodes. Equation 3.1 gives the weight function used for the edges, where $|e|$ is the length of an edge. The weight represents the number of relay nodes required to form an edge. We do not allow the relay nodes

to have edges other than the ones required to form the edge they are placed on.

$$c_e = \lceil |e| \rceil - 1 \tag{3.1}$$

Next, we compute an approximate minimum cost spanning k-vertex connected subgraph (G'_c) of G_c. The problem of finding the minimum cost spanning k-vertex connected subgraph of a graph is NP-Hard (Garey and Johnson [30]). Thus, we use the 2-approximation algorithm of Khuller and Raghavachari [40] for $k = 2$, and the k-approximation algorithm of Kortsarz and Nutov [45] for $k > 2$. For $k \leq 7$, we can use the improved approximation algorithms proposed by Auletta et al. [5], and Dinitz and Nutov [24]. It is worth noting that the weight function of Equation 3.1 is not a metric as it does not satisfy the triangle inequality. In the resulting k-vertex connected subgraph, the relay nodes are placed to form the edges (of length greater than one) of the subgraph. We later prove that this algorithm has an approximation ratio of $O(k^2)$. The solution is then improved by removing some relays. The relays are allowed to form edges with all nodes in their transmission range and sequentially removed if k-vertex connectivity is preserved. We call this step the *sequential removal step*. Algorithm 3.2.1 describes the algorithm.

Algorithm 3.2.1 Relay placement for k-vertex connectivity

1: Construct a complete graph $G_c = (V, E_c)$ by adding an edge between each pair of vertices of graph G.

2: Weight the edges of the graph as follows. $|e|$ represents the length of edge e.

$$c_e = \lceil |e| \rceil - 1$$

3: Compute an approximate minimum cost spanning k-vertex connected subgraph from this graph G_c. Let the resulting graph be G'_c.

4: Place relay nodes (number equal to the weight of the edge) on the edges in G'_c with link costs greater than zero.

5: For all pairs of nodes (including the relay nodes) in G'_c within each other's transmission range, form an edge.

6: For the relay nodes sorted arbitrarily, do the following (starting at $i = 1$):

- Remove node i (and all adjacent edges).
- Check for k-vertex connectivity between the terminals.
- If the graph is k-vertex connected, repeat for $i = i + 1$, else put back the node i and corresponding edges, and repeat for $i = i + 1$.
- Stop when all relay nodes have been considered.

7: Output the resulting graph.

3.2.1 Proof of Approximation Ratio

We now analyze the algorithm to provide with approximation guarantees. We provide the analysis for terminals distributed in the Euclidean plane.

We start with some notation. Let $\mathcal{T}$ be the set of terminals, and $\mathcal{S}$ be the set of optimally placed Steiner nodes (relay nodes) needed to achieve k-vertex connectivity among the terminal nodes. Let s be the number of Steiner nodes needed when we place them optimally, i.e, $s = |\mathcal{S}|$. In the proof, we will call the relay nodes placed on straight lines between terminals (as in our algorithm) beads and the optimally placed relay nodes Steiner nodes.

As a recap of our algorithm, it forms a k-vertex connected network among the terminal nodes by placing additional links between them, and if two terminal nodes are more than unit distance apart, it adds beads (relay nodes) to form that link. When we add such a link of length l, it consists of $\lceil l \rceil - 1$ beads.

We first prove the following lemma, and then present the main result of this section.

Lemma 3.2.1 *A partial k-vertex connected network using the minimum number of beads contains at most $(3\lceil k/2 \rceil(\lceil k/2 \rceil + 1) - 1)s$ beads, where s is the minimum number of Steiner nodes needed.*

Proof: Let $G_0 = (V_0, E_0)$ be the optimal k-vertex connected network on terminals (having the minimum number of Steiner nodes).

We follow the procedure of Algorithm 3.2.2 to construct a graph with zero Steiner nodes that is k-vertex connected on the terminals, and has bounded number

of beads. We will prove that this network contains at most $(3\lceil k/2\rceil(\lceil k/2\rceil+1)-1)s$ beads.

Algorithm 3.2.2 starts by finding the connected components (SC_i) of Steiner nodes in the graph constructed on the Steiner nodes. It constructs a minimum-degree minimum spanning tree (MST) on Steiner nodes for each connected component, starting with any Steiner node in that component as the root. Let the trees be $ST_1, .., ST_m$. The algorithm then removes Steiner nodes of a connected component SC_j from G_{j-1} and adds beads between the terminals connected to those Steiner nodes to get G_j, which is also k-vertex connected on terminal nodes (Step 4). The process is repeated for all connected components, and the resulting graph has zero Steiner nodes.

Algorithm 3.2.2 Construction of k-vertex connected graph with beads

1: Define $G_S = (S, E_S)$, where $e = (u, v) \in E_S$ if $e \in E_0$.

2: Find all the connected components (SC_i) in G_S.

3: Construct a minimum-degree MST in each connected component, and call the trees $ST_1, .., ST_m$.

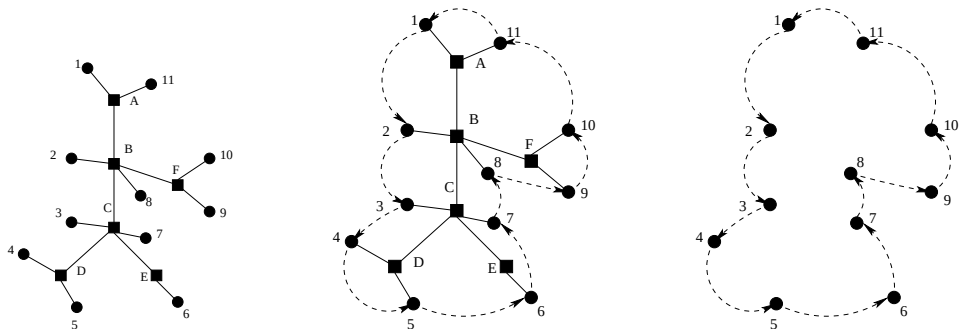
4: Set $j = 1$. While $j \leq m$:

1. Remove the Steiner nodes contained in ST_j from G_{j-1}.

2. Add beads between terminals to get the graph G_j, which is also k-vertex connected on the terminals. The procedure for adding beads and removing Steiner nodes is explained later.

3. Set $j = j + 1$.

5: Output the resulting graph G_m.



(a) Tree on Steiner and terminal nodes

(b) Depth first traversal and cycle creation

(c) Cycle after removal of Steiner nodes

Figure 3.1: Example elimination of Steiner nodes, and cycle construction

We first mention two useful properties that hold for each of the trees $ST_1, .., ST_m$, which will be used in proving the approximation bound:

Property 3.2.2 *The maximum degree of any Steiner node in the trees is bounded by five [58]. This property comes from the fact that the maximum degree of a node in a minimum-degree MST on nodes distributed in a Euclidean plane is bounded by five, and is called the MST number of the Euclidean plane.*

Property 3.2.3 *The angle between any pair of neighbors of a Steiner node in its tree ST_i is at least 60 degrees [15]. This can be seen from the fact that if the angle between two neighbors (x, y) at a Steiner node j were less than 60 degrees, the MST could be shortened by deleting an edge $(j - x$ or $j - y)$ and forming the edge $x - y$.*

Let us now explain the procedure to construct G_j from G_{j-1} by adding beads between the terminal nodes to construct a cycle (2-vertex connected graph), and removing the Steiner nodes. Consider the graph formed by the Steiner nodes in ST_j and the terminal nodes within the transmission range of these Steiner nodes. Denote

this graph by H_j. Algorithm 3.2.3 describes the algorithm for construction of a cycle between terminal nodes in H_j. We later add edges to this cycle to make it k-vertex connected on terminals in H_j. The algorithm works as follows: Start at the root of ST_j (assign any Steiner node in ST_j as the root; call the root st_1, dropping subscript j for simplicity). Connect to st_1 all terminal nodes within its transmission range, and mark them. Construct a tree T_j, with the vertex set as the Steiner nodes in ST_j and a leaf vertex corresponding to each marked terminal vertex. The edges are the edges of ST_j and the edges between each Steiner node and the marked terminal vertices connected to it. Start a Depth First Search (DFS) traversal of the tree T_j (rooted at st_1), starting with any child of st_1. The children (both Steiner nodes and terminals) of a node are traversed in an anti-clockwise manner, i.e., the next child to traverse is the first child encountered in an anti-clockwise sweep around the Steiner node, starting from the last child traversed. If no child of the Steiner node has been traversed yet, the child traversed is the one encountered in the sweep starting from the parent node. Whenever a new Steiner node st_j is encountered in the traversal, mark all unmarked terminal nodes in the Steiner node's transmission range and connect them to it in T_j. Figure 3.1(a) shows an example tree constructed using this procedure for $k = 3$. While doing the DFS traversal, add required number of beads to form a link between each terminal with the next terminal encountered in the DFS traversal. Complete the cycle by connecting the last added terminal to the first terminal encountered in the DFS traversal[1]. Figure 3.1(b) shows the

[1]Note that there will be at least two terminals connected to the Steiner nodes of ST_j. If there were only one terminal node, the Steiner nodes of ST_j could be deleted from the optimal Steiner

cycle constructed between the terminal nodes in the example, starting at terminal 1. Remove the Steiner nodes. The edges longer than unit length are added using the required number of beads. Figure 3.1(c) shows the constructed cycle after removal of Steiner nodes.

Algorithm 3.2.3 Removal of Steiner nodes and construction of a cycle in ST_j

1: Start at root st_1 of ST_j.

2: Connect to it all terminals within its transmission range, and mark them.

3: Construct a tree T_j, with the vertex set as the Steiner nodes in ST_j and a leaf vertex corresponding to each marked terminal vertex. The edges are the edges of ST_j and the edges between each Steiner node and the marked terminal vertices connected to it.

4: Do a Depth First Search (DFS) traversal of T_j rooted at st_1, starting with any child of st_1. For each node, traverse its children in an anti-clockwise manner.

5: Each time a new Steiner node st_i is encountered, connect it to all unmarked terminal vertices in its range, and mark them. Update T_j by adding these terminal vertices, and continue DFS traversal around st_i from the edge between st_i and its parent.

6: Connect the terminal vertices in order of their DFS traversal and complete the cycle between them.

7: Add beads to all added edges of length greater than one.

graph without affecting the connectivity. In case of two terminal nodes, adding one edge between the two makes it a complete graph, which suffices, as we show later.

We now prove that the cycle constructed in Algorithm 3.2.3 contains at most $5s_j$ beads for each Steiner component ST_j with s_j Steiner nodes. Lemma 3.2.4 states the result.

Lemma 3.2.4 *For each Steiner component SC_j with s_j Steiner nodes, the cycle constructed in Algorithm 3.2.3 contains at most $5s_j$ beads.*

Proof: We first define our charging scheme, i.e., how we charge the beads to the Steiner nodes in ST_j. We charge one bead to a Steiner node st_i each time one of the following ordered pair of edges is traversed:

- **Type I**: Steiner-st_i-Steiner.
- **Type II**: Steiner-st_i-Terminal, if the Euclidean distance between the end-nodes is greater than one.
- **Type III**: Terminal-st_i-Steiner, if the Euclidean distance between the end-nodes is greater than one.
- **Type IV**: Terminal-st_i-Terminal, if the Euclidean distance between the end-nodes is greater than one.

In a DFS traversal, each ordered pair of neighboring edges around a node is traversed once. Notice that for the beads charged by the pair of edges of Type II, III, IV, the angle between the edges at the Steiner node st_i is *greater* than 60 degrees. For the pair of Type I, the angle is *at least* 60 degrees (by Property 3.2.3). The pairs of Type I around a Steiner node is bounded by five (by Property 3.2.2). Also,

Table 3.1: Bead charging for example of Figure 3.1

Edge	(1,2)	(2,3)	(3,4)	(4,5)	(5,6)	(6,7)	(7,8)	(8,9)	(9,10)	(10,11)	(11,1)
Charged To	A,B	B,C	C,D	D	D,C,E	E,C	C,B	B,F	F	F,B,A	A
No. of Beads	1	0	0	1	1	1	0	0	0	1	1

as the traversal is anti-clockwise always, the angles subtended by all these pairs of edges at st_i are non-overlapping. Thus, it can be easily shown that the total pairs of such edges around the Steiner node st_i in the tree T_j is within five as the total angle traversed by these edges is bounded by 360 degrees.

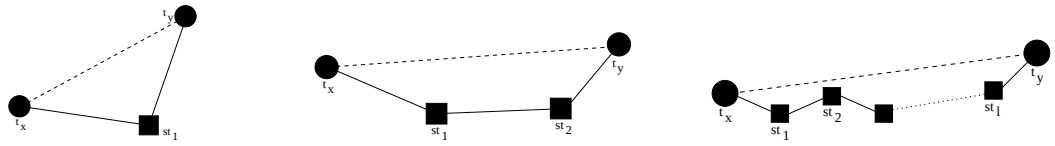
Table 3.1 gives an example charging for the example of Figure 3.1.

For each edge added to connect two terminal nodes in T_j, we claim that the number of beads required (given by Equation 3.1) can be charged to the Steiner nodes encountered in the DFS traversal between the two terminal nodes using our charging scheme. Let the two terminal nodes be t_x and t_y, and let there be $l > 0$ Steiner nodes on the DFS path between them. Renumber the Steiner nodes on the DFS path as $st_1, .., st_l$. We consider the following cases and prove that the charging scheme charges the required number of beads to the Steiner nodes:

- **Case 1**: $l = 1$: This case is depicted in Figure 3.2(a). If both the terminals are connected to the same Steiner node st_1, a bead is needed only if they are more than distance one apart. In that case, the pair of edges $t_x - st_1 - t_y$ is of Type IV and thus the Steiner node st_1 can be charged for the bead required.

- **Case 2**: $l = 2$: This case is depicted in Figure 3.2(b). Let the two Steiner nodes in the DFS path be st_1 and st_2, with st_1 being the parent of st_2 in the tree, i.e., the first encounter of st_1 in the DFS traversal is before st_2. Let t_x be connected to st_1 and t_y to st_2. Since st_1 is the parent of st_2, we connected all unmarked neighboring terminal nodes to st_1 first. Thus, t_y is more than distance one apart from st_1. If two beads are needed between t_x and t_y, the distance between t_x and st_2 is more than one and between st_1 and t_y is more than one. Thus the pairs of edges $t_x - st_1 - st_2$ and $st_1 - st_2 - t_y$ are Type III and II for Steiner nodes st_1 and st_2 respectively. Thus, one bead can be charged to each Steiner node. If we need one bead between t_x and t_y, that can be charged to st_2 as the pair of edges $st_1 - st_2 - t_y$ is always Type II for st_2. The explanation for the case where st_2 is the parent of st_1 is similar.

- **Case 3**: $l > 2$: This case is depicted in Figure 3.2(c). The total number of beads required is upper bounded by the number of Steiner nodes on any path between t_x and t_y. One bead can be charged to each of the nodes $st_2, .., st_{l-1}$ as the pair of edges involving them in the path are of Type I. If the distance between t_x and st_2 is less than one, the path can be modified by connecting t_x directly to st_2, and there is a path with $l - 1$ Steiner nodes, and thus st_1 can be removed. If it is greater than one, then a bead can be charged to st_1 as the pair of edges $t_x - st_1 - st_2$ is of Type III. Similarly, st_l can either be removed from the path between t_x and t_y, or it can be charged because of the pair of edges $st_{l-1} - st_l - t_y$ being Type II. Thus, there is a path of $l - 2$, $l - 1$ or



(a) One Steiner node, $l = 1$

(b) Two Steiner nodes, $l = 2$

(c) More than two Steiner nodes, $l > 2$

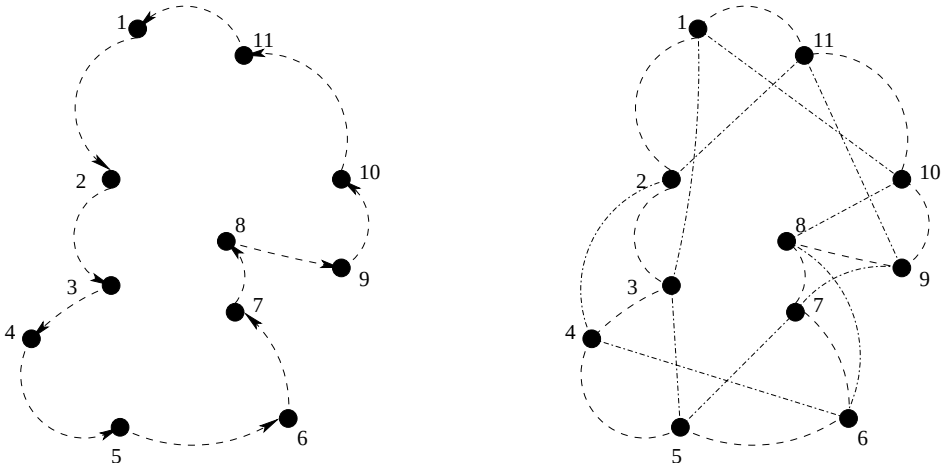
Figure 3.2: DFS paths of different lengths

l Steiner nodes between t_x and t_y (which is the upper bound for the number of beads required), and there are enough Steiner nodes that can be charged once.

We have shown that the charging scheme charges the required number of beads to the Steiner nodes, and each Steiner node is charged maximum of five times. □

We now add edges to the cycle constructed in Algorithm 3.2.3 to make the cycle k-vertex connected, for each Steiner component SC_j. We form a Harary graph, Harary [35], by connecting each terminal to preceding and successive $\lceil k/2 \rceil$ vertices on the cycle. This graph is k-vertex connected if $n \geq k + 1$, Harary [35]. We form a complete graph if there are less than $k + 1$ terminals. The edges longer than unit length are added using the required number of beads. Figures 3.3(a) and 3.3(b) show the cycle and corresponding Harary graph constructed on terminal nodes, for $k = 3$. The graph at the end of execution of Algorithm 3.2.3 along with Harary graph construction on Steiner component SC_j is G_j. The final k-vertex connected graph is represented by G_m.

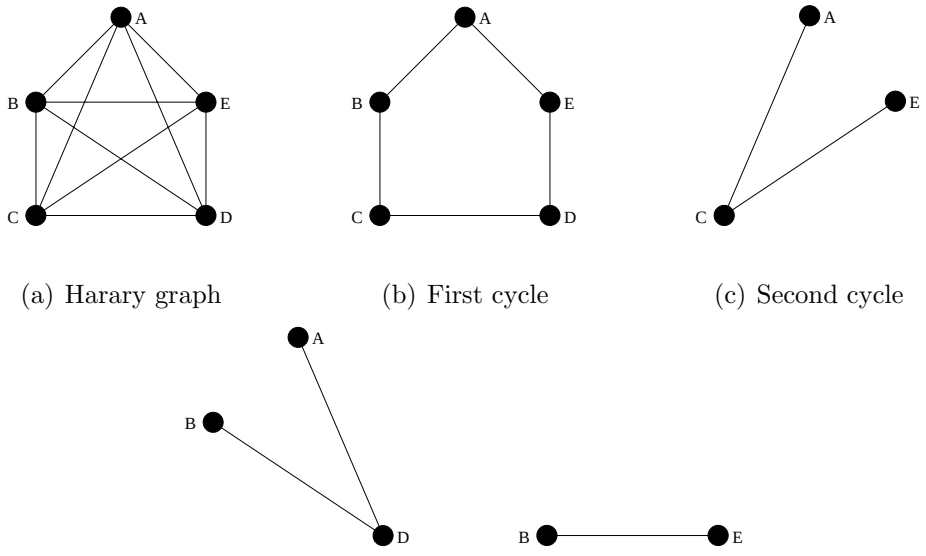
We now prove that the graph G_m constructed using the procedure described above is k-vertex connected on the terminals. Lemma 3.2.5 states the desired result.



(a) Cycle after removal of Steiner nodes

(b) Constructed Harary graph

Figure 3.3: Example construction of Harary graph for $k = 3$



(a) Harary graph

(b) First cycle

(c) Second cycle

(d) Third cycle

(e) Fourth cycle

Figure 3.4: Example decomposition of a Harary graph for $k = 3$

Lemma 3.2.5 *The graph G_m is k-vertex connected on terminals.*

Proof: The proof is based on mathematical induction and is similar to the proof of k-vertex connectivity in Bredin et al. [14].

G_0 is the optimal Steiner graph, that is k-vertex connected on the terminals. Let G_{i-1} be k-vertex connected on the terminals. Thus, removal of any set C of $k-1$ vertices does not disconnect the terminals in G_{i-1}. We prove by contradiction that all terminals are connected in $G_i - C$ as well. Let u and v be the two terminals which are disconnected in $G_i - C$. All terminal pairs (u, v) have a path in $G_{i-1} - C$. If the path does not use more than one terminal connected to component SC_i, u and v are connected in $G_i - C$ as well. If the path uses at least two terminal vertices connected to SC_i (u_1, v_1 being the first and last terminals connected to SC_i on the path), that path exists as well if there are at least $k+1$ terminals connected to SC_i, since we form a Harary graph (that is k-vertex connected) between all terminals connected to SC_i. If there are less than $k+1$ terminals (which will be u_1, v_1), a direct edge exists between them (since we formed a complete graph in that case) and thus a path exists between u and v in $G_i - C$. Thus, G_i is k-vertex connected on terminals. Therefore, by induction, G_m is k-vertex connected on terminals. □

We now prove the upper bound on number of beads in G'_m with respect to the optimal number of Steiner nodes, s. Let the number of terminal nodes in H_j (connected to the Steiner component in consideration) be N. We consider the case $N \geq k+1$, so that we can construct the Harary graph[2]. We decompose the

[2]Else, we construct a complete graph, which is a subset of the set of edges in the Harary graph (since $N < k+1$). Thus, the analysis for Harary graph is an upper bound for this case.

constructed Harary graph into complete and incomplete cycles (we call all of them cycles), and use Lemma 3.2.4 to compute its cost. Let us explain it with an example of Figure 3.4. Figure 3.4(a) shows the Harary graph constructed for $k = 3$, with each terminal connecting to preceding and successive two nodes on the cycle. We decompose the graph into multiple cycles as follows:

- **Type I cycle**: First cycle is the cycle formed between all the terminals, constructed using Algorithm 3.2.3.

- **Type II cycles**: We now consider the edges needed to connect nodes with preceding and successive nodes i hops away (number of edges between the nodes in the Type I cycle) on the Type I cycle. We start with any terminal node (node A in the example of Figure 3.4), and form a complete or incomplete cycle by starting with the node and traversing edges that connect nodes i hops away, in an anti-clockwise manner. The cycle ends before or at the node we started at. Figure 3.4(c) shows the constructed cycle for $i = 2$. We repeat the procedure for the $i - 1$ nodes successive to the node we started at (node B in the example, for $i = 2$), obtaining one complete or incomplete cycle in each case. Figure 3.4(d) shows the second Type II cycle for the example. Each cycle contains $\lfloor N/i \rfloor$ edges, and there are i Type II cycles. Each node is connected to one preceding node and one successive node i hops away. Thus, N edges are required to connect all nodes with neighbors i hops away on the Type I cycle. The total edges covered by Type II cycles is $i\lfloor N/i \rfloor$. Thus, to cover the $N - i\lfloor N/i \rfloor$ uncovered edges, we form Type III cycles, which are just single

edges.

- **Type III cycles**: These cycles are single edges, each pertaining to one of the $N - i\lfloor N/i \rfloor$ uncovered edges. Figure 3.4(e) shows the cycle for the example.

There is one Type I cycle in the Harary graph, and i Type II and $N - i\lfloor N/i \rfloor$ Type III cycles for each $i = 2, 3, .., \lceil k/2 \rceil$. These cycles cover all the edges in the Harary graph, and thus the number of beads needed for these cycles is the same as needed for the Harary graph. According to Lemma 3.2.4, the Type I cycle uses at most $5s_j$ beads. The following lemma bounds the number of beads needed for the Type II edges.

Lemma 3.2.6 *A Type II cycle constructed from edges connecting nodes i hops apart requires at most $5s_j$ beads.*

Proof: Let the set of terminals in the Harary graph be $\mathcal{T}_j$. Let the set of terminals in the Type II cycle be $\mathcal{T}'_j \subset \mathcal{T}_j$. Consider another instance of the problem, in which only the terminals of $\mathcal{T}'_j$ are connected to the Steiner node MST ST_j. Follow Algorithm 3.2.3 on this instance to form a cycle. According to Lemma 3.2.4, this cycle has at most $5s_j$ beads. The only difference between this instance and the original instance is that the terminals $\mathcal{T}_j \backslash \mathcal{T}'_j$ have been removed. The order of children traversal in the DFS traversal is anti-clockwise. Removing the terminals $\mathcal{T}_j \backslash \mathcal{T}'_j$ does not change the order in which the terminals $\mathcal{T}'_j$ are encountered (compared to the original instance). Thus, the cycle constructed is the same as the Type II cycle in consideration (or has one extra edge if the Type II cycle is not complete). Thus, the Type II cycle has at most $5s_j$ beads. $\square$

Now, we consider the Type III cycles. Each Type III cycle is just an edge. Thus, the required number of beads is at most the number of Steiner nodes in the DFS path between the end-terminals of this edge. Thus, a Type III edge requires at most s_j beads.

Thus, the total number of beads (b_j) required by the Harary graph is as given in Equation 3.2.

$$\begin{aligned} b_j &\leq 5s_j + \sum_{i=2}^{\lceil k/2 \rceil} (5i + N - i\lfloor N/i \rfloor)s_j \\ &= 5s_j + \sum_{i=2}^{\lceil k/2 \rceil} (5i + i(N/i - \lfloor N/i \rfloor))s_j \\ &\leq 5s_j + \sum_{i=2}^{\lceil k/2 \rceil} (6i)s_j \\ &= (6 \sum_{i=1}^{\lceil k/2 \rceil} i - 1)s_j \\ &= (3\lceil k/2 \rceil(\lceil k/2 \rceil + 1) - 1)s_j \end{aligned} \tag{3.2}$$

Since the Steiner components SC_j do not have common Steiner nodes, summing over all SC_j, the number of beads in G_m is at most $(3\lceil k/2 \rceil(\lceil k/2 \rceil + 1) - 1)s$. Thus, a solution with minimum number of beads requires at most $(3\lceil k/2 \rceil(\lceil k/2 \rceil + 1) - 1)s$ beads. □

Theorem 3.2.7 states the main result of this section.

Theorem 3.2.7 *If the optimal network uses s Steiner nodes so that terminals distributed in the Euclidean plane are k-vertex connected, Algorithm 3.2.1 forms a network with maximum of $c(3\lceil k/2 \rceil(\lceil k/2 \rceil + 1) - 1)s$ beads and zero Steiner nodes, in which the terminals are k-vertex connected.*

Proof: The algorithm for finding a k-vertex connected subgraph is a c-approximation (cost of each edge being number of beads required to form it). Thus, according to Lemma 3.2.1, the number of beads required is at most $c(3\lceil k/2\rceil(\lceil k/2\rceil+1)-1)s$. The last step of Algorithm 3.2.1 (sequential removal step) removes beads from the network by allowing them to connect to all nodes within the transmission range, so the resulting network after sequential removal also has maximum of $c(3\lceil k/2\rceil(\lceil k/2\rceil + 1) - 1)s$ relay nodes. $\square$

3.2.2 Computational Complexity

We discuss the time complexity of Algorithm 3.2.1 in this section. For general k, we use the k-approximation algorithm of Kortsarz and Nutov [45] for computing a k-vertex connected subgraph of a graph. The algorithm takes $O(k^2N^3M)$ time, where N is the number of terminals and M is the number of edges in the graph (which is $N(N-1)$ for a complete graph, as in our case). Thus, Steps 1-4 of Algorithm 3.2.1 take $O(k^2N^5)$ time. The sequential removal step checks for k-vertex connectivity of the network N' times, where N' is the number of relays before the sequential removal step (at the end of Step 4 of Algorithm 3.2.1). There are a number of algorithms for checking the k-vertex connectivity of a graph, Esfahanian [27]. We can use the algorithm of Cheriyan and Thurimella [17], which takes $O(k^3(N+N')^2)$ time. Thus, the sequential removal step takes $O(k^3N'(N+N')^2)$ time. Thus, Algorithm 3.2.1 takes $O(k^2N^5 + k^3N'(N+N')^2)$ time.

3.3 Edge Connectivity

We now consider the problem of providing k-edge connectivity to terminal nodes. The algorithm framework to achieve edge connectivity between terminals is similar to the framework for vertex connectivity. To connect two terminal nodes outside each other's transmission range, the relay nodes are placed on the straight line connecting the two nodes. The algorithm proceeds by forming a multi-graph G_c on the terminal nodes. There are k edges between each pair of terminal nodes in G_c. We use the weight function of Equation 3.1 (number of relays required on the edge) to weight the edges. We do not allow the relay nodes to have edges other than the ones required to form the edge they are placed on. Then we compute an approximate minimum cost spanning k-edge connected subgraph (G'_c) of the multi-graph G_c.

The problem of finding the minimum cost spanning k-edge connected subgraph of a graph is NP-Hard (Garey and Johnson [30]). Thus, we use an approximation algorithm for the problem, proposed by Khuller and Vishkin [41]. The algorithm achieves an approximation ratio of 2 for the problem, and takes $O((kn)^2)$ time for a graph with n nodes. The algorithm uses the matroid intersection based algorithm of Gabow [29], which finds k edge-disjoint spanning trees from a root vertex in a directed graph. It is worth noting that the weight function of Equation 3.1 is not a metric as it does not satisfy triangle inequality. Thus, the approximation algorithm of Khuller and Vishkin [41] is the best known for the problem. In the resulting subgraph from the approximation algorithm of Khuller and Vishkin [41], the relay nodes are placed to form the links (of length greater than one) of the

subgraph. In the next section, we prove that this algorithm has an approximation ratio of $10\lceil k/2 \rceil$ for nodes distributed in the Euclidean plane. The solution is then improved by removing some relays. The relays are allowed to form edges with all nodes in their transmission range and sequentially removed if k-edge connectivity is preserved. We call this step the *sequential removal step*. Algorithm 3.3.1 describes the algorithm.

Algorithm 3.3.1 Relay placement for k-edge connectivity

1: Construct a multi-graph $G_c = (V, E_c)$ by adding k edges between each pair of vertices of graph G.

2: Weight the edges of the graph as follows. $|e|$ represents the length of edge e.

$$c_e = \lceil |e| \rceil - 1$$

3: Compute an approximate minimum cost spanning k-edge connected subgraph from this graph G_c using the approximation algorithm proposed by Khuller and Vishkin [41]. Let the resulting graph be G'_c.

4: Place relay nodes (number equal to the weight of the edge) on the edges in G'_c with link costs greater than zero.

5: For all pairs of nodes (including the relay nodes) in G'_c within each other's transmission range, form an edge.

6: For the relay nodes sorted arbitrarily, do the following (starting at $i = 1$):

- Remove node i (and all adjacent edges).
- Check for k-edge connectivity between the terminals.
- If the graph is k-edge connected, repeat for $i = i + 1$, else put back the node i and corresponding edges, and repeat for $i = i + 1$.
- Stop when all relay nodes have been considered.

7: Output the resulting graph.

3.3.1 Proof of Approximation Ratio

We now analyze the algorithm to provide with approximation guarantees. We prove that the algorithm is a $10\lceil k/2 \rceil$-approximation for nodes distributed in the Euclidean plane.

We start with some notation. Let $\mathcal{T}$ be the set of terminals, and $\mathcal{S}$ be the set of optimally placed Steiner nodes (relay nodes) needed to achieve k-edge connectivity among the terminal nodes. Let s be the number of Steiner nodes needed when we place them optimally, i.e, $s = |\mathcal{S}|$. In the proof, we will call the relay nodes placed on straight lines between terminals (as in our algorithm) beads and the optimally placed relay nodes Steiner nodes.

We first prove the following lemma, and then the main result of this section.

Lemma 3.3.1 *A partial k-edge connected network using minimum number of beads contains at most $5\lceil k/2 \rceil s$ beads, where s is the minimum number of Steiner nodes needed.*

Proof: Let $G_0 = (V_0, E_0)$ be the optimal k-edge connected network on terminals (having the minimum number of Steiner nodes).

We follow a procedure similar to Algorithm 3.2.2 to construct a graph with zero Steiner nodes that is k-edge connected on the terminals, and has bounded number of beads. We will prove that this network contains at most $5\lceil k/2 \rceil s$ beads.

The algorithm starts by finding the connected components (SC_i) of Steiner nodes in the graph constructed on the Steiner nodes. It constructs a minimum-degree minimum spanning tree (MST) on Steiner nodes for each connected compo-

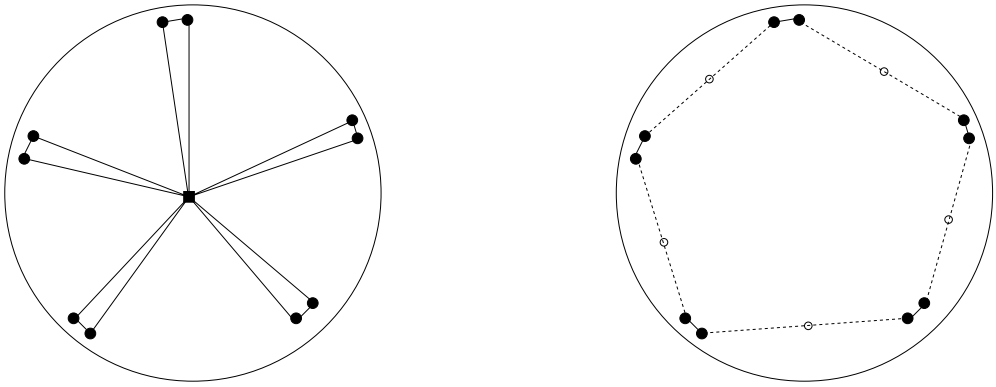
nent, starting with any Steiner node in that component as the root. Let the trees be $ST_1, .., ST_m$. The algorithm then removes Steiner nodes of a connected component SC_j from G_{j-1} and adds beads between the terminals connected to those Steiner nodes to get G_j which is also k-edge connected between terminal nodes. The process is repeated for all connected components, and the resulting graph has zero Steiner nodes and is k-edge connected on the terminals.

Let us now explain the procedure to construct G_j from G_{j-1} by adding beads between the terminal nodes and removing Steiner nodes. Consider the graph formed by the Steiner nodes in ST_j and the terminal nodes within the transmission range of these Steiner nodes. Denote this graph by H_j. We form a cycle among the terminals in H_j using Algorithm 3.2.3[3], and replicate the edges to have $\lceil k/2 \rceil$ copies of each. We use beads to form the edges longer than unit length. The terminals in H_j are k-edge connected since deleting any set of $k-1$ edges does not disconnect the terminals from each other. This procedure maintains k-edge connectivity between the terminal nodes that were k-edge connected because of the Steiner nodes in ST_j. As we do this for all trees $ST_1, .., ST_m$[4], and do not create any $(k-1)$-edge cut in any step, the resulting network is k-edge connected on the terminals.

For each Steiner component ST_j with s_j Steiner nodes, the constructed cycle

[3]There is one change for edge connectivity: if there are only two terminals in H_j, we add two edges between them to form the cycle. However, this charges the Steiner nodes on the DFS path only twice, which is less than 5, thus it does not affect the approximation bound.

[4]If two terminal nodes are adjacent in multiple cycles formed while removing the Steiner components, we form at most k beaded links between them. This suffices for maintaining k-edge connectivity.



(a) Optimal Steiner network (b) Optimal beaded network

Figure 3.5: Approximation ratio tightness example

contains at most $5s_j$ beads, according to Lemma 3.2.4. We replicate the edges to include $\lceil k/2 \rceil - 1$ additional copies of each edge, and thus the graph uses $5\lceil k/2 \rceil s_j$ beads. Since the Steiner components do not have common Steiner nodes, total number of beads required is bounded by $5\lceil k/2 \rceil s$. $\square$

The bound of Lemma 3.3.1 is tight for $k = 2$, i.e., it can be shown that the lower bound on the number of beads required is $5s$ if the optimal Steiner network uses s Steiner nodes. Consider the network in Figure 3.5(a). The circular nodes are terminal nodes, which are 2-edge connected using a single Steiner node in the middle of the circle in the optimal Steiner node solution. If we remove the Steiner node, the optimal network with beads will have a beaded link between every pair of consecutive terminal nodes to form a cycle, and that would require five beads. The resulting network is shown in Figure 3.5(b).

Theorem 3.3.2 states the main result of this section.

Theorem 3.3.2 *If the optimal network uses s Steiner nodes so that terminals distributed in the Euclidean plane are k-edge connected, Algorithm 3.3.1 forms a net-*

work with maximum of $10\lceil k/2 \rceil s$ *beads and zero Steiner nodes, in which the terminals are* k*-edge connected.*

Proof: The algorithm of Khuller and Vishkin [41] is a 2-approximation for finding the minimum cost (cost of each edge being number of beads required to form it) k-edge connected subgraph. Thus, the number of beads required is at most $10\lceil k/2 \rceil s$. The last step of Algorithm 3.3.1 (sequential removal step) removes beads from the network by allowing them to connect to all nodes within the transmission range, so the resulting network after sequential removal also has at most $10\lceil k/2 \rceil s$ relay nodes. □

3.3.2 Computational Complexity

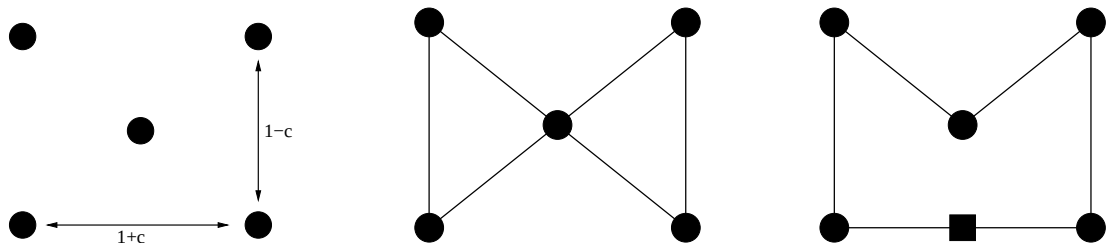
We discuss the time complexity of Algorithm 3.3.1 in this section. We use the k-approximation algorithm of Khuller and Vishkin [41] for computing a k-edge connected subgraph of a graph. The algorithm takes $O((kN)^2)$ time, where N is the number of terminals in the graph. Thus, Steps 1-4 of Algorithm 3.3.1 take $O((kN)^2)$ time. The sequential removal step checks for k-edge connectivity of the network N' times, where N' is the number of relays before the sequential removal step (at the end of Step 4 of Algorithm 3.3.1). There are a number of algorithms for checking the k-edge connectivity of a graph, Esfahanian [27]. We can use the algorithm of Matula [57], which takes $O(k(N+N')^2)$ time. Thus, the sequential removal step takes $O(kN'(N+N')^2)$ time, and Algorithm 3.3.1 takes $O((kN)^2 + kN'(N+N')^2)$. For a network in a cuboid of length L, the maximum number of relays on any edge

in G_c is $O(L)$, and the number of edges in the graph at the output of Step 3 of Algorithm 3.3.1 (G'_c) is $k(N-1)$, thus, $N' = O(kNL)$. Therefore, the algorithm takes $O(k^4N^3L^3)$ time.

3.4 Simulation Results and Discussion

We generate a random network in a unit length square in the Euclidean plane. The nodes are located uniformly and randomly in the network. We simulate the algorithm for k-edge connectivity for $k = 2, 3$. We note the results for number of relays required both at the output of Khuller and Vishkin's approximation algorithm [41], and after sequential removal of relays ensuring k-edge connectivity (see Algorithm 3.3.1). We propose another algorithm that formulates the problem as travelling salesperson problem (TSP), and compare the results with that. The algorithm constructs a complete graph with edge weights defined as in Equation 3.1, and finds a TSP tour on the graph. The total weight of the tour represents the number of relays required. The tour is 2-vertex (and thus edge) connected on terminal nodes. For higher values of k, we replicate the edges of the tour $\lceil k/2 \rceil$ times. We use Concorde [18] to compute the optimal TSP tour.

We first fix the transmission range of the nodes at 0.2 and vary the number of terminal nodes in the network. We randomly generate terminal node locations 10 times. Figures 3.7 and 3.8 show the average and maximum number of relays required (over 10 simulations) for 2-edge connectivity for varying number of terminal nodes. The average number of relays required increases with the number of terminal



(a) Terminal node network (b) Optimal two-edge connected network (c) Optimal TSP network

Figure 3.6: Lower bound on TSP performance

nodes, and then decreases as the number of terminal nodes goes beyond a threshold. The performance of Algorithm 3.3.1 is nearly the same as the TSP output. The performance of Algorithm 3.3.1 gets slightly better than the TSP performance as the number of terminals is increased. Although the TSP algorithm works almost as well as our algorithm, it can be shown that it has an approximation ratio of infinity for 2-edge connectivity. Consider the example terminal node network of Figure 3.6(a), with the constant $0 < c << 1$. Figure 3.6(b) shows the optimal 2-edge connected network, that uses zero relays. Figure 3.6(c) shows the optimal TSP tour, that uses one relay. Thus the lower bound on the approximation ratio of the TSP based algorithm is infinity, and it may work very bad on certain instances of the problem. The approximation ratio of the TSP based algorithm for 2-vertex connectivity can be very high as well, though it is not easy to give an example to prove the lower bound of the ratio to be infinity. The lower bound can in fact be finite, but examples with high performance ratios can be constructed.

Figures 3.9 and 3.10 show the average and maximum number of relays required (over 10 simulations) for 3-edge connectivity for varying number of terminal nodes.

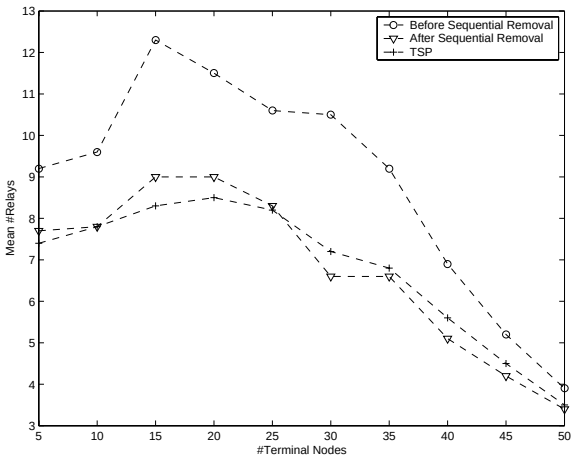


Figure 3.7: Mean number of relays for 2-edge connectivity, $\Delta = 0.2$

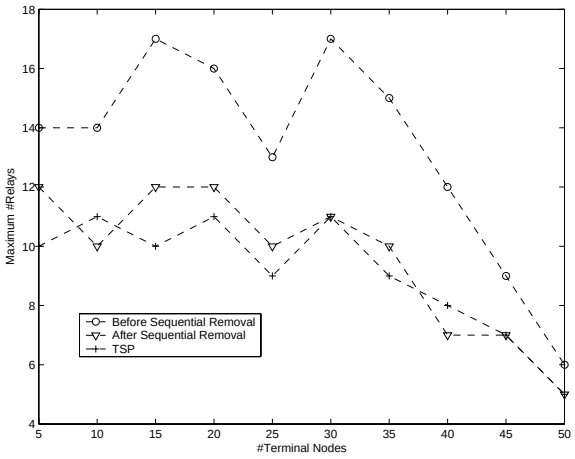


Figure 3.8: Max. number of relays for 2-edge connectivity, $\Delta = 0.2$

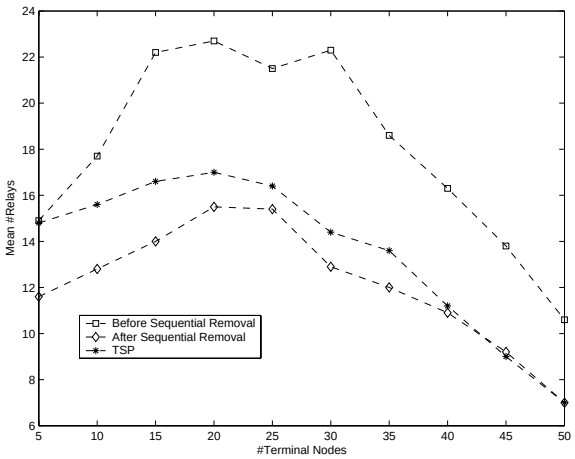


Figure 3.9: Mean number of relays for 3-edge connectivity, $\Delta = 0.2$

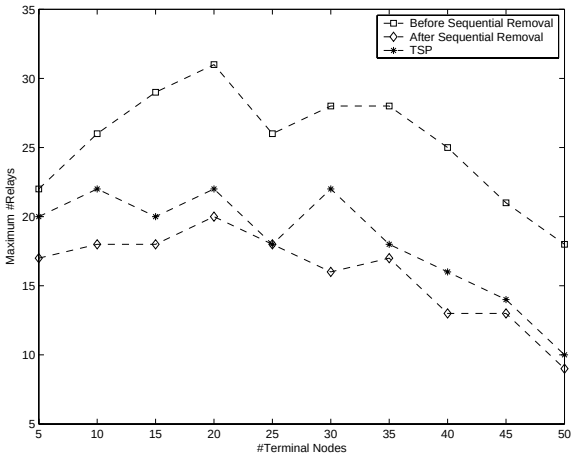


Figure 3.10: Max. number of relays for 3-edge connectivity, $\Delta = 0.2$

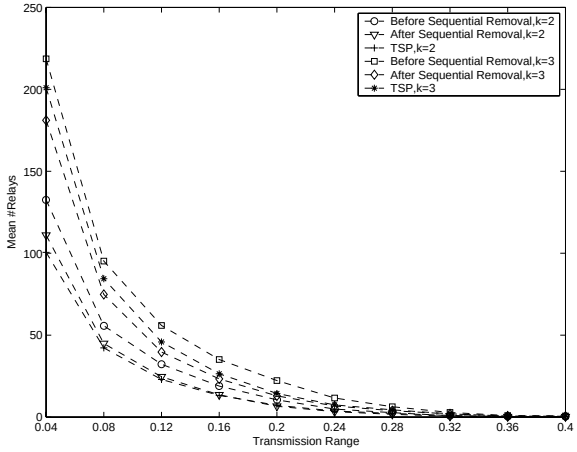


Figure 3.11: Mean number of relays needed for varying transmission range, $N = 30$

The number of relays required follows the same pattern as for 2-edge connectivity, and the gains achieved by sequential relay removal step are more than in the 2-edge connectivity case. This is because there are many more relays in the vicinity of each other in the output before sequential removal in the case of 3-edge connectivity. Thus, when allowed to form links in the neighborhood, the fraction of redundant relays (which can be removed) is much more for $k = 3$ than $k = 2$. Also, the performance of Algorithm 3.3.1 is now better than the TSP-based algorithm, since for $k = 3$, the TSP-based algorithm duplicates all relays that were used for $k = 2$ for all instances, leading to some performance loss.

We now fix the number of terminal nodes at 30, and vary the transmission range from 0.02 to 0.4. The set of locations of the terminal nodes is generated randomly 10 times, and simulations for each transmission range are run on each of those networks. Figures 3.11 and 3.12 show the mean and maximum number of relays required over the simulations for 2- and 3-edge connectivity. The number of

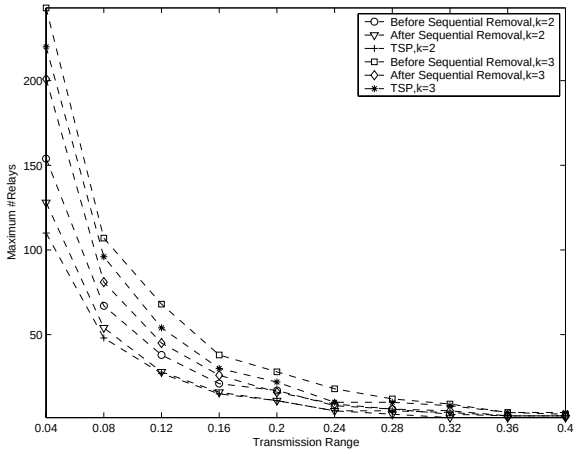


Figure 3.12: Max. number of relays needed for varying transmission range, $N = 30$

relays required decreases exponentially with the transmission range of the nodes. Also, the performance of our algorithm is similar to the TSP output. Kashyap et al. [39] showed that the maximum number of relays required for an MST based algorithm, Lin et al. [50], for connectivity decreases exponentially with the transmission range of the nodes. The simulations show a similar behavior for 2-edge and 3-edge connectivity.

It is worthwhile to note that the results presented in this section will show similar behavior for larger network area or higher number of terminal nodes.

Chapter 4

Generalizations for Connectivity Problems

In this chapter, we discuss a few generalizations of the problems presented in Chapter 3. We first extend the results for partial k-connectivity to full k-connectivity, i.e., k-vertex (edge) connectivity is desired for both terminal and added relay nodes. We then consider the scenario with polygonal obstacles present in the network area, where relay nodes cannot be placed. We extend the algorithms and prove they have the same approximation guarantees as in the case of no obstacles. We then consider an extension in which nodes are distributed in any metric space, e.g., a three dimensional Euclidean space. We provide approximation guarantees for the algorithms presented in Chapter 3 for this generalization. Finally, we consider extensions to heterogeneous networks, where the terminal and relay nodes have different transmission range. We present and analyze approximation algorithms for this network model.

4.1 Full k-Connectivity

We now present the algorithms and analysis for achieving k-vertex (edge) connectivity between terminals and added relays, which we call full k-vertex (edge) connectivity. We first discuss full k-vertex connectivity.

4.1.1 Vertex Connectivity

We use the algorithm proposed in Bredin et al. [14] for constructing a full k-vertex connected topology. The algorithm adds relays to find a subgraph on terminals that is k-vertex connected on terminals using the algorithm presented in Section 3.2. Then, for each edge of the k-connected subgraph with weight greater than zero, i.e., with at least one relay, the algorithm places additional $k-1$ relays at each of the end terminal vertices of the edge, and additional $k-1$ relays along with each relay used on the edge. In the resulting graph, all terminals and relays have k-vertex connectivity, Bredin et al. [14]. The algorithm has been proved to have an approximation ratio of $O(k^4)$ for terminals in the Euclidean plane. We improve the analysis, and prove the algorithm to be an $O(k^3)$-approximation. Theorem 4.1.1 states the result. Here, c is the approximation ratio of the best algorithm for finding a minimum-weight k-vertex connected subgraph of a k-vertex connected graph.

Theorem 4.1.1 *If the optimal network uses s Steiner nodes so that terminals and Steiner nodes are k-vertex connected, the algorithm of Bredin et al. [14] forms a network with at most $3c(3\lceil k/2\rceil(\lceil k/2\rceil+1)-1)ks$ relays and zero Steiner nodes, in which the terminal and relay nodes are k-vertex connected.*

Proof: The optimal network that is k-vertex connected only on terminals uses $s' \leq s$ Steiner nodes. Thus, according to Theorem 3.2.7, the intermediate graph that is k-vertex connected on terminals has at most $c(3\lceil k/2\rceil(\lceil k/2\rceil+1)-1)s' \leq c(3\lceil k/2\rceil(\lceil k/2\rceil+1)-1)s$ relays. As shown in Bredin et al. [14], for each edge between terminals with $w \geq 1$ relays, we duplicate them to have a total of $kw+2(k-1) < 3kw$

relays on the edge. Thus, the total number of relays in the final fully k-vertex connected graph is bounded by $3c(3\lceil k/2\rceil(\lceil k/2\rceil + 1) - 1)ks$, which is an $O(k^3)$-approximation, an $O(k)$ improvement over the bounds proved by Bredin et al. [14]. □

4.1.2 Edge Connectivity

For edge connectivity, we use the algorithm presented in Section 3.3. We use the algorithm to construct a k-edge connected subgraph on terminals, and then duplicate the relays needed on the edges of that subgraph. For each edge of weight greater than zero, we place additional $\lceil k/2\rceil - 1$ relays at each end terminal vertex of the edge, and each relay location on that edge. The resulting graph is k-edge connected on both terminals and relays. We prove the algorithm is an $O(k^2)$-approximation. Theorem 4.1.2 states the result.

Theorem 4.1.2 *If the optimal network uses s Steiner nodes so that terminals and Steiner nodes are k-edge connected, our algorithm forms a network with at most $30\lceil k/2\rceil^2 s$ relays and zero Steiner nodes, in which the terminal and relay nodes are k-edge connected.*

Proof: The optimal network that is k-edge connected only on terminals uses $s' \leq s$ Steiner nodes. Thus, according to Theorem 3.3.2, the intermediate graph that is k-edge connected on terminals has at most $10\lceil k/2\rceil s' \leq 10\lceil k/2\rceil s$ relays. Then, for each edge between terminals with $w \geq 1$ relays, we duplicate them to have a total of $\lceil k/2\rceil w + 2(\lceil k/2\rceil - 1) < 3\lceil k/2\rceil w$ relays on the edge. Thus, the total

number of relays in the final full k-edge connected graph is bounded by $30\lceil k/2 \rceil^2 s$.

□

4.2 Generalization to Restricted Relay Placement

We extend our results for terminals distributed in a Euclidean plane to the scenario where relays cannot be placed in certain polygonal regions of the network. We call these regions forbidden regions. We assume that two nodes can communicate if they are within each other's transmission range even when there is a forbidden region between them. We modify the edge and vertex connectivity algorithms to work with the same approximation guarantees for this generalization.

We follow the same algorithms as before for both edge connectivity and vertex connectivity. It may not be possible to connect two terminals by placing relay nodes on the straight line between them due to the forbidden regions. Thus, Equation 3.1, which represents the number of relays needed to connect two terminals by placing relays on the line between them, cannot be used to weight the edges of the network formed on terminal nodes in our algorithms. Arkin et al. [4] have proposed a polynomial time algorithm for placing the minimum number of relay nodes needed to form a link between two nodes with the presence of polygonal forbidden regions between them. The problem is called the puddle-jumper problem. We modify our edge weights by running the algorithm of Arkin et al. [4] on each pair of terminals in the network to find the minimum number of relay nodes needed for each link, and using that as the weight of each edge. We then run our edge connectivity and

vertex connectivity algorithms on a network with these edge weights. Then, for the selected links, we place the relays according to the algorithm of Arkin et al. [4].

4.2.1 Proof of Approximation Ratio

We first prove that the approximation ratio for the partial k-vertex connectivity algorithm is $O(k^2)$ for terminals distributed in the Euclidean plane. We follow the same construction as before, the only change being that beads (relay nodes) are not placed on straight lines between terminal nodes now; instead they are placed optimally taking forbidden regions into account. The only part of the proof that needs reconsideration to take forbidden regions into account is when Steiner nodes on a tree (ST_j) are removed from the optimal Steiner solution and beads are placed to construct the cycle (and the Harary graph) between terminal nodes connected to tree ST_j (see Algorithm 3.2.3). We argue that the number of relays needed to form a beaded link between two terminals is still upper bounded by the number of Steiner nodes encountered in the depth first traversal between the two terminals: Take any two terminals being connected using beads, and let a be the number of Steiner nodes on the DFS path between them. Thus, there is a placement of Steiner nodes to connect the two terminal nodes using a Steiner nodes. As even Steiner nodes could not be placed in forbidden regions, and we connect the terminals using beads placed according to the optimal algorithm of Arkin et al. [4], the number of beads required is upper bounded by a. Thus, each bead can still be charged to a different Steiner node on the DFS path between the terminals. According to Lemma 3.2.1, each Steiner

node is charged at most $(3\lceil k/2\rceil(\lceil k/2\rceil+1)-1)$ times , so the total number of beads required for replacing the Steiner node tree ST_j is still $(3\lceil k/2\rceil(\lceil k/2\rceil+1)-1)s_j$, s_j being the number of Steiner nodes in ST_j. Thus the total number of beads required in the network is at most $(3\lceil k/2\rceil(\lceil k/2\rceil+1)-1)s$ for the beaded network using minimum number of beads, s being the number of optimal Steiner nodes. As our algorithm uses c-approximations for finding the optimal beaded network, the algorithm is $c(3\lceil k/2\rceil(\lceil k/2\rceil+1)-1)$-approximation. Also, the full k-vertex connectivity algorithm is a $3c(3\lceil k/2\rceil(\lceil k/2\rceil+1)-1)k$-approximation.

The same arguments can be used to prove that the partial k-edge connectivity algorithm is a $10\lceil k/2\rceil$-approximation, and full k-edge connectivity algorithm is a $30\lceil k/2\rceil^2$-approximation.

4.3 Generalization to other Metric Spaces

In this section, we consider the case of achieving k-connectivity (edge or vertex) between terminal nodes placed in any metric space with MST number M, Robins and Salowe [65]. MST number is defined as the maximum node degree in a minimum-degree Minimum Spanning Tree (MST) spanning points from the space. MST number for the Euclidean plane is 5 (Monma and Suri [58]), three-dimensional Euclidean space is 12, and rectilinear plane (two-dimensional space with metric defined by L_1 norm) is 4 (Robins and Salowe [65]). The approximation ratio for the MST based algorithm of Lin et al. [50] for connecting terminals using minimum relays has been shown to be $M-1$ by Măndoui and Zelikovsky [56]. We analyze the worst-case

performance of our algorithms and prove a performance bound of the algorithm with respect to the optimal solution. We first prove that the algorithm proposed for 2-vertex connectivity is a $2M$-approximation.

4.3.1 2-Vertex Connectivity

We start with some notation. Let $\mathcal{T}$ be the set of terminals, and $\mathcal{S}$ be the set of optimally placed Steiner nodes (relay nodes) needed to achieve 2-vertex connectivity among the terminal nodes. Let s be the number of Steiner nodes needed when we place them optimally, i.e, $s = |\mathcal{S}|$. In the proof, we will call the relay nodes placed on straight lines between terminals (as in our algorithm) beads and the optimally placed relay nodes Steiner nodes.

As a recap of our algorithm (Algorithm 3.2.1), it computes a 2-vertex connected subgraph of a complete graph between terminals using the 2-approximation algorithm of Khuller and Raghavachari [40]. If two terminal nodes are more than unit distance apart, it adds beads (relay nodes) to form that link. A link of length l consists of $\lceil l \rceil - 1$ beads.

We first prove the following lemma, followed by the main result of this section.

Lemma 4.3.1 *A partial 2-vertex connected network using minimum number of beads contains at most Ms beads, where s is the number of Steiner nodes in an optimal partial 2-vertex connected network.*

Proof: Let $G_0 = (V_0, E_0)$ be the optimal 2-vertex connected network on terminals (having the minimum number of Steiner nodes).

We follow the procedure of Algorithm 4.3.1 to construct a 2-vertex connected network that has beads and no Steiner nodes. We will prove that this network does not contain more than Ms beads.

Algorithm 4.3.1 starts by finding the connected components (SC_i) of Steiner nodes in the graph constructed on the Steiner nodes. It constructs a Breadth First Search (BFS) spanning tree[1] on Steiner nodes for each connected component, starting with any Steiner node in that component as the root. Let the trees be $ST_1, .., ST_m$. The algorithm then removes Steiner nodes of a connected component SC_j from G_{j-1} and adds beads between the terminals connected to those Steiner nodes to get G_j which is also 2-vertex connected between terminal nodes (Step 4). The process is repeated for all connected components, and the resulting graph has zero Steiner nodes and is 2-vertex connected on the terminals.

[1]Note that we constructed a minimum degree MST in the proof for the Euclidean plane.

Algorithm 4.3.1 Construction of 2-vertex connected network with beads

1: Define a graph $G_S = (S, E_S)$ on the Steiner nodes, where an edge (u, v) is in E_S if it is an edge between the Steiner nodes u, v in G_0.

2: Find all the connected components (SC_i) in G_S.

3: Construct a BFS spanning tree in each connected component, and call the trees $ST_1, .., ST_m$.

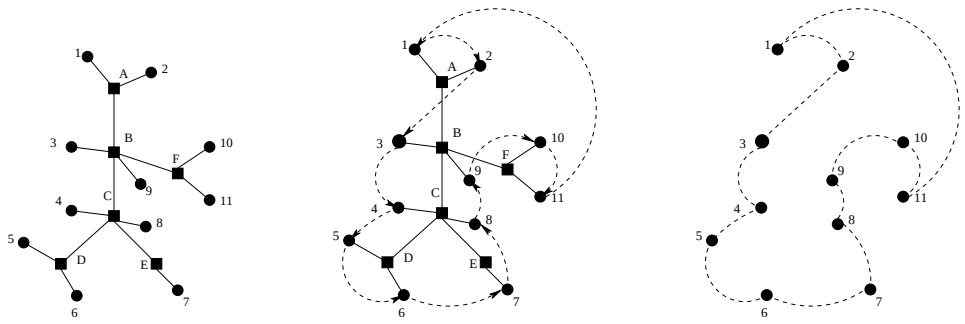
4: Set $j = 1$. While $j \leq m$:

1. Remove the Steiner nodes contained in ST_j from G_{j-1}.

2. Add beads between terminals to get the graph G_j, which is also k-vertex connected on the terminals. The procedure for adding beads and removing Steiner nodes is explained later.

3. Set $j = j + 1$.

5: Output the resulting graph G_m.



(a) Tree on Steiner nodes and terminals

(b) Depth first traversal and cycle creation

(c) Graph after removal of Steiner nodes

Figure 4.1: Example for removal of Steiner nodes and addition of beads

Let us now explain the procedure to construct G_j from G_{j-1} by adding beads between the terminal nodes to construct a cycle (2-vertex connected graph), and removing the Steiner nodes. Consider the graph formed by the Steiner nodes in ST_j and the terminal nodes within the transmission range of these Steiner nodes. Denote this graph by H_j. We add a cycle using beaded (and direct) links between the terminals contained in H_j in G_{j-1} and delete the Steiner nodes of ST_j to get G_j. If there are only two terminals in H_j, then we connect the terminals using a single edge. According to Lemma 3.2.5, G_j is 2-vertex connected.

Algorithm 4.3.2 describes the algorithm for construction of the cycle between terminal nodes connected to the Steiner nodes of ST_j. The algorithm works as follows: Start at the root of ST_j (call the root st_1, dropping subscript j for simplicity). Connect to st_1 all terminal nodes within its transmission range, and mark them. Let the set of marked terminal nodes be $\{t_1, .., t_l\}$. Start a Depth First Search (DFS) traversal of the tree formed by $ST_j \cup \{t_1, .., t_l\}$ (rooted at st_1, denote it by T_j), starting with any child of st_1. The children of a node are traversed in the order of

their distance from each other, i.e., the next child to traverse is the one closest to the current child being traversed and the first to traverse is the one closest to the parent node[2]. Whenever a new Steiner node st_j is encountered in the traversal, all unmarked terminal nodes in its transmission range are connected to it, and added to the set of marked terminal nodes (thus l increases at this step). Figure 4.1(a) shows an example tree constructed using this procedure. While doing the DFS traversal, add required number of beads to form a link between each terminal with the next terminal encountered in the DFS traversal. Complete the cycle by connecting the last added terminal to the first terminal encountered in the DFS traversal. If there are only two terminals in T_j, then connect the terminals using a single edge. The edges longer than unit length are added using the required number of beads. Figure 4.1(b) shows the cycle created between the terminal nodes in the example, starting at terminal 1. Finally, remove the Steiner nodes. Figure 4.1(c) shows the final topology on these terminals nodes.

[2]Note that the order of traversal was anti-clockwise in the proof for the Euclidean plane.

Algorithm 4.3.2 Removal of Steiner nodes and addition of beads in ST_j

1: Start at root st_1 of ST_j.

2: Connect to it all terminals within its transmission range, and mark them.

3: Construct a tree T_j, with the vertex set as the Steiner nodes in ST_j and a leaf vertex corresponding to each marked terminal vertex. The edges are the edges of ST_j and the edges between each Steiner node and the marked terminal vertices connected to it.

4: Do a Depth First Search (DFS) traversal of T_j rooted at st_1, starting with any child of st_1. For each node, traverse its children according to their distance from each other, i.e, the next child traversed is the child closest to the current child being traversed. The first child to be traversed at a Steiner node (except st_1) is the one closest to the parent node.

5: Each time a new Steiner node st_i is encountered, connect it to all unmarked terminal vertices in its range, and mark them. Update T_j by adding these terminal vertices, and continue DFS traversal around st_i from the edge between st_i and its parent.

6: Connect all the terminal vertices in order of their DFS traversal and complete the cycle between them.

7: Add beads to all added edges of length greater than one.

We now give a bound on the number of beads added while constructing G_j from G_{j-1}. The condition in Equation 4.1 states the bound, where b_j is the number of beads added and s_j is the number of Steiner nodes in ST_j. Then, Equation 4.2 bounds the total number of beads required, which proves Lemma 4.3.1.

$$b_j \leq Ms_j \tag{4.1}$$

$$b = \sum_{j=1}^{m} b_j \leq \sum_{j=1}^{m} Ms_j = Ms \tag{4.2}$$

We now prove the bound. We first prove a property of the nearest-neighbor traversal of neighbors around a node. The property is stated in Lemma 4.3.2.

Lemma 4.3.2 *Let there be an arbitrary set of points $P = \{x_1, .., x_l\}, l > 0$ distributed in a unit ball centered at a point x_0 in a metric space. We form a cycle C consisting of the points in P: Add x_1 to C. From the last point added to C, add an edge to the closest point in $P \backslash C$, add the new point to C, and repeat. Finally, add an edge to x_1 when $P = C$. The maximum number of edges of length greater than one is bounded by M, the MST number of the space.*

Proof: Rename the points in P such that x_i is the ith point being added to C. Start constructing C, and each time an edge of length greater than one is encountered, mark the end-point of the edge that was already in C, call it y_i ($i = 1$ for first such edge) and increase i by one. Let the set of marked points be P'.

Remove the unmarked points of P ($x_i \notin P'$) from the unit ball. The ball now contains x_0 and the marked points $y_i \in P'$. When a point in P was added to P',

it was more than unit distance from all the points not in C, thus the distance of $y_i, i \geq 1$ is greater than one from all points $y_j, j > i$. Thus, the distance of each y_i is greater than one from all points in $P' \backslash \{y_i\}$. Thus, the only possible tree spanning the set of points in $P' \cup x_0$ is a star with x_0 directly connected to all points $y_i \in P'$. The degree of x_0 in this MST is equal to the number of points in P', which is equal to the number of edges of length greater than one in C. As the degree of a minimum degree MST in the space is bounded by M, there cannot be more than M edges of length greater than one in C. $\square$

We now define our charging scheme, i.e., how we charge the beads to the Steiner nodes in ST_j. We call the parent Steiner node of a Steiner node as PS, the child Steiner nodes as CS, and terminal node as T. When a node in consideration can be any one of CS or T, we refer to it as CS/T. We classify the ordered pairs of edges traversed around a Steiner node st_i during a DFS traversal of the tree into six types. Figure 4.2 shows the types of ordered pair of edges, where the arrows represent the order of traversal among the edges. We charge one bead to a Steiner node st_i each time one of the six types of ordered pair of edges is traversed. The six types of edge pairs of interest are listed below:

- **Type I**: PS-st_i-CS/T (Figure 4.2(a)). The distance between the parent Steiner node and the node at the other end is always greater than one. If the node at the other end is a Steiner node, the distance is greater than one as the tree ST_j is a BFS tree (nodes two levels away in the tree cannot have distance less than or equal to one). If the other end is a terminal node, the distance is

greater than one due to the construction procedure of T_j (if the distance were less than one, the terminal would be a child of the PS node).

- **Type II**: CS/T-st_i-PS (Figure 4.2(b)). The distance between the parent Steiner node and the node at the other end is always greater than one, reason being the same as explained for Type I edge pair.

- **Type III**: CS-st_i-T (Figure 4.2(c)), if the distance between the end-nodes is greater than one.

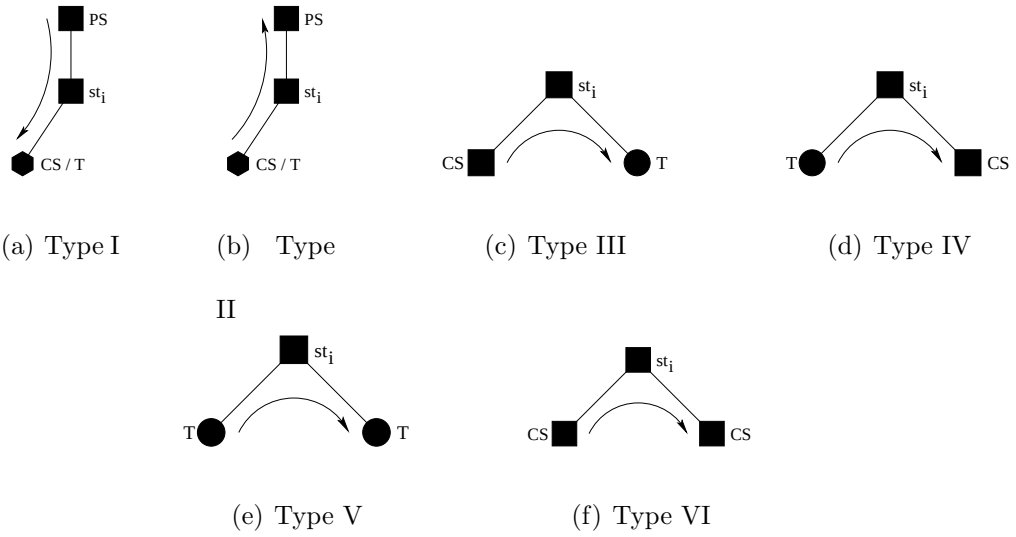
- **Type IV**: T-st_i-CS (Figure 4.2(d)), if the distance between the end-nodes is greater than one.

- **Type V**: T-st_i-T (Figure 4.2(e)), if the distance between the end-nodes is greater than one.

- **Type VI**: CS-st_i-CS (Figure 4.2(f)), if the distance between the end-nodes is greater than one.

As all ordered pairs of edges are traversed once during the traversal, and the ones for which a bead is charged have the end-points more than distance one apart, the maximum number of times a Steiner node is charged for a bead is M (using Lemma 4.3.2, since all neighbors of a Steiner node are contained in a unit ball centered at it).

We need to prove that this charging scheme charges the required number of beads in the construction of the cycle among terminals. Every time we add an edge to connect two terminal nodes in T_j, we claim that the number of beads required



(a) Type I (b) Type II (c) Type III (d) Type IV

(e) Type V (f) Type VI

Figure 4.2: Types of edge pairs charged to Steiner node st_i

(given by Equation 3.1) can be charged to the Steiner nodes encountered in the DFS traversal between the two terminal nodes using our charging scheme. Let the two terminal vertices be t_x and t_y, and let there be $l > 0$ Steiner nodes on the DFS path between them. Renumber the Steiner nodes on the DFS path as $st_1, .., st_l$. We consider the following cases and prove that the charging scheme charges the required number of beads to the Steiner nodes:

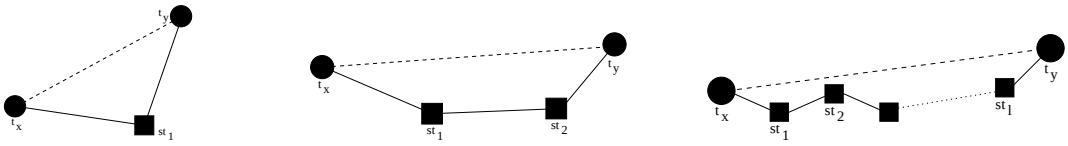
- **Case 1**: $l = 1$: This case is depicted in Figure 4.3(a). If both terminals are connected to the same Steiner node st_1, a bead is needed only if they are more than distance one apart. In that case, the pair of edges $t_x - st_1 - t_y$ is of Type V and thus the Steiner node st_1 can be charged for the bead required.

- **Case 2**: $l = 2$: This case is depicted in Figure 4.3(b). Let the two Steiner nodes in the DFS path be st_1 and st_2, with st_1 being the parent of st_2 in the tree. Let t_x be connected to st_1 and t_y to st_2. Since st_1 is the parent of st_2, we connected all unmarked neighboring terminal nodes to st_1 first. Thus, t_y

is more than distance one apart from st_1. If two beads are needed between t_x and t_y, the distance between t_x and st_2 is more than one and between st_1 and t_y is more than one. Thus the pairs of edges, $t_x - st_1 - st_2$ and $st_1 - st_2 - t_y$ are Type IV and I respectively for Steiner nodes st_1 and st_2 respectively. Thus, one bead can be charged to each Steiner node. If we need one bead between t_x and t_y, that can be charged to st_2 as the pair of edges $st_1 - st_2 - t_y$ is always Type I for st_2. The explanation for the case where st_2 is the parent of st_1 is similar.

- **Case 3**: $l > 2$: This case is depicted in Figure 4.3(c). The total number of beads required is upper bounded by the number of Steiner nodes on any path between t_x and t_y. There are two cases to be considered:

    1. The DFS path stays on one branch of the DFS tree. Let st_i be the parent of st_{i+1} for $i \in \{1, .., l-1\}$. One bead can be charged to each of the nodes $st_2, .., st_l$ as the pair of edges involving them in the path are of Type I. If the distance between t_x and st_2 is less than one, the path can be modified by connecting t_x directly to st_2, and there is a path with $l-1$ Steiner nodes, and thus st_1 can be removed. If it is greater than one, then a bead can be charged to st_1 as the pair of edges $t_x - st_1 - st_2$ is of Type IV. Thus, there is a path of $l-1$ or l Steiner nodes between t_x and t_y (which is the upper bound for the number of beads required), and there are enough Steiner nodes that can be charged once. The case of going up a DFS branch, i.e., when st_i is the parent of st_{i-1} for $i \in \{2, .., l\}$ is



(a) One Steiner node, $l = 1$

(b) Two Steiner nodes, $l = 2$

(c) More than two Steiner nodes, $l > 2$

Figure 4.3: DFS paths of different lengths

similar.

2. The DFS path moves between two branches of the DFS tree. Let st_k be the Steiner node at which the two branches start. In this DFS path, st_i is the parent node of st_{i-1}, $i \in \{2, .., k\}$, and st_i is the parent node of st_{i+1}, $i \in \{k, .., l-1\}$. One bead can be charged to each of the nodes $st_1, .., st_{k-1}$ as the pair of edges involving them in the path are of Type II. Similarly, one bead can be charged to each of the nodes $st_{k+1}, .., st_l$ as the pair of edges involving them in the path are of Type I. If the distance between st_{k-1} and st_{k+1} is less than one, the path can be modified by connecting st_{k-1} directly to st_{k+1}, and there is a path with $l-1$ Steiner nodes, and thus st_k can be removed. If it is greater than one, then a bead can be charged to st_k as the pair of edges $st_{k-1} - st_k - st_{k+1}$ is of Type VI. Thus, there is a path of $l-1$ or l Steiner nodes between t_x and t_y (which is the upper bound for the number of beads required), and there are enough Steiner nodes that can be charged once.

We have shown that the charging scheme charges the required number of beads to the Steiner nodes, and each Steiner node is charged at most M times. Summing

over all connected components of Steiner nodes in the network, the total number of beads required is within M times the number of Steiner nodes. Thus, the relation in Equation 4.2 holds for the beaded network we constructed. Hence, the relation holds for the optimal 2-vertex connected beaded network as well. □

Theorem 4.3.3 states the main result of this section.

Theorem 4.3.3 *If the optimal network uses s Steiner nodes so that terminals distributed in metric space of MST number M are 2-vertex connected, Algorithm 3.2.1 forms a network with maximum of $2Ms$ beads and zero Steiner nodes, in which the terminals are 2-vertex connected.*

Proof: The algorithm of Khuller and Raghavachari [40] is used to compute a 2-vertex connected subgraph of a graph, which is a 2-approximation. According to Lemma 4.3.1, the optimal beaded network uses at most Ms beads. Thus, the number of beads required by Algorithm 3.2.1 is at most $2Ms$. The last step (sequential removal) removes beads from the network by allowing them to connect to all nodes within the transmission range, so the resulting network after sequential removal also has at most of $2Ms$ relay nodes. □

For the Euclidean plane, for $k = 2$, the result of Theorem 4.3.3 matches that of Theorem 3.2.7 since $M = 5$ for the Euclidean plane. The analysis also extends the results of full 2-vertex connectivity. The algorithm can be proved to be a $12M$-approximation of the optimal.

4.3.2 Edge Connectivity

We now consider partial k-edge connectivity. We prove Algorithm 3.3.1 to be a $2M\lceil k/2\rceil$-approximation. We first prove the following lemma, followed by the main result.

Lemma 4.3.4 *A partial k-edge connected network using minimum number of beads contains at most $M\lceil k/2\rceil s$ beads, where s is the number of Steiner nodes in an optimal partial k-edge connected network.*

Proof: We follow the Algorithm 4.3.1 to construct a cycle (2-edge connected graph) with zero Steiner nodes. The only difference is that when there are only two terminal nodes connected to a Steiner component, we connect them using two edges[3]. Thus, this graph has at most Ms beads. We replicate the edges to get $\lceil k/2\rceil$ copies of each edge. Thus, the resulting network has at most $M\lceil k/2\rceil s$ beads. We showed in Section 3.3.1 that this procedure maintains the k-edge connectivity for terminal nodes. Thus, a partial k-edge network with minimum number of beads contains at most $M\lceil k/2\rceil s$ beads. □

The bound of Ms for an optimal beaded network is tight for 2-edge connectivity. Here is an example that shows the lower bound on number of beads is Ms: let there be M pairs of nodes (placed right next to each other) placed more than unit distance apart around a Steiner node in an optimal Steiner solution. The optimal beaded network will connect them in a cycle using M beads in place of one Steiner

[3]This charges the Steiner nodes on the DFS path only twice, which is less than M in general, thus it does not affect the approximation bound.

node. The example is similar to the example to prove tightness of the analysis for 2-edge connectivity in the Euclidean plane in Section 3.3.

Theorem 4.3.5 states the main result of this section.

Theorem 4.3.5 *If the optimal network uses s Steiner nodes so that terminals distributed in metric space of MST number M are k-edge connected, Algorithm 3.3.1 forms a network with maximum of $2M\lceil k/2\rceil s$ beads and zero Steiner nodes, in which the terminals are k-edge connected.*

Proof: The algorithm of Khuller and Vishkin [41] is used to compute a k-edge connected subgraph of a graph, which is a 2-approximation. According to Lemma 4.3.4, the optimal beaded network uses at most $M\lceil k/2\rceil s$ beads. Thus, the number of beads required by Algorithm 3.3.1 is at most $2M\lceil k/2\rceil s$. The last step (sequential removal) removes beads from the network by allowing them to connect to all nodes within the transmission range, so the resulting network after sequential removal also has maximum of $2M\lceil k/2\rceil s$ relay nodes. □

For the Euclidean plane, the result of Theorem 4.3.5 matches that of Theorem 3.3.2 since $M = 5$ for the Euclidean plane. The analysis also extends the results of full k-edge connectivity. The algorithm can be proved to be a $6M\lceil k/2\rceil^2$-approximation of the optimal.

4.4 Generalization to Heterogeneous Networks

In this section, we consider networks with non-uniform transmission range of terminal and relay nodes. We assume the sensor nodes have transmission range in

the range $[T_{min}, T_{max}]$, and the relay nodes have a transmission range of T_{relay}. We normalize the transmission range, and let $T_{min} = 1$, $T_{max} = \alpha$, $T_{relay} = \gamma$. We assume two nodes can communicate if both are within each other's transmission range. That is, for nodes i and j to communicate, the distance between them should not be more than the minimum of their transmission ranges. We first consider k-vertex connectivity for terminals distributed in the Euclidean plane, under this heterogenous model. We later propose and analyze an algorithm for achieving k-edge connectivity.

4.4.1 Vertex Connectivity

The algorithm for achieving partial k-vertex connectivity is similar to Algorithm 3.2.1, and the same as proposed by Han et al. [33]. The only modification from Algorithm 3.2.1 is the set of positions the relays are placed at, thus affecting the weight function of Equation 3.1. In the algorithm for the heterogenous model, if two nodes i, j with transmission ranges T_i, T_j are more than $\min\{T_i, T_j\}$ apart (let the distance be l_{ij}), we use Algorithm 4.4.1 to place relays for connecting them. For each pair of terminals, we assign the number of relays used as the edge weight that is used in Step 2 of Algorithm 3.2.1.

Algorithm 4.4.1 Relay placement in heterogeneous networks

1: Let $T_i \leq T_j$. Place one relay r at distance $\min\{T_i, \gamma\}$ from i.

2: If $l_{rj} > \min\{T_j, \gamma\}$, place one relay r' at distance $\min\{T_j, \gamma\}$ from j.

3: Place $\lceil l_{rr'}/\gamma \rceil - 1$ relays at equal spacing between the two relays r, r'.

We prove for $k = 2$, this algorithm has an approximation ratio of

$$2(5 + 11\lceil \log_{\sqrt{3}}(\min\{\alpha, \gamma\})\rceil + 5\mathcal{I}_{\lceil \log_{\sqrt{3}} \gamma\rceil > \lceil \log_{\sqrt{3}} \alpha\rceil}).$$

Here, $\mathcal{I}_x$ is the indicator function, which is 1 if condition x is true, else it is 0. We still call the relays placed on straight line between terminals beads, and optimally placed relays as Steiner nodes. We prove the following lemma, followed by the main result.

Lemma 4.4.1 *A heterogeneous network that is 2-vertex connected on terminal nodes using the minimum number of beads contains at most* $(5 + 11\lceil \log_{\sqrt{3}}(\min\{\alpha, \gamma\})\rceil + 5\mathcal{I}_{\lceil \log_{\sqrt{3}} \gamma\rceil > \lceil \log_{\sqrt{3}} \alpha\rceil})s$ *beads, where* s *is the minimum number of Steiner nodes needed.*

Proof: Let $G_0 = (V_0, E_0)$ be the optimal 2-vertex connected network on terminals (having the minimum number of Steiner nodes).

We follow a procedure similar to Algorithm 3.2.2 to construct a graph with zero Steiner nodes that is 2-vertex connected on the terminals, and has bounded number of beads. We will prove that this network contains at most $(5+11\lceil \log_{\sqrt{3}}(\min\{\alpha, \gamma\})\rceil + 5\mathcal{I}_{\lceil \log_{\sqrt{3}} \gamma\rceil > \lceil \log_{\sqrt{3}} \alpha\rceil})s$ beads. Thus, the network with minimum number of beads has at most $(5 + 11\lceil \log_{\sqrt{3}}(\min\{\alpha, \gamma\})\rceil + 5\mathcal{I}_{\lceil \log_{\sqrt{3}} \gamma\rceil > \lceil \log_{\sqrt{3}} \alpha\rceil})s$ beads.

The algorithm starts by finding the connected components (SC_i) of Steiner nodes in the graph constructed on the Steiner nodes. It constructs a minimum-degree minimum spanning tree (MST) on Steiner nodes for each connected component, starting with any Steiner node in that component as the root. Let the trees be $ST_1, .., ST_m$. The algorithm then removes Steiner nodes of a connected component SC_j from G_{j-1} and adds beads between the terminals connected to those

Steiner nodes to get G_j which is also 2-vertex connected between terminal nodes. The process is repeated for all connected components, and the resulting graph has zero Steiner nodes and is 2-vertex connected on the terminals.

Let us now explain the procedure to construct G_j from G_{j-1} by adding beads between the terminal nodes and removing Steiner nodes. We first consider the case $\gamma \geq 1$, i.e., $T_{relay} \geq T_{min}$. Consider the graph formed by the Steiner nodes in ST_j and the terminal nodes within the transmission range of these Steiner nodes. Denote this graph by H_j. The algorithm is similar to Algorithm 3.2.3, but differing in the order of traversal of children of a node in the DFS traversal. The algorithm works as follows: Start at the root of ST_j (call the root st_1, dropping subscript j for simplicity). Connect to st_1 all terminal nodes within its transmission range, and mark them. Let the set of marked terminal nodes be $\{t_1, .., t_l\}$. Start a Depth First Search (DFS) traversal of the tree formed by $ST_j \cup \{t_1, .., t_l\}$ (rooted at st_1), starting with any child of st_1. The order of traversal of children of a Steiner node is described in Algorithm 4.4.2, and explained next. Table 4.1 describes the notation used in Algorithm 4.4.2.

Let the Steiner node in consideration be st_a. We partition the area around st_a into concentric rings, with the radius of outer boundary of ring i being d^{i-1}, where d is a constant we choose. The procedure was used for the analysis of size of maximal independent sets in a heterogeneous network by Banerjee and Khuller [8]. The first ring is a circle of radius one. Since we only consider bi-directional links, all neighbors of st_a are within a distance γ from it. Thus, the last ring of interest is such that $d^{i-1} = \gamma$. Thus, $i \leq \lceil \log_d \gamma \rceil + 1$. If $\lceil \log_d \gamma \rceil > \lceil \log_d \alpha \rceil$, only Steiner node

Table 4.1: Notation

Symbol	**Definition**
st_a	Steiner node whose children to traverse
d	Constant for calculating radius of rings
i_m	First ring with non-zero number of nodes
i_M	Last ring with non-zero number of nodes
l_i	Number of nodes in ring i
n_j^i	jth node in ring i
$\mathcal{C}_a$	Cycle among children and parent of st_a

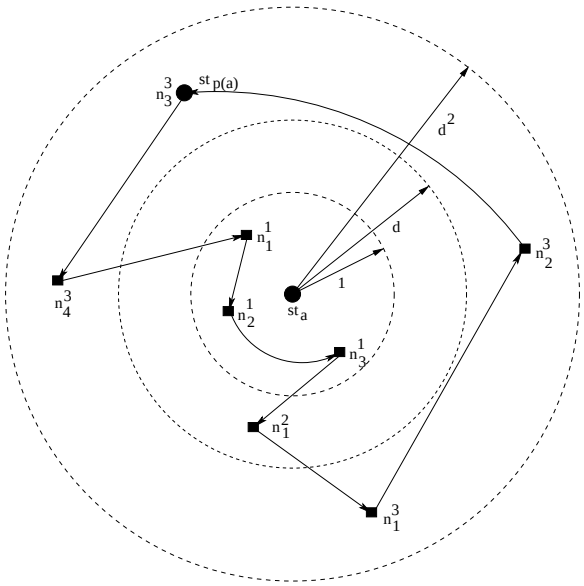


Figure 4.4: Order of DFS traversal around a Steiner node in a heterogeneous network

neighbors of st_a are present beyond a distance α from st_a. In this case, we construct rings of outer boundary radius d^{i-1} for $i \leq \lceil \log_d \alpha \rceil + 1$, and one ring between circles of radius $d^{\lceil \log_d \alpha \rceil}$ and γ. Figure 4.4 shows an example ring division around node st_a. The circular nodes are Steiner nodes, and square nodes are terminals.

Let the number of nodes connected to st_a (Steiner nodes and terminals, excluding st_a) in ring i be l_i. Let the ring with the lowest index with non-zero nodes be i_m, and the ring with the highest index with non-zero nodes be i_M. For all rings i, denote the node with the largest distance from next neighbor in the ring in a clockwise sweep around st_a by n_1^i. Make an *anti-clockwise* sweep around st_a, starting at n_1^i, and sequentially number the encountered nodes $n_2^i, n_3^i, .., n_{l_i}^i$. Start at ring $i = i_m$. Make an incomplete cycle by forming the edges (n_1^i, n_2^i),(n_2^i, n_3^i),..,$(n_{l_i-1}^i, n_{l_i}^i)$. Connect node $n_{l_i}^i$ with n_1^j, such that j is the next higher ring with non-zero nodes. Construct the incomplete cycle for ring $i = j$, and continue the procedure till $i = i_M$. Complete the cycle by connecting $n_{l_{i_M}}^{i_M}$ with $n_1^{i_m}$. If there is one node ($l_i = 1$) in a ring, then n_1^i is connected to both $n_{l_{i-1}}^{i-1}$ and n_1^{i+1}. Call the constructed cycle $\mathcal{C}_a$. Figure 4.4 shows the cycle constructed for the example.

$\mathcal{C}_a$ yields the order of traversal of children around the node st_a. In the DFS traversal, let the parent Steiner node of st_a be $st_{p(a)}$. The first child of st_a traversed is the neighbor of $st_{p(a)}$ in the anti-clockwise direction in $\mathcal{C}_a$. All children are then traversed in the order they are encountered in an anti-clockwise traversal of $\mathcal{C}_a$. Figure 4.4 shows the order of traversal, starting at node $st_{p(a)}$.

Algorithm 4.4.2 DFS traversal order around a Steiner node

1: Partition the area around the Steiner node (st_a) into concentric rings, with the radius of outer boundary of ring i being d^{i-1}, $1 \leq i \leq \lceil \log_d(\min\{\alpha, \gamma\}) \rceil + 1$.

2: If $\lceil \log_d \gamma \rceil > \lceil \log_d \alpha \rceil$, add a concentric ring between circles of radius $d^{\lceil \log_d \alpha \rceil}$ and γ.

3: Set $i = i_m$. While $i \leq i_M$:

1. Make a clockwise sweep in ring i around st_a.
2. Index node with largest distance from next neighbor in the sweep as n_1^i.
3. Make an anticlockwise sweep around st_a in ring i, starting at n_1^i.
4. Sequentially connect the nodes in order of traversal. Number them as $n_2^i, n_3^i, ...$ Do not connect $n_{l_i}^i$ with n_1^i.
5. Set $i = i + 1$.

4: Set $i = i_m$. While $i < i_M$:

1. Let $j > i$ be the next higher ring with $l_j > 0$. Connect $n_{l_i}^i$ with n_1^j.
2. Set $i = j$.

5: Connect $n_{l_{i_M}}^{i_M}$ with $n_1^{i_m}$ to form cycle $\mathcal{C}_a$.

6: DFS traversal order is anti-clockwise around $\mathcal{C}_a$.

The rest of the algorithm is the same as Algorithm 3.2.3. When a new Steiner node st_j is encountered in the DFS traversal around st_a, we mark all unmarked terminal nodes in the Steiner node's transmission range and connect them to it. While doing the DFS traversal, use Algorithm 4.4.1 to add required number of beads to form a link between each terminal with the next terminal encountered in the DFS traversal. Complete the cycle by connecting the last added terminal to the first terminal encountered in the DFS traversal[4]. We then remove the Steiner nodes. Repeat this procedure to construct cycles for each ST_j to get the final beaded graph G_m. According to Lemma 3.2.5, a beaded graph constructed by forming a cycle between terminals connected to each ST_j is 2-vertex connected. Thus, G_m is 2-vertex connected.

We use the charging scheme used in Section 3.2.1 to charge each Steiner node st_a. We set $d = \sqrt{3}$ for computational ease in the Euclidean plane, Banerjee and Khuller [8]. As a recap, we charge the following type of ordered edge pairs in the cycle $\mathcal{C}_a$:

- **Type I**: Steiner-st_a-Steiner.
- **Type II**: Steiner-st_a-Terminal, if the Euclidean distance between the end-nodes is greater than minimum of the transmission range of the two end-nodes.

[4]Note that there will be at least two terminals connected to the Steiner nodes of ST_j. If there were only one terminal node, the Steiner nodes of ST_j could be deleted from the optimal Steiner graph without affecting the connectivity. In case of two terminal nodes, adding one edge between the two makes it a complete graph, which suffices.

- **Type III**: Terminal-st_i-Steiner, if the Euclidean distance between the end-nodes is greater than minimum of the transmission range of the two end-nodes.

- **Type IV**: Terminal-st_i-Terminal, if the Euclidean distance between the end-nodes is greater than minimum of the transmission range of the two end-nodes.

We call an edge between the end-points of an ordered edge pair that we charge a long edge. Since each pair of neighbors of st_a could connect to each other through st_a, an edge between them would use at most one bead (since beads and Steiner nodes have the same transmission range). Thus, each long edge charges st_a one bead (at most one for Type I edges). We first state the following result about the number of long edges in each ring i around st_a, that is an extension of a result by Banerjee and Khuller [8]:

Lemma 4.4.2 *The number of long edges in the anti-clockwise sweep in each ring i, $i > 1$ is at most 11.*

Proof: Type I edges are between Steiner nodes. Since each ST_j is a minimum degree MST, we know from Property 3.2.3 that the angle covered by the sweep between the two Steiner nodes is at least 60 degrees. We now consider the other three types of long edges, each of which has the condition that the two neighbor nodes cannot communicate directly. The transmission range of all nodes in ring i, $i > 1$ is at least their distance from st_a (which is at least d^{i-2}) since they can connect to st_a and bi-directionality of links is necessary for communication. Consider a pair of nodes n_j^i, n_{j+1}^i such that n_j^i is on the inner boundary of ring i, and n_{j+1}^i is on the

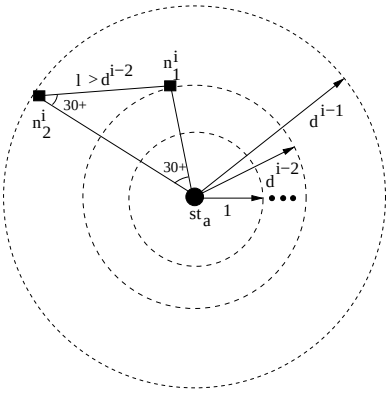


Figure 4.5: Minimum angle covered by a long edge of Type II/III/IV

outer boundary of ring i. Let the transmission range of n_j^i be d^{i-2}. Thus, the two nodes cannot communicate with each other directly if the distance between them is greater than d^{i-2}. In this case, for $d = \sqrt{3}$, the angle between the two edges $n_j^i - st_a$ and $st_a - n_{j+1}^i$ is more than 30 degrees. Figure 4.5 shows the scenario. If the node n_j^i is above the inner boundary, or the node n_{j+1}^i is inside the outer boundary, the angle between the two edges for the same distance is higher. If the transmission range of n_j^i is more than d^{i-2}, the distance required for the two nodes to be outside communication range increases, thus increasing the angle. Since the anti-clockwise sweep around st_a covers 360 degrees around it, the number of such pairs is bounded by $\lfloor \frac{2\pi}{\min\{\frac{\pi}{6}+\epsilon, \frac{\pi}{3}\}} \rfloor = \lfloor \frac{2\pi}{\frac{\pi}{6}+\epsilon} \rfloor = 11$. $\square$

The first ring is a circle of radius one, and the transmission range of all nodes (including st_a, since we are looking at the case $\gamma \geq 1$) is at least one. Thus, the number of long edges in the sweep in this ring is at most 5, as shown in Section 3.2.1. If $\lceil \log_{\sqrt{3}} \gamma \rceil > \lceil \log_{\sqrt{3}} \alpha \rceil$, then the last ring (between circles of radius $d^{\lceil \log_{\sqrt{3}} \alpha \rceil}$ and γ) contains only Steiner nodes. According to Property 3.2.2, there are at most 5

Steiner nodes connected to st_a in ST_j. Thus, there are at most 5 long edges (all Type I) in the sweep in this ring.

In cycle $\mathcal{C}_a$, we use all edges encountered in the anti-clockwise sweep of the rings except the longest edge (between $n^i_{l_i}$ and n^i_1). Since we do not use the longest edge, st_a is charged at most 10 times for each cycle edges in each ring $i > 1$. For the first ring, st_a is charged at most 4 times. When we connect $n^i_{l_i}$ with the first node in the next higher ring (to ring i_m when $i = i_M$), that edge uses at most one bead (since they both were directly connected to st_a, and beads have the same transmission range as st_a). Thus, we charge st_a for one bead for each such cross-ring edge. The number of such beads is equal to the number of non-empty rings. Thus, each non-empty ring i charges st_a for at most 11 beads for $i > 1$, and the first ring charges st_a for at most 5 beads. If $\lceil \log_{\sqrt{3}} \gamma \rceil > \lceil \log_{\sqrt{3}} \alpha \rceil$, the last ring charges st_a for at most 5 beads.

Thus, st_a is charged for at most $5 + 11\lceil \log_{\sqrt{3}}(\min\{\alpha, \gamma\}) \rceil + 5\mathcal{I}_{\lceil \log_{\sqrt{3}} \gamma \rceil > \lceil \log_{\sqrt{3}} \alpha \rceil}$ beads. Here, $\mathcal{I}_x$ is the indicator function, which is 1 if condition x is true, else it is 0. Thus, the final solution G_m uses at most $(5 + 11\lceil \log_{\sqrt{3}}(\min\{\alpha, \gamma\}) \rceil + 5\mathcal{I}_{\lceil \log_{\sqrt{3}} \gamma \rceil > \lceil \log_{\sqrt{3}} \alpha \rceil})s$ beads.

We now consider the case $\gamma < 1$. In this case, the relay nodes have a transmission range smaller than all terminal nodes. Since the communication topology is bi-directional, all edges adjacent to the Steiner nodes in the optimal Steiner solution are at most of length γ. Therefore, all neighbors of a Steiner node are within a circle of radius γ, and all have a transmission range at least γ. Thus, each long edge of Type II, III or IV covers more than 60 degrees around a Steiner node, and

there are at most 5 Type I edges (Property 3.2.2), each pair covering at least 60 degrees. As proved in Section 3.2.1, each Steiner node is charged for at most 5 beads by the anti-clockwise DFS traversal in this case. Thus, G_m uses at most $(5 + 11\lceil \log_{\sqrt{3}}(\min\{\alpha, \gamma\})\rceil + 5\mathcal{I}_{\lceil \log_{\sqrt{3}} \gamma\rceil > \lceil \log_{\sqrt{3}} \alpha\rceil})s$ beads. $\square$

Theorem 4.4.3 states the main result of this section.

Theorem 4.4.3 *If an optimal heterogeneous network uses s Steiner nodes so that terminals distributed in the Euclidean plane are 2-vertex connected, Algorithm 3.2.1 (using Algorithm 4.4.1 for relay placement) forms a network with maximum of* $2(5 + 11\lceil \log_{\sqrt{3}}(\min\{\alpha, \gamma\})\rceil + 5\mathcal{I}_{\lceil \log_{\sqrt{3}} \gamma\rceil > \lceil \log_{\sqrt{3}} \alpha\rceil})s$ *beads and zero Steiner nodes, in which the terminals are 2-vertex connected.*

Proof: The algorithm of Khuller and Raghavachari [40] is used to compute a 2-vertex connected subgraph of a graph, which is a 2-approximation. Thus, according to Lemma 4.4.1, the number of beads required by our algorithm is at most $2(5 + 11\lceil \log_{\sqrt{3}}(\min\{\alpha, \gamma\})\rceil + 5\mathcal{I}_{\lceil \log_{\sqrt{3}} \gamma\rceil > \lceil \log_{\sqrt{3}} \alpha\rceil})s$. The last step of Algorithm 3.2.1 (sequential removal step) removes beads from the network by allowing them to connect to all nodes within the transmission range, so the resulting network after sequential removal also has maximum of $2(5 + 11\lceil \log_{\sqrt{3}}(\min\{\alpha, \gamma\})\rceil + 5\mathcal{I}_{\lceil \log_{\sqrt{3}} \gamma\rceil > \lceil \log_{\sqrt{3}} \alpha\rceil})s$ relay nodes. $\square$

For homogeneous networks ($\alpha = \gamma = 1$), the result of Theorem 4.4.3 matches that of Theorem 3.2.7 for 2-vertex connectivity. The analysis also extends the results of full 2-vertex connectivity. The algorithm can be proved to be a $12(5 + 11\lceil \log_{\sqrt{3}}(\min\{\alpha, \gamma\})\rceil + 5\mathcal{I}_{\lceil \log_{\sqrt{3}} \gamma\rceil > \lceil \log_{\sqrt{3}} \alpha\rceil})$-approximation.

4.4.2 Edge Connectivity

We now consider partial k-edge connectivity. The algorithm is similar to partial vertex connectivity. We follow Algorithm 3.3.1, with one modification: we place the relays using Algorithm 4.4.1. The difference from vertex connectivity is that we allow for multiple edges between the same pair of nodes.

We first prove the following lemma, followed by the main result.

Lemma 4.4.4 *A partial k-edge connected heterogeneous network using minimum number of beads contains at most $(5 + 11\lceil\log_{\sqrt{3}}(\min\{\alpha,\gamma\})\rceil + 5\mathcal{I}_{\lceil\log_{\sqrt{3}}\gamma\rceil>\lceil\log_{\sqrt{3}}\alpha\rceil})$ $\lceil k/2\rceil s$ beads, where s is the number of Steiner nodes in an optimal partial k-edge connected heterogeneous network.*

Proof: We follow the same procedure as in the proof for 2-vertex connectivity in heterogeneous networks to construct a 2-edge connected graph with zero Steiner nodes. The only difference is that when there are only two terminal nodes connected to a Steiner component, we connect them using two edges[5]. Thus, this graph has at most $(5 + 11\lceil\log_{\sqrt{3}}(\min\{\alpha,\gamma\})\rceil + 5\mathcal{I}_{\lceil\log_{\sqrt{3}}\gamma\rceil>\lceil\log_{\sqrt{3}}\alpha\rceil})s$ beads. We replicate the edges to get $\lceil k/2\rceil$ copies of each edge. Thus, the resulting network has at most $(5+$ $11\lceil\log_{\sqrt{3}}(\min\{\alpha,\gamma\})\rceil+5\mathcal{I}_{\lceil\log_{\sqrt{3}}\gamma\rceil>\lceil\log_{\sqrt{3}}\alpha\rceil})\lceil k/2\rceil s$ beads. We showed in Section 3.3.1 that this procedure maintains the k-edge connectivity for terminal nodes. Thus, a partial k-edge network with minimum number of beads contains at most $(5 +$ $11\lceil\log_{\sqrt{3}}(\min\{\alpha,\gamma\})\rceil + 5\mathcal{I}_{\lceil\log_{\sqrt{3}}\gamma\rceil>\lceil\log_{\sqrt{3}}\alpha\rceil})\lceil k/2\rceil s$ beads. □

[5]This charges the Steiner nodes on the DFS path only twice, thus it does not affect the approximation bound.

Theorem 4.4.5 states the main result of this section.

Theorem 4.4.5 *If an optimal network uses s Steiner nodes so that terminals distributed in the Euclidean plane are k-edge connected, Algorithm 3.3.1 (using Algorithm 4.4.1 for relay placement) forms a network with maximum of* $2(5+11\lceil\log_{\sqrt{3}}(\min\{\alpha,\gamma\})\rceil+5\mathcal{I}_{\lceil\log_{\sqrt{3}}\gamma\rceil>\lceil\log_{\sqrt{3}}\alpha\rceil})\lceil k/2\rceil s$ *beads and zero Steiner nodes, in which the terminals are k-edge connected.*

Proof: The algorithm of Khuller and Vishkin [41] is used to construct a k-edge connected subgraph of a graph, which is a 2-approximation. Thus, according to Lemma 4.4.4, the number of beads required by our algorithm is at most $2(5+11\lceil\log_{\sqrt{3}}(\min\{\alpha,\gamma\})\rceil+5\mathcal{I}_{\lceil\log_{\sqrt{3}}\gamma\rceil>\lceil\log_{\sqrt{3}}\alpha\rceil})\lceil k/2\rceil s$. The last step (sequential removal) removes beads from the network by allowing them to connect to all nodes within the transmission range, so the resulting network after sequential removal also has maximum of $2(5+11\lceil\log_{\sqrt{3}}(\min\{\alpha,\gamma\})\rceil+5\mathcal{I}_{\lceil\log_{\sqrt{3}}\gamma\rceil>\lceil\log_{\sqrt{3}}\alpha\rceil})\lceil k/2\rceil s$ relay nodes. □

The analysis also extends the results of full k-edge connectivity. The algorithm can be proved to be a $6(5+11\lceil\log_{\sqrt{3}}(\min\{\alpha,\gamma\})\rceil+5\mathcal{I}_{\lceil\log_{\sqrt{3}}\gamma\rceil>\lceil\log_{\sqrt{3}}\alpha\rceil})\lceil k/2\rceil^2$-approximation of the optimal.

4.4.3 Networks in Three Dimensional Euclidean Space

In this section, we consider the scenario where the heterogeneous network is in the three dimensional Euclidean space. The algorithms for vertex and edge connectivity are the same as for the Euclidean plane. The approximation guarantees are different. We start with the following lemma for vertex connectivity.

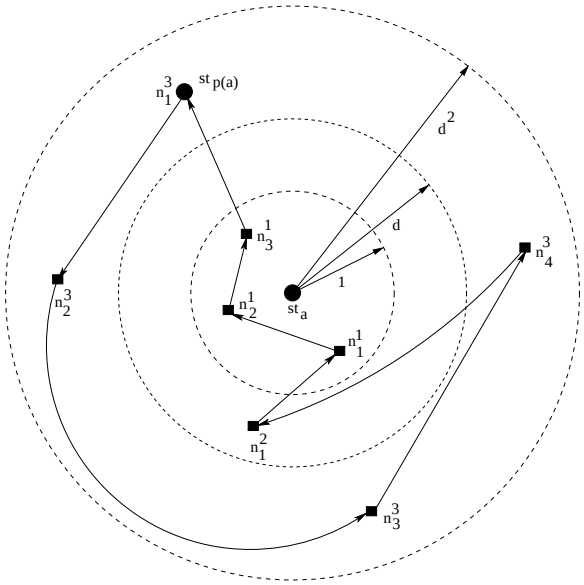


Figure 4.6: Order of DFS traversal around a Steiner node in a heterogeneous network in three dimensions

Lemma 4.4.6 *A partial 2-vertex connected heterogeneous network using the minimum number of beads contains at most* $(12 + 37\lceil \log_{1.5}(\min\{\alpha, \gamma\})\rceil + 13\mathcal{I}_{\lceil \log_{1.5}\gamma\rceil > \lceil \log_{1.5}\alpha\rceil})s$ *beads for terminals distributed in the three dimensional space using Euclidean metrics , where* s *is the minimum number of Steiner nodes needed.*

Proof: We construct a 2-vertex connected graph using a procedure similar to the one used for the Euclidean plane. The only difference is in the DFS traversal order around a Steiner node in the construction. We divide the area around a Steiner node into spherical rings, outer boundary of ring i being of radius d^{i-1}. However, rather than doing an anti-clockwise traversal inside each ring, we do a traversal based on distance in each ring. We fix $d = 1.5$ for our construction.

We use Algorithm 4.4.3 to construct the order of traversal of children of a Steiner node st_a. We start with the ring i_p that contains the parent node $st_{p(a)}$ (we start with the first ring when st_a is the root of the tree). We start with $st_{p(a)}$ (mark it, call it $n_1^{i_p}$), and connect it to the closest (using the Euclidean metric) child of st_a in ring i_p. We mark this child, call it $n_2^{i_p}$, and connect it to the unmarked child in ring i_p closest it. This procedure is continued to connect and number all children of st_a in the ring i_p. The last marked node in the ring is $n_{l_{i_p}}^{i_p}$. For all rings $i \neq i_p$, let n_1^i be the node with the largest distance from the node closest to it in its ring. Connect $n_{l_{i_p}}^{i_p}$ with node n_1^j, such that j is the next higher ring with at least one node. Connect all nodes in ring j starting at n_1^j, in the smallest distance order used for ring i_p, and connect the last node to the first node in the next higher ring. Once we reach ring i_M, we connect its last node to the first node in the next lower ring $j < i_p$. We then follow the above described procedure, now connecting the last node in each ring with the first node in next lower non-empty ring. When we reach ring i_m, we connect its last node to $st_{p(a)}$ to complete the cycle $\mathcal{C}_a$. The order of addition of nodes to the cycle is the order of their DFS traversal. Figure 4.6 shows an example DFS traversal order around a Steiner node.

Algorithm 4.4.3 DFS traversal order around a Steiner node

1: Partition the area around the Steiner node (st_a) into concentric rings, with the radius of outer boundary of ring i being d^{i-1}, $1 \leq i \leq \lceil \log_d(\min\{\alpha, \gamma\}) \rceil + 1$.

2: If $\lceil \log_d \gamma \rceil > \lceil \log_d \alpha \rceil$, add a concentric ring between circles of radius $d^{\lceil \log_d \alpha \rceil}$ and γ.

3: Let $st_{p(a)}$ be in ring i_p. Set $n_1^{i_p} = st_{p(a)}$, $i = i_p$. While $i < i_M$:

1. Mark n_1^i, start at $r = 1$. While there are unmarked nodes in ring i:

   - Connect n_r^i to the closest unmarked node. Mark this node, call it n_{r+1}^i. Set $r = r + 1$.

2. Let $j > i$ be the next higher ring with number of nodes $l_j > 0$. Let n_1^j be the node with largest nearest neighbor (in its ring) distance in ring j. Connect $n_{l_i}^i$ with n_1^j. Set $i = j$.

4: Set $i = i_M$. While $i \geq i_m$:

1. Mark n_1^i, start at $r = 1$. While there are unmarked nodes in ring i:

   - Connect n_r^i to the closest unmarked node. Mark this node, call it n_{r+1}^i. Set $r = r + 1$.

2. Let $j < i_p$ be the next lower ring with number of nodes $l_j > 0$. Let n_1^j be the node with largest nearest neighbor distance. Connect $n_{l_i}^i$ with n_1^j. If $i = i_m$, connect $n_{l_i}^i$ with $n_1^{i_p} = st_{p(a)}$ to complete the cycle $\mathcal{C}_a$. Set $i = j$.

5: The order in which the children get marked is the order of their DFS traversal.

We first consider the case $\gamma \geq 1$, i.e., $T_{relay} \geq T_{min}$. We first prove the following bound on the number of long edges in the cycle constructed in each ring (if we connected first and last nodes in the ring). Recall that $d = 1.5$ in our construction.

Lemma 4.4.7 *Let there be an arbitrary set of points $P = \{x_1, .., x_l\}, l > 0$ distributed in a ring between spheres of radius 1.5^i and 1.5^{i+1}, centered at a point x_0 in the three dimensional space using the Euclidean metric. We form a cycle C consisting of the points in P: Add x_1 to C. From the last point added to C, add an edge to the closest point in $P \backslash C$, add the new point to C, and repeat. Finally, add an edge to x_1 when $P = C$. The number of edges of length greater than 1.5^i is at most 36.*

Proof: Normalize the distances such that the radius of inner boundary of the ring is 1, and outer boundary is 1.5. Thus, we are interested in the number of edges in C of length greater than one (call these long edges). Rename the points in P such that x_i is the ith point being added to C. Start constructing C, and each time an edge of length greater than one is encountered, mark the end-point of the edge that was already in C, call it y_i ($i = 1$ for first such edge) and increase i by one. Let the set of marked points be P'. The number of points in P' is the number of long edges in C.

Remove the unmarked points of P ($x_i \notin P'$) from the ring. The ring now contains x_0 and the marked points $y_i \in P'$. When a point in P was added to P', it was more than unit distance from all the points not in C, thus the distance of $y_i, i \geq 1$ is greater than one from all points $y_j, j > i$. Thus, the distance of each y_i is greater than one from all points in $P' \backslash \{y_i\}$. Thus, the number of long edges

in C is equal to the number of points more than unit distance apart that we can have inside the spherical ring. If we have spheres of radius half centered at those points, they are non-overlapping. Since the width of the ring is 0.5, and the spheres are of radius 0.5, the maximum number of such points that can be placed so the spheres do not overlap is when they are placed on the outer surface of the ring. Since the width of the ring is 0.5, placing any other point in the ring will make its sphere overlap with the spheres around the points placed on the surface. Thus, the upper bound on total number of points more than unit distance apart is equal to the upper bound on the maximum number of such points that can be placed on the outer surface of the ring. Now we compute the maximum number of such outer ring surface points. Consider a point x on the surface. The sphere of radius half around x covers a certain area on the outer surface. The intersection of the half radius sphere with the outer ring surface forms a cone at the center of the ring, covering a certain solid angle. The half radius spheres are non-overlapping, thus dividing the total solid angle in the sphere of radius d by the solid angle of each cone gives an upper bound on the number of points that can be on the outer surface. The solid angle of a cone is given by $2\pi(1 - \cos\frac{a}{2})$, where a is the apex angle of a cone. Consider the projection of the cone onto a plane passing through the center of its base and perpendicular to the base. The result is a triangle with two sides of length $d = 1.5$. The apex angle can be computed, and is $a = 2\arccos(1 - \frac{1}{8d^2})$. Thus, the solid angle of the cone is $\frac{\pi}{4d^2}$. The solid angle of a sphere is 4π, thus the total number of such cones is at most $16d^2 = 36$. Therefore, the number of points in P', and thus the total number of long edges in the cycle C is at most 36. $\square$

We now list the type of ordered edge pairs around a Steiner node that we charge. As a recap from the proof for vertex connectivity in any metric space, we charge the following six types of edge pairs around st_a. Here, PS denotes the parent Steiner node of st_a, CS denotes a child Steiner node of st_a, and T denotes a child terminal node.

- **Type I**: PS-st_a-CS/T (Figure 4.2(a)). The distance between the parent Steiner node and the node at the other end is always greater than the transmission range of either end-point. If the node at the other end is a Steiner node, the distance is greater than the transmission range of either end-point as the tree ST_j is a BFS tree (nodes two levels away in the tree cannot have distance less than the transmission range of either end-point). If the other end is a terminal node, the distance is greater than the transmission range of either end-point due to the construction procedure of T_j (if the distance were less than the transmission range of either end-point, the terminal would be a child of the PS node).

- **Type II**: CS/T-st_a-PS (Figure 4.2(b)). The distance between the parent Steiner node and the node at the other end is always greater than the transmission range of either end-point, reason being the same as explained for Type I edge pair.

- **Type III**: CS-st_a-T (Figure 4.2(c)), if the distance between the end-nodes is greater than the transmission range of either end-point.

- **Type IV**: T-st_a-CS (Figure 4.2(d)), if the distance between the end-nodes is greater than the transmission range of either end-point.

- **Type V**: T-st_a-T (Figure 4.2(e)), if the distance between the end-nodes is greater than the transmission range of either end-point.

- **Type VI**: CS-st_a-CS (Figure 4.2(f)), if the distance between the end-nodes is greater than the transmission range of either end-point.

Note that the distance between the end-nodes is greater than the transmission range of either end-point in all these cases. For the edges inside each ring, the number of these edges is upper bounded by 36 for $d = 1.5$ according to Lemma 4.4.7. This is because the transmission range of each node in a ring is at least the radius of the inner sphere of the ring due to the bi-directionality of links (the nodes could connect to st_a). Also, each such edge charges for one bead since the two end-points could connect using one Steiner node st_a.

The first ring is a circle of radius one, and the transmission range of all nodes (including st_a, since we are looking at the case $\gamma \geq 1$) is at least one. Thus, according to Lemma 4.3.2, the number of long edges in the cycle in this ring is at most the MST number of the space, which is 12. If $\lceil \log_{1.5} \gamma \rceil > \lceil \log_{1.5} \alpha \rceil$, then the last ring (between circles of radius $1.5^{\lceil \log_{\sqrt{3}} \alpha \rceil}$ and γ) contains only Steiner nodes. Since the transmission range of all Steiner nodes is the same ($= \gamma$), according to Lemma 4.3.2, the number of long edges in the cycle in a circle of radius γ is at most 12. Thus, the number of long edges in the last ring is at most 12.

In cycle $\mathcal{C}_a$, we use all edges encountered in the cycle in each ring (that does not contain the parent node $st_{p(a)}$) except the longest edge (between $n^i_{l_i}$ and n^i_1). Since we do not use the longest edge, st_a is charged at most 35 times by edges in each ring $i > 1$ (36 times if the ring contains $st_{p(a)}$). For the first ring, st_a is charged at most 11 times if it does not contain $st_{p(a)}$, and 12 times if it contains $st_{p(a)}$. When we connect two nodes in different rings, that edge uses at most one bead (since they both were directly connected to st_a, and beads have the same transmission range as st_a). Thus, we charge st_a for one bead for each such cross-ring edge. The number of such beads is equal to the number of non-empty rings. Thus, we add one to the number of beads charged by each ring to take care of this case. If there is just one ring (the first circle of radius one, or only this ring is non-empty), $\mathcal{C}_a$ is just a cycle in that ring, and it charges st_a for at most 12 beads. Otherwise, if $st_{p(a)}$ is in the first ring, the first ring charges for 13 beads, and the other rings charge for 36 beads (12 for the last ring if $\lceil \log_{1.5} \gamma \rceil > \lceil \log_{1.5} \alpha \rceil$). Since there are at least one more non-empty ring in this case, we can change the charging to charge the other rings 37 beads (13 for the last ring if $\lceil \log_{1.5} \gamma \rceil > \lceil \log_{1.5} \alpha \rceil$), and the first ring 12 beads. If $st_{p(a)}$ is not in the first ring, the first ring charges for 12 beads, and other rings charge for at most 37 beads (13 for the last ring if $\lceil \log_{1.5} \gamma \rceil > \lceil \log_{1.5} \alpha \rceil$). Thus, each non-empty ring i charges st_a for at most 37 beads for $i > 1$ (13 for the last ring if $\lceil \log_{1.5} \gamma \rceil > \lceil \log_{1.5} \alpha \rceil$), and the first ring charges st_a for at most 12 beads.

Thus, st_a is charged for at most $(12 + 37\lceil \log_{1.5}(\min\{\alpha, \gamma\}) \rceil + 13\mathcal{I}_{\lceil \log_{1.5} \gamma \rceil > \lceil \log_{1.5} \alpha \rceil})s$ beads. Here, $\mathcal{I}_x$ is the indicator function, which is 1 if condition x is true, else it is 0. Thus, the final solution G_m uses at most $(12 + 37\lceil \log_{1.5}(\min\{\alpha, \gamma\}) \rceil +$

$13\mathcal{I}_{\lceil \log_{1.5} \gamma \rceil > \lceil \log_{1.5} \alpha \rceil})s$ beads.

We now consider the case $\gamma < 1$. In this case, the relay nodes have a transmission range smaller than all terminal nodes. Since the communication topology is bi-directional, all edges adjacent to the Steiner nodes in the optimal Steiner solution are at most of length γ. Therefore, all neighbors of a Steiner node are within a circle of radius γ, and all have a transmission range at least γ. Thus, according to Lemma 4.3.2, each Steiner node is charged for at most 12 beads. Thus, G_m uses at most $(12 + 37\lceil \log_{1.5}(\min\{\alpha, \gamma\}) \rceil + 13\mathcal{I}_{\lceil \log_{1.5} \gamma \rceil > \lceil \log_{1.5} \alpha \rceil})s$ beads. □

Theorem 4.4.8 states the approximation ratio of the algorithm.

Theorem 4.4.8 *If an optimal heterogeneous network uses s Steiner nodes so that terminals distributed in the three dimensional Euclidean space are 2-vertex connected, Algorithm 3.2.1 (using Algorithm 4.4.1 for relay placement) forms a network with maximum of* $2(12+37\lceil \log_{1.5}(\min\{\alpha, \gamma\}) \rceil + 13\mathcal{I}_{\lceil \log_{1.5} \gamma \rceil > \lceil \log_{1.5} \alpha \rceil})s$ *beads and zero Steiner nodes, in which the terminals are 2-vertex connected.*

Proof: The algorithm of Khuller and Raghavachari [40] is used to compute a 2-vertex connected subgraph of a graph, which is a 2-approximation. Thus, according to Lemma 4.4.6, the number of beads required by our algorithm is at most $2(12+37\lceil \log_{1.5}(\min\{\alpha, \gamma\}) \rceil + 13\mathcal{I}_{\lceil \log_{1.5} \gamma \rceil > \lceil \log_{1.5} \alpha \rceil})s$. The last step of Algorithm 3.2.1 (sequential removal step) removes beads from the network by allowing them to connect to all nodes within the transmission range, so the resulting network after sequential removal also has maximum of $2(12+37\lceil \log_{1.5}(\min\{\alpha, \gamma\}) \rceil + 13\mathcal{I}_{\lceil \log_{1.5} \gamma \rceil > \lceil \log_{1.5} \alpha \rceil})s$ relay nodes. □

For homogeneous networks ($\alpha = \gamma = 1$) in three dimensions, the result of Theorem 4.4.8 matches that of Theorem 4.3.3 since MST number $M = 12$ for the three dimensional Euclidean space. The analysis also extends the results of full 2-vertex connectivity. The algorithm can be proved to be a $12(12 + 37\lceil \log_{1.5}(\min\{\alpha, \gamma\})\rceil + 13\mathcal{I}_{\lceil \log_{1.5}\gamma\rceil > \lceil \log_{1.5}\alpha\rceil})$-approximation.

Similar arguments yields the following result for partial k-edge connectivity.

Theorem 4.4.9 *If an optimal network uses s Steiner nodes so that terminals distributed in the three dimensional Euclidean space are k-edge connected, Algorithm 3.3.1 (using Algorithm 4.4.1 for relay placement) forms a network with maximum of $2(12 + 37\lceil \log_{1.5}(\min\{\alpha, \gamma\})\rceil + 13\mathcal{I}_{\lceil \log_{1.5}\gamma\rceil > \lceil \log_{1.5}\alpha\rceil})\lceil k/2\rceil s$ beads and zero Steiner nodes, in which the terminals are k-edge connected.*

For homogeneous networks ($\alpha = \gamma = 1$) in three dimensions, the result of Theorem 4.4.9 matches that of Theorem 4.3.5 since MST number $M = 12$ for the three dimensional Euclidean space. The analysis also extends the results of full k-edge connectivity. The algorithm can be proved to be a $6(12 + 37\lceil \log_{1.5}(\min\{\alpha, \gamma\})\rceil + 13\mathcal{I}_{\lceil \log_{1.5}\gamma\rceil > \lceil \log_{1.5}\alpha\rceil})\lceil k/2\rceil^2$-approximation of the optimal.

Chapter 5

Movement of Relay Nodes for Topology Reconfiguration

In this chapter, we consider the problem of topology reconfiguration, as an application of the algorithms proposed in Chapter 3. We can use the algorithms proposed in Chapter 3 to add relays to establish a fault-tolerant topology between network nodes. But as the network nodes move, the desired fault-tolerance might be lost due to their limited transmission range. We propose algorithms to reconfigure the topology by moving the existing relay nodes a minimum amount and adding the minimum number of additional relay nodes.

The main application of these algorithms is in battlefield communication networks, which are networks of mobile nodes, communicating with each other using wireless links. The nodes in the battlefield refer to soldiers, army vehicles, UAVs, robots, etc. The network may also be a team of nodes engaged together to perform a large scale reconnaissance mission. The capability of a device to communicate with all nodes (using single or multiple hops) is very critical for the collaborative missions to succeed. In these applications, the nodes are in hostile conditions, and the nodes or communication links between nodes may fail. Thus, it is critical for the communication network between the nodes to be connected even after a few failures. Since the nodes are mobile, it is necessary to re-configure the communication topology as it changes substantially.

We assume the network nodes move at a slow time scale. Thus, once their locations have changed significantly, we would like to re-establish the desired topology using minimum number of relays. Since some relays already exist in the network (used in the topology on previous terminal locations), the secondary objective is to move the existing relay nodes a minimum distance to the new relay positions so that the topology is constructed quickly.

We first present the algorithms, followed by the computational complexity and simulation results.

5.1 Placement and Movement Algorithms

We first describe the framework followed by the proposed algorithms to minimize the distance travelled by existing relays as a secondary objective. Table 5.1 lists the notation used in the algorithms. We consider the objective of achieving k-edge connectivity between the terminals, thus we will use Algorithm 3.3.1 in the algorithms described next. To achieve k-vertex connectivity, Algorithm 3.2.1 can be used instead.

5.1.1 Framework for Minimizing Distance

We use the current positions of the existing relay nodes along with the number of new relays required to form an edge in the edge weight function c_e used in Algorithm 3.3.1. Algorithm 3.3.1 is executed to compute an approximately minimum weight k-connected subgraph, followed by sequential removal of new relay nodes.

Table 5.1: Notation

Symbol	**Definition**
T	Set of terminal nodes
N	Number of terminal nodes
k	Desired level of edge or vertex connectivity
R^o	Set of existing relay nodes
G_o	Topology on existing relays and terminals at old locations
$R^o_{i,j}$	Set of relays associated with terminals i, j in G_o
$r_{i,j}$	Number of relays associated with terminals i, j in G_o
R^o_1	Set of vertices, one vertex per $R^o_{i,j} \forall i, j \in T$
G_c	Complete graph on terminals T at their new locations
E_c	Set of edges of G_c
R^p	Set of potential relay nodes on edges in G_c
M_e	Number of matched relay nodes on edge e in G_c
$R^p_{i,j}$	Set of relays associated with terminals i, j in G_c
R^p_1	Set of vertices, one vertex per $R^p_{i,j} \forall i, j \in T$
G_n	Topology on relays and terminals at new locations
R^n	Set of new relay nodes in G_n

We then use maximum weight matching (Lovász and Plummer [54]) to move the existing relay nodes to the new relay positions such that the total movement is minimized. If more relay nodes are needed, they are added to the network. The framework is given in Algorithm 5.1.1. The initial weighting favors certain edges to be included in the k-connected subgraph by giving them a lower weight. The weight of each edge is reduced from the number of relays needed if existing relays can be moved to the relay positions on that edge. The algorithms we propose differ in the way they assign these edge weights, and thus lead to different total relay movement. Steps 2, 3 and 4 are the same for all algorithms.

Algorithm 5.1.1 Framework for relay and distance minimizing algorithms

1: Execute Algorithm 3.3.1, with edge weights c_e depending on the positions of existing relays and the number of relays required to form edge e.

2: Move the existing relay nodes to the new relay positions (at the output of Step 1) according to the following:

- Construct a bipartite graph $G_M = (V_M, E_M)$, where $V_M = R^o \cup R^n$. E_M consists of edges between each pair of vertices $(v^o \in R^o, v^n \in R^n)$.
- Set weight w_e of each edge $e \in E_M$ according to the distance between the corresponding actual and new relay positions.
- Let $W = \max_{e \in E_M} w_e$. Set $w_e = W - w_e + 1, \forall e \in E_M$.
- Perform maximum weight matching (Lovász and Plummer [54]) on this graph to get a matching. A matching is a subgraph of pairs of vertices connected to each other, such that no vertex is connected to more than one vertex.

3: Move each existing relay node to the new relay position it is mapped to in the solution. This minimizes the total movement of relay nodes.

4: Add new relays if there are unmatched new positions.

5.1.2 Minimum Relays Algorithm (MRA)

Minimum Relays Algorithm (MRA) uses the same weights c_e as in Equation 3.1, i.e., the number of relays needed to form an edge. Thus, the algorithm does not consider the existing relay locations in computation of the k-connected subgraph.

5.1.3 Individual Matching based Algorithm (IMA)

Individual Matching based Algorithm (IMA) considers the existing relay locations in the computation of weights c_e of edges in E_c. The computation of edge weights is described in Algorithm 5.1.2. The algorithm matches the existing relay locations with the potential relay locations in R^p, such that the existing relays move a minimum distance. Then, the algorithm weights each edge in E_c by the number of unmatched relays on the edge.

Algorithm 5.1.2 Computation of initial edge weights in IMA

1: Construct a complete graph $G_c = (T, E_c)$ by forming edges between all terminals.

2: Mark the positions of relays needed to form each edge in E_c. The number of relays needed to form an edge e of length $|e|$ is $\lceil |e| \rceil - 1$. In a network in the first quadrant of a Euclidean plane, the position (x, y) of relays for edge e of length greater than one between vertices i and j can be computed as:

$$\begin{aligned} x_m &= x_i + (x_j - x_i)m/\lceil |e| \rceil, m \in \{1, \cdot\cdot, \lceil |e| \rceil - 1\} \\ y_m &= y_i + (y_j - y_i)m/\lceil |e| \rceil, m \in \{1, \cdot\cdot, \lceil |e| \rceil - 1\} \end{aligned} \tag{5.1}$$

3: Construct a bipartite graph $G_M = (V_M, E_M)$, where $V_M = R^o \cup R^p$. E_M consists of edges between each pair of vertices $(v^o \in R^o, v^p \in R^p)$.

4: Find a matching by inverting the weights and computing maximum weight matching as in G_M as in Step 2 of Algorithm 5.1.1.

5: For each edge $e \in E_c$ with M_e matched new relay positions, define weight function c_e as:

$$c_e = \lceil |e| \rceil - 1 - M_e, \forall e \in E_c \tag{5.2}$$

5.1.4 Same Terminal Pair Algorithm (STPA)

Same Terminal Pair Algorithm (STPA) tries to associate the existing relays to the same terminal pair they were associated with before the terminals moved. The weight function for initial weighting c_e is as defined in Equation 5.3, where the symbols are as defined in Table 5.1. $e_{i,j}$ represents the edge between terminals i and j in G_c. The algorithm will favor the formation of the same edges as in the topology G_o by assigning them a lower weight than the number of relays required. The algorithm is expected to work well if the terminal nodes do not move too far.

$$W(e_{i,j}) = \max\{\lceil |e_{i,j}| \rceil - 1 - r_{i,j}, 0\}, \forall e_{i,j} \in E_c \tag{5.3}$$

5.1.5 Group Matching based Algorithm (GMA)

Group Matching based Algorithm (GMA) is a hybrid of IMA and STPA. The algorithm maps each set of existing relays associated with a single pair of terminals in G_o to potential relay positions, all of which are associated with a single terminal pair in G_c. However, unlike STPA, the two terminal pairs can be different. We do not consider sets with zero existing or potential relays. The algorithm uses minimum weight matching as in Step 2 of Algorithm 5.1.1 to find minimum distances between each such pair of sets of existing and potential relays. Then it uses the distances as weights and uses matching again to map the sets of existing relays (R_1^o) to sets of potential relays (R_1^p) such that minimum distance is travelled. Then, the algorithm weights each edge as the number of unmatched relays among the relays needed on each edge in E_c. Algorithm 5.1.3 describes the algorithm in more detail.

Algorithm 5.1.3 Computation of initial edge weights in GMA

1: Construct a complete graph $G_c = (T, E_c)$ on terminals.

2: Mark the positions of potential relays needed to form each edge in E_c, as in Step 2 of Algorithm 5.1.2.

3: Make a vertex corresponding to $R^o_{i,j}$ for all pairs of terminals (i, j) which have at least one relay associated with them. Make a vertex corresponding to $R^p_{i,j}$ for all pairs of terminals (i, j) which are more than distance one apart at new positions. Call the sets of vertices as R^o_1, R^p_1.

4: For each pair of vertices $(v^o \in R^o_1, v^p \in R^p_1)$:

- Construct a bipartite graph on the existing and potential relay nodes in v^o and v^p with an edge between each (existing, potential) relay pair.
- Find a matching in G_M as in Step 2 of Algorithm 5.1.1.
- Assign a weight to the pair of this set of existing relays and set of potential relays as the sum of distances between matched vertices.

5: Construct a bipartite graph $G_M = (V_M, E_M)$, where $V_M = R^o_1 \cup R^p_1$. E_M consists of edges between each pair of vertices $(v^o \in R^o_1, v^p \in R^p_1)$.

6: Weight each edge in E_M as the weight assigned to that pair of vertices in Step 4 of this algorithm.

7: Find a matching in G_M as in Step 2 of Algorithm 5.1.1.

8: For each edge $e \in E_c$, define weight function c_e as:

$$c_e = \lceil |e| \rceil - 1 - M_e, \forall e \in E_c \tag{5.4}$$

5.1.6 Enhanced Group Matching based Algorithm (EGMA)

Enhanced Group Matching based Algorithm (EGMA) is a variation of GMA that takes into account the difference between the number of relays in sets $R^o_{i,j}$ and $R^p_{i,j}$ while assigning them weights for matching in Step 6 of Algorithm 5.1.3. Rather than keeping the weight as the minimum distance needed to move the existing relays in $R^o_{i1,j1}$ to $R^p_{i2,j2}$ for all $(i1, j1), (i2, j2)$, the algorithm multiplies the weight by $\max\{|R^o_{i1,j1}|/R^p_{i2,j2}, |R^p_{i2,j2}|/R^o_{i1,j1}\}$. Here, $|R^o_{i,j}|$ ($|R^p_{i,j}|$) denotes the number of relays (> 0) in the set $R^o_{i,j}$ ($R^p_{i,j}$). Thus, the algorithm favors matching two sets which have less disparity in the number of relays.

5.1.7 Computational Complexity

We discuss the computational complexity for the objective of maintaining k-edge connectivity between the terminals. Analysis for vertex connectivity is similar. Let the number of terminal nodes be N, and the network area be a square of length L. We showed in Section 3.3.2 that Algorithm 3.3.1 takes $O(k^4N^3L^3)$ time. Maximum weight matching on a bipartite graph of n vertices takes $O(n^{2.5})$ time. Thus, Step 2 of Algorithm 5.1.1 takes $O((kNL)^{2.5})$ time. The algorithms differ in the time required for computation of weight function c_e for Step 1 of the framework. If that time is $O(f(k, N, L))$ for an algorithm, the total time required is $O(f(k, N, L) + k^4N^3L^3 + (kNL)^{2.5}) = O(f(k, N, L) + k^4N^3L^3)$. Thus, for MRA and STPA, the total time is $O(k^4N^3L^3)$; for IMA, the total time is $O((kNL + N^2L)^{2.5} + k^4N^3L^3)$; and for GMA and EGMA, the total time is $O(kN^3L^{4.5} + (kNL + N^2L)^{2.5} + k^4N^3L^3)$.

5.2 Simulation Results and Discussion

The networks simulated were in a square region of side length 10km. The nodes were assumed to have a transmission range of 1km. The mobility model used was random waypoint, in which the nodes move to a point within a circle of a certain radius (R) around their current location randomly. Each node independently picks a distance between 0 and R uniformly randomly, an angle from 0 to 2π uniformly randomly, and moves to a point at that distance and that angle. Whenever any coordinate of the point to move to is outside the square network region, the point is taken to be the boundary of the network region in that coordinate.

We implemented the algorithms for finding a k-edge connected network. The initial node locations were chosen independently randomly with a uniform distribution. A k-edge connected topology was formed using Algorithm 3.3.1, whose objective is to minimize the number of relays. The relays were placed at the required positions and the terminals were moved using the mobility model described above. Then, the algorithms were run to find the new relay positions to construct the desired topology. The algorithms were compared for the number of relays they require and the movement of the existing relays. We study the performance of the algorithms for different values of R, and different number of terminal nodes (N) in the network. The matching algorithm used is an implementation of Gabow's N-cubed weighted matching algorithm, Gabow [29].

5.2.1 Variation with Movement of Terminals

We first fixed the number of terminal nodes (N) at 20, and varied the amount of movement of the terminal nodes. The network was formed with 10 different randomly generated node locations, and for each set of node locations, 10 different sets of node movements were generated. Thus, the algorithms were run a total of 100 times. The first set of results are for achieving 2-edge connectivity among the terminal nodes. The maximum movement allowed (R) for each node was varied from 500m to 5km. Figure 5.1 shows the average total distance moved by existing relays in MRA, IMA, STPA, GMA and EGMA. Figure 5.2 shows the average total number of relays required in the new topology formed by MRA, IMA, STPA, GMA and EGMA. STPA and EGMA work better than GMA and IMA, which in turn work better than MRA in terms of the total relay movement. MRA saves only a few relays compared to the other methods. This savings is offset by the larger movement distance it requires. STPA works slightly better than EGMA when the terminal nodes do not move much, whereas EGMA works much better than STPA as the terminal movement increases. This is expected as STPA tries to move the existing relays to connect the same terminal pair they were associated with before, and thus works well for small terminal node movements. For large movements, EGMA works better as the terminals may move quite far from their original locations and thus it may be better to move existing relays to connect terminal nodes other than the ones they were associated with before.

Figures 5.3 and 5.4 show the ratio between EGMA and MRA of distance

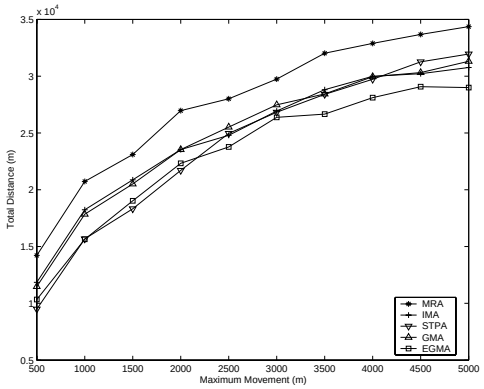


Figure 5.1: Total relay movement, varying R, $N = 20$, $k = 2$

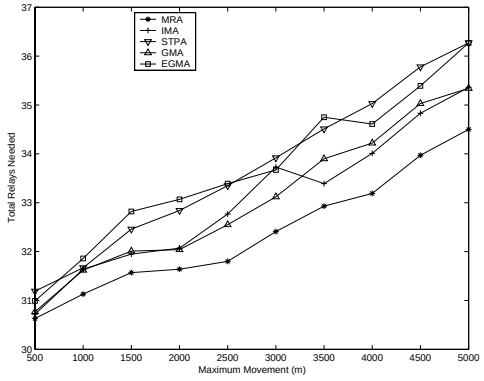


Figure 5.2: Number of relays needed, varying R, $N = 20$, $k = 2$

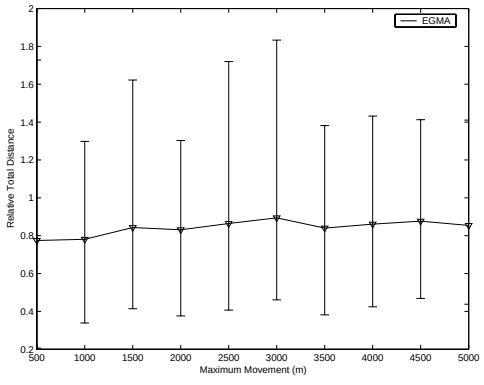


Figure 5.3: Total relative relay movement in EGMA, varying R, $N = 20$, $k = 2$

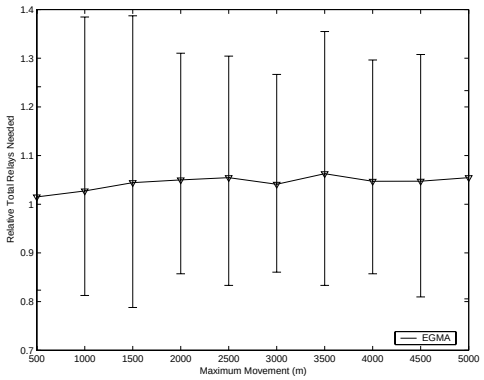


Figure 5.4: Relative number of relays in EGMA, varying R, $N = 20$, $k = 2$

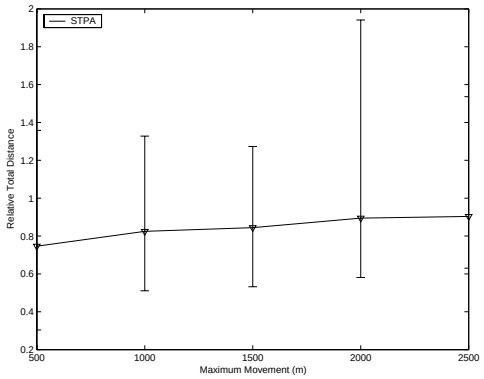


Figure 5.5: Total relative relay movement in STPA, varying R, $N = 20$, $k = 3$

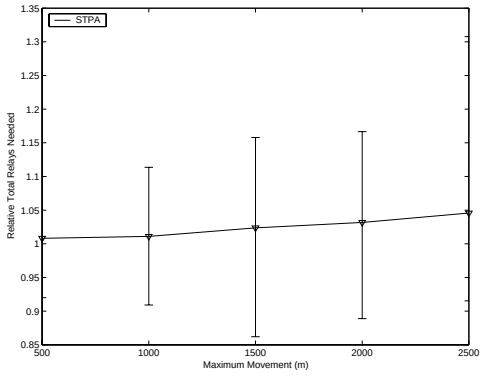


Figure 5.6: Relative number of relays in STPA, varying R, $N = 20$, $k = 3$

moved by relays and of total number of relays required. EGMA leads to a savings of 20% in the distance travelled by existing relays on an average, for all values of R. Also, the number of relays required by EGMA is almost the same as in MRA on an average. It is worthwhile to note than in some instances MRA may require more relays because the algorithm for finding a minimum-relay k-edge connected topology is not optimal. Thus, other algorithms may use less relays than MRA in some instances, though that is not true on an average (as the results show).

We also simulated MRA and STPA (STPA works as well as EGMA for the values of R used) for the objective of constructing a 3-edge connected topology. The number of simulations were the same as before. Figures 5.5 and 5.6 show the ratio between STPA and MRA of distance moved by relays and of total number of relays required. Compared to MRA, STPA leads to a savings of 20-25% in total relay movement on an average for varying values of R for $k = 3$ as well. Also, the number of relays used is almost the same as in MRA on an average.

5.2.2 Variation with Number of Terminals

We now study the variation with respect to the number of terminal nodes in the network. The simulation set up is the same as before, and they are done on 10 sets of network locations with 10 sets of random movements. The objective is to construct a 2-edge connected topology among the terminal nodes. The number of terminals is varied from 10 to 50. The maximum movement allowed (R) for each node is fixed at 1500m. Figures 5.7 and 5.8 show the ratio between STPA and MRA

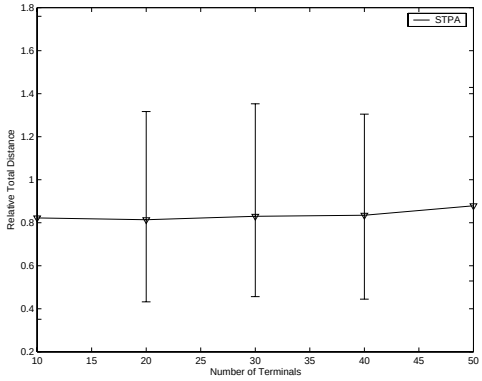


Figure 5.7: Total relative relay movement in STPA, varying N, $R = 1500$, $k = 2$

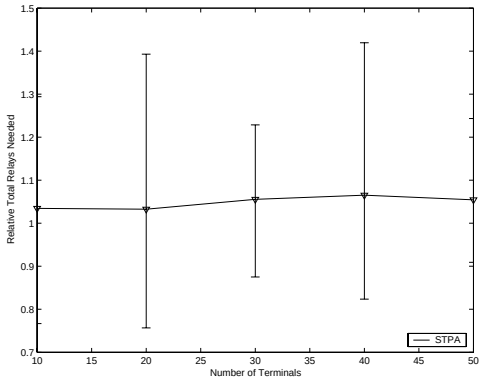


Figure 5.8: Relative number of relays in STPA, varying N, $R = 1500$, $k = 2$

of distance moved by relays and of total number of relays required. STPA leads to about 15-20% less movement of existing relays than MRA, and uses almost the same number of relays on an average for all values of N considered.

Thus, STPA works much better than MRA for a wide range of number of terminals and amount of movement of terminal nodes (R). If the amount of terminal movement is very high, then EGMA performs better than STPA (and all other algorithms), and thus should be used in those cases.

Chapter 6

Joint Traffic-Oblivious Routing and Power Control

In this chapter, we consider another aspect of robustness of a wireless network: robustness against traffic variation. We assume the traffic in the network is variable, and is constrained by a polytope defined by the limits on traffic between each ingress-egress pair and on traffic originating/sinking at each ingress/egress node. We exploit the cross-layer characteristics of wireless networks, and provide joint routing and power control policies. The output of our algorithms is a fixed routing and power control policy, that is schedulable for all traffic within the polytope. We further enhance our algorithms to optimize the link transmission powers according to the network traffic.

We first describe the network model, followed by the problem statement, algorithms and simulation results.

6.1 Network Model

We model the given wireless network as a graph $G = (V, E)$, where V represents the set of nodes in the network, and E represents the set of (uni-directional) edges between pairs of nodes. An edge exists from node i to node j if j is in the receiving range of i, and hence i can transmit directly to j.

We explain the notation used in this chapter, listed in Table 6.1. For each

node $v \in V$, O_v and I_v represent the sets of outgoing and incoming edges at v, respectively. Let $N_v := I_v \cup O_v$ be the set of all edges incident at v. The end-to-end traffic or flow between each pair of nodes is called a commodity, and the set of all commodities in the network is denoted by C. For each commodity, the ingress node is called its source, and the egress node is called its destination. S represents the set of sources, and D represents the set of destinations in the network. The traffic demand for commodity c is represented by t_c, and its source and destination nodes are represented by s_c and d_c, respectively. We assume that the exact value of t_c is not known, although the set $\{t_c, c \in C\}$ (called a traffic matrix) is constrained by a polytope $\mathcal{T}$. The polytope $\mathcal{T}$ is defined as follows:

$$\mathcal{T} = \{t_c : \sum_{c|s_c=v} t_c \leq \lambda_v \ \forall v \in S, \tag{6.1a}$$

$$\sum_{c|d_c=v} t_c \leq \eta_v \ \forall v \in D, \tag{6.1b}$$

$$t_c \leq \omega_c \ \forall c \in C\} \tag{6.1c}$$

Equations 6.1a and 6.1b represent constraints on the total traffic originating at a source node and the total traffic sinking at a destination node, respectively. These constraints are called hose model constraints, Duffield et al. [25], and limit total incoming and outgoing traffic of source and destination nodes, respectively. Equation 6.1c represents pipe model constraints, which limit the traffic rate between source-destination pairs.

The fraction of traffic for commodity c traversing edge e is denoted by $f_c(e)$, and f denotes the routing in the network. The rate of transmission on link e is represented by $R(e)$ (R denotes the set of rates $\{R(e)|e \in E\}$), and the transmission

Table 6.1: Notation

Symbol	**Definition**
V/E	Set of nodes/edges
O_v/I_v	Set of outgoing/incoming edges at node v
N_v	Set of all edges incident at node v
$C/S/D$	Set of commodities/sources/destinations
$t_c/s_c/d_c$	Traffic/source/destination for commodity c
$\mathcal{T}$	Traffic matrix polytope
λ_v, η_v	Hose model bounds on traffic demands
ω_c	Pipe model bounds on traffic demands
f	Routing (set of flows)
$f_c(e)$	Fractional traffic of commodity c traversing edge e
$P(e)$	Transmission power for edge e
$R(e)$	Rate of edge e
R	Set of rates, $\{R(e) \mid e \in E\}$
B	Schedulability constant (2/3 for sufficiency, 1 for necessity)

power is denoted by $P(e)$. For the additive white Gaussian noise (AWGN) channel, assuming no interference from other simultaneous transmissions, the achievable link rate in bits/second for link e, $R(e)$, is upper bounded by (Cover and Thomas [21]):

$$R(e) = W \log(1 + \frac{P(e)\sigma(e)}{N_0 W}), \tag{6.2}$$

where W is the link bandwidth, $\sigma(e)$ the channel gain (that takes attenuation into account as well) and N_0 the noise spectrum density. We use the upper bound as the achievable link rate.

A routing f is said to be valid if it satisfies the flow conservation laws given below,

$$\sum_{e \in O_{s_c}} f_c(e) = 1 \quad \forall c \in C \tag{6.3a}$$

$$\sum_{e \in O_v} f_c(e) = \sum_{e \in I_v} f_c(e) \; \forall v \in V \backslash \{s_c, d_c\}, \; \forall c \in C \tag{6.3b}$$

Equation 6.3a states that total fractional traffic originating at the source should be equal to one for each commodity. Equation 6.3b states that total fractional traffic entering an intermediate node should be equal to total fractional traffic exiting it, for each commodity.

6.1.1 Interference Model

We consider the "free of secondary interference" model, Hajek and Sasaki [31], Kodialam and Nandagopal [43], Bhatia and Kodialam [12], Li et al. [49], Wieselthier et al. [76]. In this model, transmissions on two links do not interfere with each other as long as no end point of the links is common. The scheduling constraint

in this model is that a node cannot communicate (either transmit or receive) with more than one node at any instance of time. Such an interference model is applicable to networks employing TDMA/CDMA MAC, in which different nodes use different spread spectrum sequences for communication, and thus do not interfere with each other. Such an interference model is also applicable to networks that use highly directional antennas. Use of highly directional antennas can provide higher throughput than omni-directional antennas (due to reduction in interference).

We assume that the time is divided into synchronized slots for communication between nodes, and all nodes have knowledge about the slot boundaries. The scheduling constraint for this model can be expressed by the following equation

$$\sum_{e \in N_v} \sum_{c \in C} \frac{f_c(e)}{R(e)} t_c \leq B \ \forall v \in V \tag{6.4}$$

Here, $f_c(e)t_c/R(e)$ is the fraction of time needed to transmit the traffic for commodity c on edge e at the data rate $R(e)$. Kodialam and Nandagopal [43] proved that for any set of routing and rates (f, R), the set is schedulable under the interference model if for each node v, the sum of $f_c(e)t_c/R(e)$ over all commodities and over all links e adjacent to v is less than 2/3, and it is not schedulable if the sum is greater than 1. Thus, setting $B = 1$ in Equation 6.4 defines the necessary constraint, and setting $B = 2/3$ defines the sufficient constraint for feasibility.

The average transmission power at each link e is,

$$\sum_{c \in C} \frac{f_c(e)t_c}{R(e)} \frac{N_0 W}{\sigma(e)} (2^{\frac{R(e)}{W}} - 1) \tag{6.5}$$

This expression is the sum (over commodities) of link transmission power needed while transmitting (derived from Equation 6.2) multiplied with the fraction of time

it transmits for each commodity.

6.2 Joint Routing and Power Assignment

6.2.1 Problem Statement

We now formally state the problem we study in this chapter: Given a network graph $G = (V, E)$, find a routing and power assignment that minimizes the maximum (over all traffic matrices t in $\mathcal{T}$) total transmission power used in the network. The optimization formulation representing this problem is

$$\min_{f,R} \max_{t_c \in \mathcal{T}} \sum_{e \in E} \sum_{c \in C} \frac{f_c(e) t_c}{R(e)} \frac{N_0 W}{\sigma(e)} \left(2^{\frac{R(e)}{W}} - 1\right) \tag{6.6a}$$

s.t.

$$\sum_{e \in O_{s_c}} f_c(e) = 1 \ \ \forall\ c \in C \tag{6.6b}$$

$$\sum_{e \in O_v} f_c(e) = \sum_{e \in I_v} f_c(e)\ \forall\ v \in V \backslash \{s_c, d_c\},\ \forall\ c \in C \tag{6.6c}$$

$$\sum_{e \in N_v} \sum_{c \in C} \frac{f_c(e)}{R(e)} t_c \leq B \ \ \forall v \in V, \ \ \forall\ t \in \mathcal{T} \tag{6.6d}$$

$$0 \leq f_c(e) \leq 1,\ 0 \leq R(e) \ \ \forall\ c \in C, \ \ \forall\ e \in E. \tag{6.6e}$$

The constraints on f and R are that f should be a valid routing (defined by the conditions of Equation 6.3) and (f, R) should admit a valid scheduling (defined by the conditions of Equation 6.4) for all traffic matrices in $\mathcal{T}$. The flow constraints on routing f are given in Equations 6.6b and 6.6c. Equation 6.6d represents the scheduling constraints on the routing and rates. The variables f and R are bounded as in Equation 6.6e.

The problem is a non-linear (neither convex nor concave) min-max problem,

with maximization over an infinite set of traffic matrices. The problem has infinite non-linear constraints of Equation 6.6d, since there are infinite traffic matrices $t = \{t_c, c \in C\} \in \mathcal{T}$.

If we introduce a new variable P, the problem could equivalently be written by replacing the objective function with $\min_{f,R} P$, and adding the constraint:

$$\sum_{e \in E} \sum_{c \in C} \frac{f_c(e) t_c}{R(e)} \frac{N_0 W}{\sigma(e)} \left(2^{\frac{R(e)}{W}} - 1\right) \leq P \quad \forall\, t \in \mathcal{T} \tag{6.7}$$

In this way, the total transmission power is always less than or equal to P, and minimizing P would be equivalent to minimizing the maximum power. Thus, the min-max problem is equivalent to a minimization problem with two sets of infinite constraints, specified in Equations 6.6d and 6.7, respectively.

6.2.2 Replacement of Infinite Constraints

We first replace the infinite constraints induced by the maximization term of the objective function with a finite number of constraints by adding some auxiliary variables. To this end, we write an oracle that, for a given solution (f, R), computes the maximum value of the objective function of Equation 6.6a over all traffic matrices in $\mathcal{T}$. The oracle is a linear program, as written in Equations 6.8a-6.8d. The variables of the program are $t = \{t_c, c \in C\}$, and the constraints are the hose and

pipe model constraints that define $\mathcal{T}$.

$$\max_{t} \sum_{e \in E} \sum_{c \in C} \frac{f_c(e)}{R(e)} t_c \frac{N_0 W}{\sigma(e)} \left(2^{\frac{R(e)}{W}} - 1\right) \tag{6.8a}$$

s.t.

$$\sum_{c|s_c=v} t_c \leq \lambda_v \ \forall v \in S \tag{6.8b}$$

$$\sum_{c|d_c=v} t_c \leq \eta_v \ \forall v \in D \tag{6.8c}$$

$$t_c \leq \omega_c \ \forall c \in C \tag{6.8d}$$

The problem of Equations 6.8a-6.8d is a maximization problem, and thus it is hard to replace the objective function of Equation 6.6a, which is a minimization problem, with it. Let the variables α_v, β_v and γ_c be dual variables for the constraints in Equations 6.8b, 6.8c and 6.8d respectively. We write the dual of Equations 6.8a-6.8d:

$$\min_{\alpha,\beta,\gamma} \sum_{v \in S} \lambda_v \alpha_v + \sum_{v \in D} \eta_v \beta_v + \sum_{c \in C} \omega_c \gamma_c \tag{6.9a}$$

s.t.

$$\sum_{e \in E} \frac{f_c(e)}{R(e)} \frac{N_0 W}{\sigma(e)} \left(2^{\frac{R(e)}{W}} - 1\right) \leq \alpha_{s_c} + \beta_{d_c} + \gamma_c \ \forall c \in C \tag{6.9b}$$

$$0 \leq \alpha_v \forall v \in S, 0 \leq \beta_v \forall v \in D, 0 \leq \gamma_c \forall c \in C \tag{6.9c}$$

The duality gap is zero (strong duality) for linear programs, Boyd and Vandenberghe [13]. Hence, the objectives of Equations 6.8a and 6.9a are equal at optimality. Thus, for any (f, R), the max term in the objective function of Equation 6.6a can be replaced by the objective function of Equation 6.9a, subject to the additional constraints of Equations 6.9b and 6.9c. Therefore, the infinite constraints of the objective function of Equation 6.6a have been removed.

There still are infinite constraints, which are the scheduling constraints of Equation 6.6d. We follow the same trick as before, and replace the infinite constraints using auxiliary variables and a finite number of constraints. For a fixed (f, R), we write an oracle, for each node $v_1 \in V$, that maximizes the total fraction of transmissions and receptions at that node,

$$\max_{t} \sum_{e \in N_{v_1}} \sum_{c \in C} \frac{f_c(e)}{R(e)} t_c \tag{6.10a}$$

s.t.

$$\sum_{c|s_c=v} t_c \leq \lambda_v \ \forall v \in S \tag{6.10b}$$

$$\sum_{c|d_c=v} t_c \leq \eta_v \ \forall v \in D \tag{6.10c}$$

$$t_c \leq \omega_c \ \forall c \in C \tag{6.10d}$$

Here, the variables are the traffic demands $t = \{t_c : c \in C\}$, and the constraints are the hose and pipe model constraints that define the traffic matrix polytope $\mathcal{T}$. The solution (f, R) is feasible if the objective of the oracle is less than B for all nodes v. We again write the dual of this oracle to turn the maximization into a minimization problem:

$$\min_{\alpha^{v_1}, \beta^{v_1}, \gamma^{v_1}} \sum_{v \in S} \lambda_v \alpha_v^{v_1} + \sum_{v \in D} \eta_v \beta_v^{v_1} + \sum_{c \in C} \omega_c \gamma_c^{v_1}$$

s.t.

$$\sum_{e \in N_{v_1}} \frac{f_c(e)}{R(e)} \leq \alpha_{s_c}^{v_1} + \beta_{d_c}^{v_1} + \gamma_c^{v_1} \ \forall c \in C$$

$$0 \leq \alpha_v^{v_1} \ \forall v \in S, \ 0 \leq \beta_v^{v_1} \ \forall v \in D$$

$$0 \leq \gamma_c^{v_1} \ \forall c \in C \tag{6.11}$$

The variables $\alpha_v^{v_1}, \beta_v^{v_1}$ and $\gamma_c^{v_1}$ are dual variables for the constraints of Equa-

tions 6.10b, 6.10c and 6.10d respectively. Since strong duality holds for linear programs, the objectives of Equations 6.10 and 6.11 are the same at their respective solutions. Since the problem of Equation 6.11 is a minimization problem, we replace the infinite constraints of Equation 6.6d with the following constraints for all $v_1 \in V$:

$$\begin{aligned}
&\sum_{v \in S} \lambda_v \alpha_v^{v_1} + \sum_{v \in D} \eta_v \beta_v^{v_1} + \sum_{c \in C} \omega_c \gamma_c^{v_1} \leq B \\
&\sum_{e \in N_{v_1}} \frac{f_c(e)}{R(e)} \leq \alpha_{s_c}^{v_1} + \beta_{d_c}^{v_1} + \gamma_c^{v_1} \ \forall c \in C \\
&0 \leq \alpha_v^{v_1} \ \forall v \in S, \ 0 \leq \beta_v^{v_1} \ \forall v \in D \\
&0 \leq \gamma_c^{v_1} \ \forall c \in C
\end{aligned} \tag{6.12}$$

We now consider the term $2^{R(e)/W}$. The exponent $R(e)/W$ represents the spectral efficiency of the wireless links. In most practical systems, the term is less than one. Thus, we replace $2^{R(e)/W}$ by its Taylor series expansion, and take the first four terms given below (Bhatia and Kodialam [12]):

$$2^{\frac{R(e)}{W}} \simeq 1 + \frac{\ln 2}{W} R(e) + \frac{\ln^2 2}{2W^2} R^2(e) + \frac{\ln^3 2}{6W^3} R^3(e) \tag{6.13}$$

We now replace infinite constraints of Equation 6.6d with equivalent finite set of constraints given in Equation 6.12. Further, we use the Taylor series expansion of Equation 6.13 to get an approximate optimization problem. We also replace infinite constraints in the objective function of Equation 6.6a with objective function in Equation 6.9a and constraints of Equations 6.9b-6.9c. The optimization problem

becomes:

$$\min_{f,R,\alpha,\beta,\gamma,\alpha^v,\beta^v,\gamma^v} \sum_{v\in S}\lambda_v\alpha_v + \sum_{v\in D}\eta_v\beta_v + \sum_{c\in C}\omega_c\gamma_c$$

s.t.

$$\sum_{e\in O_{s_c}} f_c(e) = 1 \ \ \forall\ c\in C$$

$$\sum_{e\in O_v} f_c(e) = \sum_{e\in I_v} f_c(e) \ \ \forall\ v\in V\backslash\{s_c,d_c\},\ \ \forall\ c\in C$$

$$\sum_{e\in E} f_c(e)\frac{N_0\ln 2}{\sigma(e)}\left(1+\frac{\ln 2}{2W}R(e)+\frac{\ln^2 2}{6W^2}R^2(e)\right) \le \alpha_{s_c}+\beta_{d_c}+\gamma_c\ \forall c\in C$$

$$\sum_{v\in S}\lambda_v\alpha_v^{v_1} + \sum_{v\in D}\eta_v\beta_v^{v_1} + \sum_{c\in C}\omega_c\gamma_c^{v_1} \le B\ \forall v_1\in V$$

$$\sum_{e\in N_{v_1}} \frac{f_c(e)}{R(e)} \le \alpha_{s_c}^{v_1}+\beta_{d_c}^{v_1}+\gamma_c^{v_1}\ \forall c\in C\ \forall v_1\in V$$

$$0\le f_c(e)\le 1,\ 0\le R(e)\ \forall c\in C\ \forall e\in E$$

$$0\le \alpha_v\ \forall v\in S,\ 0\le\beta_v\ \forall v\in D,\ 0\le\gamma_c\ \forall c\in C$$

$$0\le \alpha_v^{v_1}\ \forall v\in S\ ,\ 0\le\beta_v^{v_1}\ \forall v\in D\ \forall v_1\in V$$

$$0\le \gamma_c^{v_1}\ \forall c\in C\ \forall v_1\in V \tag{6.14}$$

We have now removed all infinite constraints from the optimization problem. But, this problem is still not convex as the functions $f_c(e)/R(e),\ f_c(e)R(e),\ f_c(e)R^2(e)$ are neither convex nor concave in $(f_c(e), R(e))$. Due to non-convexity, even for small size networks, it is not possible to find the optimal solution in a reasonable time using current non-linear optimization tools. Thus, we propose heuristics to solve this problem. The heuristics solve quadratic convex optimization problems iteratively, which can be solved very fast using existing algorithms. In fact, in the simulations, the algorithms converged in very few iterations (3-4 iterations in most instances). The performance of the heuristics were close to the performance of an algorithm

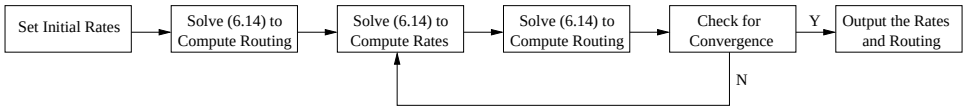


Figure 6.1: Iterative algorithm for (6.14)

for computing routing and link powers specific to each traffic matrix, Bhatia and Kodialam [12] (which has been shown to perform close to the optimal for the specific traffic matrix).

6.2.3 Iterative Algorithm

We propose an iterative algorithm to compute a solution to the optimization problem (6.14). The algorithm is depicted in Figure 6.1, and explained below. We fix $B = 2/3$ so that the obtained solution is schedulable.

1: Set rates $R(e) = B\Delta \max_{u \in S, v \in D, c \in C}\{|S|\lambda_u, |D|\eta_v, |C|\omega_c\}$, where Δ is the maximum node degree in the network. The value of R is chosen so that there is a feasible solution to (6.14). It is easy to see that with these values for R, there exists a solution that satisfies the constraints imposed by Equation 6.12. Set $k = 1$ and $R^k = R$.

2: Set $R = R^k$, and fix $f_c(e) = 0$ for all $c \in C$ for all edges e such that $R(e) = 0$. Solve (6.14) to compute f, which is a linear program for fixed R and can be solved in polynomial time. Set $f^k = f$.

3: Fix $f = f^k$, and solve (6.14) to compute $R^k = R$, which is a separable convex program for fixed f and can be solved in polynomial time.[1] Increment k.

4: Repeat Steps 2 and 3 (call one execution of Steps 2 and 3 an iteration), and stop when the objective value stops decreasing (this is the convergence criteria).

The transmission power to use on each edge e is then given by $N_0 W/\sigma(e)(2^{R(e)/W} - 1)$. In our experiments the algorithm usually converges in three to four iterations. Since the algorithm stops execution when the objective function stops decreasing, the algorithm always converges to a local minimum.

We next present some extensions to the fixed solution, which optimize the link transmission powers for the current traffic, while using the static routing computed above.

6.2.4 Static Routing, Centralized Rate Change

The first extension uses the routing computed by our algorithm, but uses optimal link rates (power assignment) for a given traffic matrix t_c. It is practical to assume that rates can adapt to the changing traffic, as it does not have as disrupting an impact as a change of routing has. The algorithm still suffers from the problem of global traffic estimation at any point in time. The algorithm solves the following

[1] We do not fix rates to zero for edges carrying zero traffic as $R(e)$ is in the denominator in the scheduling constraints. We set those rates to zero after solving (6.14) for R.

optimization problem to find the rates:

$$\min_{R} \sum_{e \in E} \sum_{c \in C} f_c(e) t_c \frac{N_0 ln2}{\sigma(e)} (1 + \frac{ln2}{2W} R(e) + \frac{ln^2 2}{6W^2} R^2(e))$$

s.t.

$$\sum_{e \in N_v} \sum_{c \in C} \frac{f_c(e)}{R(e)} t_c \leq B \quad \forall \; v \in V$$

$$0 \leq R(e) \quad \forall \; e \in E \tag{6.15}$$

The optimization problem is a convex-constrained convex minimization problem that can be solved in polynomial time, Bhatia and Kodialam [12]. The performance of this algorithm is always better than that of the static routing/rates algorithm.

6.2.5 Static Routing, Distributed Rate Change

This algorithm also uses the static routing computed by the iterative algorithm, but computes the traffic specific rates in a distributed manner, which is more practical for a wireless network. The algorithm only needs to know the traffic using a particular node, since all commodities can be ignored for which $f_c(e) = 0$ for edges e adjacent to the node. Each node solves the following optimization problem and each edge is assigned the maximum of the rates assigned by its head and tail nodes:

$$\min_{R} \sum_{e \in N_v} \sum_{c \in C} f_c(e) t_c \frac{N_0 ln2}{\sigma(e)} (1 + \frac{ln2}{2W} R(e) + \frac{ln^2 2}{6W^2} R^2(e))$$

s.t.

$$\sum_{e \in N_v} \sum_{c \in C} \frac{f_c(e)}{R(e)} t_c \leq B$$

$$0 \leq R(e) \quad \forall \; e \in N_v \tag{6.16}$$

The algorithm has been proved to be a 2-approximation of the centralized algorithm of (6.15) if only the first three terms of the Taylor series expansion of Equation 6.13 are used, Bhatia and Kodialam [12].

6.2.6 Traffic Scheduling

Each edge needs to transmit traffic for commodity c for a fraction $f_c(e)t_c/R(e)$ of the time. We assume a TDMA protocol among contending transmissions (which are transmissions on edges at common nodes). The length of the time slots is usually fixed, but mapping the time fraction to number of discrete slots in a frame produces a very minute performance loss as the time slot length is very small (of the order of milliseconds), as proved in Kodialam and Nandagopal [43]. Since the computed routing/rate pair (f, R) is schedulable for all traffic matrices in $\mathcal{T}$ (we set B=2/3), we can either use the optimal scheduling algorithm of Hajek and Sasaki [31] for small networks, or faster heuristic algorithms like the ones proposed by Post et al. [62] and Tassiulas et al. [72].

6.2.7 Extension to other Interference Models

The interference model can be extended to incorporate the restriction that no node can transmit when its neighbor is transmitting or receiving (as in CSMA/CA with RTS-CTS used in 802.11). This can be incorporated by adding a constraint on routing and rates given in Kodialam and Nandagopal [43] to our formulation. We can also incorporate the interference models studied in Kodialam and Nandagopal [44].

They consider two models: half-duplex, in which a node can either transmit to one node or receive from up to $\Omega(v)$ nodes at any time. The other model is the full-duplex model, in which a node v can simultaneously transmit to one node and receive from up to $\Omega(v)$ nodes.

6.3 Simulation Results and Discussion

We implement the proposed algorithms (using MOSEK [59] for separable convex optimization), and simulate them on the network shown in Figure 6.2, Kodialam and Nandagopal [43]. The network has 15 nodes and 56 uni-directional edges. We choose nodes 1,6,9,12,14 as ingress/egress nodes, and thus we have 20 commodities. The particular nodes are chosen as ingress/egress to allow most edges to be used in shortest paths between some ingress/egress pair. We use the values of W, N_0 as in Bhatia and Kodialam [12], and generate channel gains $\sigma(e)$ uniformly randomly, of the same order as in Bhatia and Kodialam [12]. We generate the hose model traffic bounds according to a model similar to the gravity model, Roughan et al. [67]. For each ingress/egress node v, the values of λ_v and η_v are proportional to their degrees (δ_v). The values are scaled by a scaling factor (t_s) to demonstrate performance under different traffic loads. The commodity upper bounds (ω_c) are then taken as proportional to the minimum of λ_{s_c} and η_{d_c}, where s_c and d_c are the source and destination nodes for commodity c respectively. The constant multiplier is chosen so most of the pipe and hose model constraints are not redundant. Table 6.2 lists the parameter values.

Table 6.2: Simulation Parameters

Parameter	**Value**
Number of nodes, N	15
Number of edges, E	56
Channel Bandwidth, W	10MHz
Channel gain, $\sigma(e)$	Uniform random, $(5-10)$x10^{-13}
Noise spectral density, N_0	4x10^{-18}mW/Hz
Source (Ingress) Nodes, S	$\{1, 6, 9, 12, 14\}$
Destination (Egress) Nodes, D	$\{1, 6, 9, 12, 14\}$
Number of commodities, M	20
Traffic scaling factor, t_s	1-10
Total outgoing traffic bound, λ_v	$20\delta_v t_s$ kbps
Total incoming traffic bound, η_v	$20\delta_v t_s$ kbps
Bound on each commodity, ω_c	$2\min\{\lambda_{s_c}, \eta_{d_c}\}/\|S\|$
Schedulability constant, B	2/3

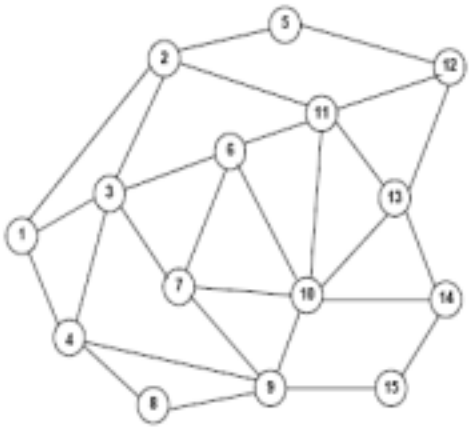


Figure 6.2: Network used for simulations

We compare our algorithms for specific traffic matrices in the traffic matrix polytope $\mathcal{T}$. The problem is too hard to compute an optimal solution for even a very small size network. Thus, we compare our algorithm with an algorithm that has been proposed for online traffic-specific routing and rate computation. We first describe this algorithm.

6.3.1 Traffic Specific Routing and Rates

We compare our algorithms against the algorithm proposed by Bhatia and Kodialam [12], that computes routing and rates specific to a traffic matrix. The algorithm is a 3-approximation to the optimal, and has been shown to perform very close to optimal traffic-specific solution in practice. This algorithm needs global traffic matrix information, and needs to change the routing and rates as traffic changes.

We compare the algorithms for a traffic matrix, T^*, that leads to the maximum power consumption by our algorithm in the network. We compute the traffic

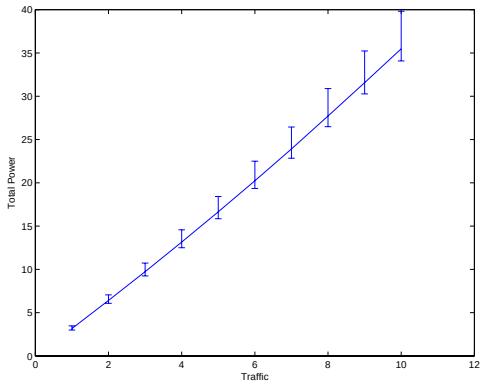


Figure 6.3: Worst case (traffic matrix T^*) total transmission power for static routing and rate assignment

matrix, T^*, by solving the oracle of (6.8) for the routing and rates computed by our algorithm. We then execute the algorithms that do traffic-specific optimization. The scaling factor t_s is varied from 1 to 10 (thus scaling the polytope boundaries), and the channel gains $\sigma(e)$ are generated randomly 10 times (same sets of channel gains are used for each value of t_s). Different sets of channel gains lead to different paths and rates for each commodity. Figure 6.3 shows the variation (average and range over the 10 sets of channel gain values) of total transmission power consumption in the network for traffic matrix T^*, with respect to the traffic scaling factor, for the static routing/rates algorithm. The worst case total power consumption increases almost linearly with the traffic scaling factor t_s, since the rate is less than W, and thus the first few terms of the Taylor series expansion dominate.

Next, we demonstrate the relative performance of our algorithms to the performance of the traffic specific routing and rate computation algorithm of Bhatia and Kodialam [12]. The relative performance for each algorithm is total power of

the traffic specific routing and rate algorithm divided by total power of the corresponding algorithm. Figure 6.4 shows the relative performance for the static routing and rates algorithm, with respect to t_s. For low traffic, the performance is better than the traffic specific routing and rates algorithm of Bhatia and Kodialam [12]. As traffic increases, the relative performance of the traffic-specific algorithm improves, but is only marginally better than the static routing and rates algorithm. Figure 6.5 shows the relative performance for the static routing centralized rate change algorithm. Results show that centralized rate optimization works slightly better than the traffic specific routing and rates algorithm for almost all values of t_s. Thus, just modifying the transmission powers (rates) with static routing, which is not as disruptive as modifying both routing and rates, results in a performance slightly better than the traffic specific routing and rates algorithm for the traffic values considered. Figure 6.6 shows the relative performance for the static routing distributed rate change algorithm. The performance is slightly worse than that for the centralized traffic specific rate optimization, but is still slightly better the centralized traffic specific routing and rates algorithm.

Thus, as the results show, computing a fixed routing and set of link rates (transmission powers), that does not have the drawbacks of online traffic specific routing and rate changes, has a comparable performance compared to a recent traffic specific routing and rate optimization algorithm, Bhatia and Kodialam [12]. The performance is further improved by using online traffic specific distributed or centralized rate changes with the fixed routing produced by our algorithm. All the algorithms we propose do not have the disadvantage of disrupting current traffic

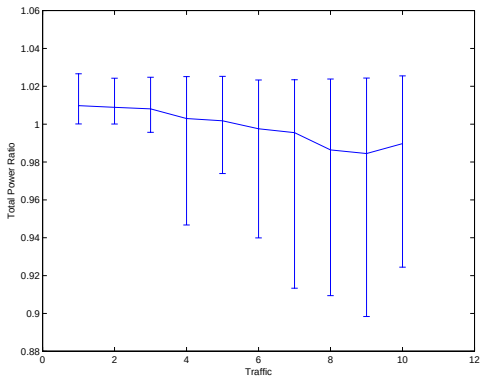


Figure 6.4: Relative performance for static routing and rates for traffic matrix T^*

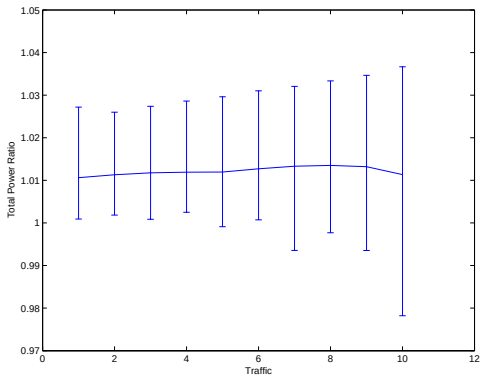


Figure 6.5: Relative performance for centralized traffic specific rates with static routing for traffic matrix T^*

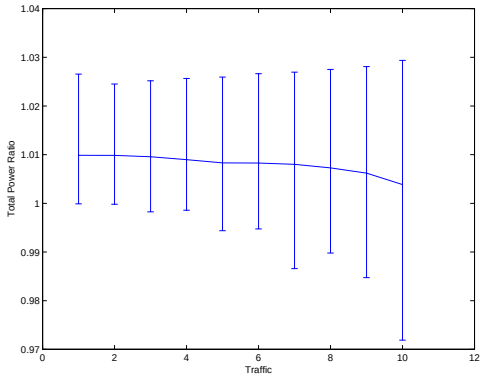


Figure 6.6: Relative performance for distributed traffic specific rates with static routing for traffic matrix T^*

flows or causing routing instabilities, as can be the case with the traffic specific routing and rate change algorithm. Also, the routing can be computed off-line in a centralized way, while the rates can be assigned in a distributed manner depending only on the traffic flowing through each node, and the algorithm performs comparable to the centralized online traffic specific routing and rate change algorithm.

Chapter 7

Conclusion and Future Work

We studied the problem of robust design of wireless networks. We first considered the problem of relay placement in wireless sensor networks for design of a topology that is tolerant to desired number of device and link failures. We proposed approximation algorithms for the problem with various generalizations. We provided analysis for the algorithms for networks in the Euclidean plane, as well as for networks in arbitrary metric spaces. We provided analysis for both homogeneous and heterogeneous networks, i.e., networks with the same transmission range for all nodes, as well as with different transmission range for each node. We also considered the generalization where there are obstacles in the network where relay nodes cannot be placed. We show the algorithms have the same approximation guarantees as for the case of no obstacles. We showed via simulations that the algorithms perform better than what the approximation results suggest. We also considered the application of the algorithms to topology reconfiguration with minimal movement of relay nodes. The application of the movement algorithms is to the backbone of battlefield ad-hoc networks, which have low mobility and require high degree of fault-tolerance.

The second problem we considered is of joint routing and power control in a multi-hop wireless network with unknown traffic. We modelled the problem as a

semi-infinite non-linear optimization problem. We reduced the problem to a semi-definite non-linear program, and provided efficient algorithms to solve it. We showed via extensive simulation results that our algorithm has a performance slightly better than a policy that continuously changes the routing and link transmission powers with changing traffic. In practice, since traffic demands in the network are hard to measure, and they change continuously, deployment of a traffic-dependent policy results in sub-optimal routing that has to be updated continuously. Frequent path updates can cause further problems, e.g., communication overhead which consumes power and bandwidth, temporal route instability due to asynchronous information exchange, and disruption in traffic flow. Our algorithm computes a static routing, that remains fixed for any traffic in the network, i.e., our routing is traffic-oblivious. Thus, our policy does not have the above mentioned drawbacks. The traffic-optimized algorithm we compare with is a 3-approximation of the optimal, and has been shown to have a performance close to the optimal. Thus, our traffic-oblivious algorithms perform close to optimal as well.

7.1 Future Research Directions

This dissertation leads to some interesting problems to work on in the future. We list them in the following sections.

7.1.1 Relay Placement

The first set of problems is on proposing and analyzing improved algorithms for relay placement. There is a considerable interest in the research community on relay placement for connectivity and fault-tolerance in sensor networks. There are a few directions that can be taken after understanding the analysis described in this dissertation, and the algorithms can be improved.

Another interesting and challenging extension is to relax the assumption of having circular transmission area around sensor and relay nodes. Although the transmission range changes continuously depending on the fading and interference from other nodes [64], it would be quite hard to propose algorithms for initial topology design of sensor networks based on real-time fading and interference. It is easier to take into account long-term effects, like those of the terrain and obstacles in the networks. They lead to shadowing and excessive fading, inducing regions in the area around each node where a receiving node might not be able to receive its transmissions. Thus, the transmission area around each node may not be circular. The problem involves modelling these constraints, and proposing and analyzing algorithms for relay placement in this scenario.

The problem of relay placement while taking into account the correlation among sensor nodes is also interesting. Multiple sensors in a sensor network may have correlated data, and thus it is not necessary to provide the same level of connectivity between every pair of sensor nodes. The objective of a sensor network is to communicate the sensed data to the sink. Thus, the problem is to provide

fault-tolerance to the sensed data, rather than to paths between sensor nodes. The correlation between sensor nodes provides a degree of fault-tolerance, which may lead to savings in the number of relays needed. The problem is to design relay placement algorithms that take into account the correlation between sensor nodes, and place relay nodes to provide the desired fault-tolerance. The correlation can be modelled as distance-based as proposed by Pattem et al. [61]. This problem further leads to the problem of integrating data aggregation (that saves energy consumption at sensor nodes) with the relay placement algorithm. The algorithms can be proposed for known correlation between sensor nodes, as well as for the scenario where the correlation between each pair is known to lie in a certain range. The last generalization makes the algorithms more application independent, and can be termed as correlation-oblivious network design.

Relay placement can also help achieve load balancing in a sensor network, that leads to energy balancing among sensor nodes, and thus network lifetime elongation. The problem arises due to limited transmission range of sensor nodes, and non-uniform placement of sensor nodes in the network. Relay nodes can be placed to fill the *holes* created in the network due to non-uniform placement of sensor nodes, and thus provide for the opportunity of load balancing. The problem can be further extended to take care of the data correlation and aggregation among sensor nodes.

Finally, we propose an extension to the topology reconfiguration problem addressed in Chapter 5. The proposed algorithms are centralized. A useful problem would be to propose distributed algorithms for topology reconfiguration as nodes move. The algorithms can be based on reinforcement learning and dynamic pro-

gramming, Sutton and Barto [69], Bertsekas [10].

7.1.2 Traffic-Oblivious Cross-Layer Design

The first problem we propose is to provide theoretical guarantees for the proposed algorithms. We show via simulations that the proposed algorithms for joint routing and power control perform well, but do not provide approximation guarantees. An algorithm with theoretical guarantees is a challenging area of future work.

The second problem is to consider more interference models. The network may not be using CDMA codes, and thus transmissions can potentially interfere with all transmissions in the network. In this model, since it is not practical to have only one node transmit at a time in the network, power-rate relation needs to take care of the Signal to Interference and Noise (SINR) ratio. Thus, the problem of determining the sets of nodes which transmit simultaneously has to be integrated with the routing and power control problem.

The last proposed problem is of optimizing the network topology along with computing traffic oblivious routing and power assignments. A considerable amount of work has been done for integrated topology design and routing in Free Space Optical (FSO) networks, Kashyap et al. [38, 37], Kalantari et al. [36], Zhuang et al. [79], and Desai and Milner [23]. The FSO links are very narrow point-to-point links with very low interference. Effectively, there is no interference constraint in the network. The results in these papers show that the topology and routing are inter-related. Thus, the problem of determining the nodes to communicate with at each

node (in addition to computing the routing and transmission power) is important in our problem as well, especially in the case of directional antennas, where the number of interfaces is limited.

Bibliography

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[2] I. F. Akyildiz, X. Wang, and W. Wang, "Wireless mesh networks: a survey," *Computer Networks*, vol. 47, no. 4, pp. 445–487, 2005.

[3] D. Applegate and E. Cohen, "Making intra-domain routing robust to changing and uncertain traffic demands: understanding fundamental tradeoffs," *ACM SIGCOMM*, pp. 313–324, 2003.

[4] E. Arkin, E. Demaine, and J. Mitchell, "The puddle-jumper problem," *Personal Communication*, 2005.

[5] V. Auletta, Y. Dinitz, Z. Nutov, and D. Parente, "A 2-approximation algorithm for finding an optimum 3-vertex connected spanning subgraph," *Journal of Algorithms*, vol. 32, pp. 21–30, 1999.

[6] Y. Azar, E. Cohen, A. Fiat, H. Kaplan, and H. Räcke, "Optimal oblivious routing in polynomial time," *ACM Symposium of Theory of Computing*, pp. 383–388, 2003.

[7] M. Bahramgiri, M. Hajiaghayi, and V. S. Mirrokni, "Fault-tolerant and 3-dimensional distributed topology control algorithms in wireless multi-hop networks," *IEEE International Conference on Computer Communications and Networks*, 2005.

[8] S. Banerjee and S. Khuller, "A clustering scheme for hierarchical routing in wireless networks," *IEEE INFOCOM*, pp. 1028–1037, 2001.

[9] P. Basu and J. Redi, "Movement control algorithms for realization of fault-tolerant ad hoc robot networks," *IEEE Network*, pp. 36–44, July/August, 2004.

[10] D. P. Bertsekas, *Dynamic Programming and Optimal Control.* Athena Scientific, 2005, vol. 1.

[11] C. Bettstetter, "On the connectivity of ad hoc networks," *The Computer Journal*, vol. 47, no. 4, pp. 432–447, 2004.

[12] R. Bhatia and M. Kodialam, "On power efficient communication over multi-hop wireless networks: joint scheduling, routing and power control," *IEEE INFOCOM*, 2004.

[13] S. Boyd and L. Vandenberghe, *Convex Optimization.* Cambridge University Press, 2004.

[14] J. L. Bredin, E. D. Demaine, M. Hajiaghayi, and D. Rus, "Deploying sensor networks with guaranteed capacity and fault tolerance," *ACM MobiHoc*, pp. 309–319, 2005.

[15] D. Chen, D.-Z. Du, X.-D. Hu, G.-H. Lin, L. Wang, and G. Xue, "Approximations for Steiner trees with minimum number of Steiner points," *Theoretical Computer Science*, vol. 262, pp. 83–99, 2001.

[16] X. Cheng, D.-Z. Du, L. Wang, and B. Xu, "Relay sensor placement in wireless sensor networks," *ACM Winet*, 2004.

[17] J. Cheriyan and R. Thurimella, "Algorithms for parallel k-vertex connectivity and sparse certificates," *SIAM Journal of Computing*, vol. 22, no. 1, pp. 157–174, 1993.

[18] Concorde, *http://www.tsp.gatech.edu/concorde.html*.

[19] P. Corke, S. Hrabar, R. Peterson, D. Rus, S. Saripalli, and G. Sukhatme, "Autonomous deployment and repair of a sensor network using an unmanned aerial vehicle," *IEEE International Conference on Robotics and Automation*, 2004.

[20] ——, "Deployment and connectivity repair of a sensor net with a flying robot," *IEEE International Symposium on Experimental Robotics*, 2004.

[21] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. Wiley Series in Telecommunications and Signal Processing, Wiley-Interscience, 2006.

[22] R. L. Cruz and A. V. Santhanam, "Optimal routing, link scheduling and power control in multi-hop wireless networks," *IEEE INFOCOM*, 2003.

[23] A. Desai and S. Milner, "Autonomous reconfiguration in free-space optical sensor networks," *IEEE JSAC Optical Communications and Networking Series*, vol. 23, no. 8, pp. 1556–1563, 2005.

[24] Y. Dinitz and Z. Nutov, "A 3-approximation algorithm for finding optimum 4,5-vertex connected spanning subgraphs," *Journal of Algorithms*, vol. 32, pp. 31–40, 1999.

[25] N. G. Duffield, P. Goyal, A. G. Greenberg, P. P. Mishra, K. K. Ramakrishnan, and J. E. var der Merwe, "A flexible model for resource management in virtual private networks," *ACM SIGCOMM*, 1999.

[26] T. Elbatt and A. Ephremides, "Joint scheduling and power control for wireless ad-hoc networks," *IEEE INFOCOM*, 2002.

[27] A.-H. Esfahanian, *Selected Topics in Graph Theory*. Cambridge University Press, ch. On the evolution of connectivity algorithms.

[28] G. Foschini and Z. Miljanic, "A simple distributed autonomous power control algorithm and its convergence," *IEEE Transactions on Vehicular Technology*, vol. 42, no. 4, pp. 641–646, 1993.

[29] H. N. Gabow, "A matroid approach to finding edge connectivity and packing arborescences," *IEEE Annual Symposium on Foundations of Computer Science*, pp. 812–822, 1991.

[30] M. Garey and D. Johnson, "Computers and intractability: A guide to the theory of NP-Completeness," 1979.

[31] B. Hajek and G. Sasaki, "Link scheduling in polynomial time," *IEEE Transactions on Information Theory*, vol. 34, no. 5, pp. 910–917, 1988.

[32] M. T. Hajiaghayi, N. Immorlica, and V. S. Mirrokni, "Power optimization in fault-tolerant topology control algorithms for wireless multi-hop networks," *ACM MobiCom*, pp. 300–312, 2003.

[33] X. Han, X. Cao, E. L. Lloyd, and C.-C. Shen, "Fault-tolerant relay node placement in heterogeneous wireless sensor networks," *Personal Communication*, 2006.

[34] B. Hao, J. Tang, and G. Xue, "Fault-tolerant relay node placement in wireless sensor networks: Formulation and approximation," *IEEE High Performance Switching and Routing*, pp. 246–250, 2004.

[35] F. Harary, "The maximum connectivity of a graph," *Proc. of the National Academy of Sciences*, vol. 48, no. 7, pp. 1142–1146, 1962.

[36] M. Kalantari, A. Kashyap, K. Lee, and M. Shayman, "Network topology control and routing under interface constraints by link evaluation," *Conference on Information Sciences and Systems*, 2005.

[37] A. Kashyap, M. Kalantari, K. Lee, and M. Shayman, "Rollout algorithms for topology control and routing of unsplittable flows in wireless optical backbone networks," *Conference on Information Sciences and Systems*, 2005.

[38] A. Kashyap, S. Khuller, and M. Shayman, "Topology control and routing over wireless optical backbone networks," *Conference on Information Sciences and Systems*, 2004.

[39] ——, "Relay placement for higher order connectivity in wireless sensor networks," *IEEE INFOCOM*, 2006.

[40] S. Khuller and B. Raghavachari, "Improved approximation algorithms for uniform connectivity problems," *Journal of Algorithms*, vol. 21, no. 2, pp. 434–450, 1996.

[41] S. Khuller and U. Vishkin, "Biconnectivity approximations and graph carvings," *Journal of the ACM*, vol. 41, no. 2, pp. 214–235, 1994.

[42] M. Kodialam, T. V. Lakshman, and S. Sengupta, "Maximum throughput routing of traffic in the hose model," *IEEE INFOCOM*, 2006.

[43] M. Kodialam and T. Nandagopal, "Characterizing achievable rates in multi-hop wireless networks: the joint routing and scheduling problem," *ACM Mobicom*, 2003.

[44] ——, "Characterizing achievable rates in multi-hop wireless mesh networks with orthogonal channels," *IEEE/ACM Transactions on Networking*, vol. 13, no. 4, pp. 868–880, 2005.

[45] G. Kortsarz and Z. Nutov, "Approximating node connectivity problems via set covers," *Algorithmica*, vol. 37, pp. 75–92, 2003.

[46] N. Li and J. C. Hou, "FLSS: a fault tolerant topology control algorithm for wireless networks," *ACM Mobicom*, pp. 275–286, 2004.

[47] X.-Y. Li, W.-Z. Song, and Y. Wang, "Efficient topology control for wireless ad hoc networks with non-uniform transmission ranges," *ACM Wireless Networks*, vol. 11, no. 3, 2005.

[48] X.-Y. Li, P.-J. Wan, Y. Wang, and C.-W. Yi, "Fault tolerant deployment and topology control in wireless networks," *ACM MobiHoc*, pp. 117–128, 2003.

[49] Y. Li, J. Harms, and R. Holte, "Traffic-oblivious energy-aware routing for multihop wireless networks," *IEEE INFOCOM*, 2006.

[50] G.-H. Lin and L. Wang, "Steiner tree problem with minimum number of Steiner points and bounded edge-length," *Information Processing Letters*, vol. 69, pp. 53–57, 1999.

[51] Y.-H. Lin and R. L. Cruz, "Power control and scheduling for interfering links," *IEEE Information Theory Wowrkshop*, 2004.

[52] H. Liu, P. Wan, and X. Jia, "Fault-tolerant relay node placement in wireless sensor networks," *International Computing and Combinatorics Conference*, 2005.

[53] E. L. Lloyd, R. Liu, M. V. Marathe, R. Ramanathan, and S. S. Ravi, "Algorithmic aspects of topology control problems for ad hoc networks," *ACM MobiHoc*, 2002.

[54] L. Lovász and M. D. Plummer, *Matching Theory*. North-Holland, 1986.

[55] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless sensor networks for habitat monitoring," *ACM International Workshop on Wireless Sensor Networks and Applications*, 2002.

[56] I. Măndoiu and A. Zelikovsky, "A note on the MST heuristic for bounded edge-length Steiner trees with minimum number of Steiner points," *Information Processing Letters*, vol. 75, no. 4, pp. 165–167, 2000.

[57] D. W. Matula, "Determining edge connectivity in o(mn)," *IEEE Annual Symposium on Foundations of Computer Science*, pp. 249–251, 1987.

[58] C. Monma and S. Suri, "Transitions in geometric minimum spanning tree," *Discrete and Computational Geometry*, vol. 8, pp. 265–293, 1992.

[59] Mosek optimization toolbox, *http://www.mosek.com*.

[60] M. J. Neely, E. Modiano, and C. E. Rohrs, "Dynamic power allocation and routing for time varying wireless networks," *IEEE INFOCOM*, 2003.

[61] S. Pattem, B. Krishnamachari, and R. Govindan, "The impact of spatial correlation on routing with compression in wireless sensor networks," *ACM IPSN*, pp. 28–35, 2004.

[62] M. J. Post, A. S. Kershenbaum, and P. E. Sarachik, "Scheduling multi-hop cdma networks in the presence of secondary conflicts," *Algorithmica*, pp. 365–393, 1989.

[63] R. Ramanathan and R. Rosales-Hain, "Topology control of multihop wireless networks using transmit power adjustment," *IEEE INFOCOM*, pp. 404–413, 2000.

[64] T. S. Rappaport, *Wireless Communications: Principles and Practice*. Prentice Hall, 2002.

[65] G. Robins and J. S. Salowe, "Low-degree minimum spanning trees," *Discrete Computational Geometry*, vol. 14, pp. 151–165, 1995.

[66] V. Rodoplu and T. H. Meng, "Minimum energy mobile wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, pp. 1333–1344, 1999.

[67] M. Roughan, A. Greenberg, C. Kalmanek, M. Rumsewicz, J. Yates, and Y. Zhang, "Experience in measring backbone traffic variability: Models, metrics, measurements and meaning," *International Teletraffic Congress*, 2003.

[68] C. Shields, Jr., V. Jain, S. Ntafos, R. Prakash, and S. Venkatesan, "Fault tolerant mobility planning for rapidly deployable wireless networks," *Workshop on Fault-Tolerant Parallel and Distributed Systems*, pp. 770–789, 1998.

[69] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[70] V. Tabatabaee, A. Kashyap, S. Bhattacharjee, R. La, and M. Shayman, "Robust routing with unknown traffic matrices," *ISR Technical Report TR 2006-9, University of Maryland*, 2006.

[71] J. Tang, B. Hao, and A. Sen, "Relay node placement in large scale wireless sensor networks," *Computer Communications*, vol. 29, pp. 490–501, 2006.

[72] L. Tassiulas, A. Ephremides, and J. Gunn, "Solving hard optimization problems arising in packet radio networks using hopfield nets," *Conference on Informations Sciences and Systems*, pp. 603–608, 1989.

[73] B. Thallner and H. Moser, "Topology control for fault-tolerant communication in highly dynamic wireless networks," *IEEE International Workshop on Intelligent Solutions in Embedded Systems*, 2005.

[74] R. Wattenhofer, L. Li, P. Bahl, and Y.-M. Wang, "Distributed topology control for power efficient operation in multihop wireless ad hoc networks," *IEEE INFOCOM*, pp. 1388–1397, 2001.

[75] R. Wattenhofer and A. Zollinger, "XTC: a practical topology control algorithm for ad-hoc networks," *IEEE International Parallel and Distributed Processing Symposium*, 2004.

[76] J. E. Wieselthier, C. M. Barnhart, and A. Ephremides, "A neural network approach to routing without interference in multi-hop networks," *IEEE Transactions on Communications*, pp. 365–393, 1994.

[77] R. Yates, "A framework for uplink power control in cellular radio systems," *IEEE Journal on Selected Areas of Communications*, vol. 13, no. 7, pp. 1341–1348, 1995.

[78] J. Zander, "Distributed cochannel interference control in cellular radio systems," *IEEE Transactions on Vehicular Technology*, vol. 41, no. 3, pp. 305–311, 1992.

[79] J. Zhuang, M. J. Casey, S. D. Milner, S. A. Gabriel, and G. Baecher, "Multiobjective optimization techniques in topology control of free space optical networks," *IEEE MILCOM*, 2004.