SYNTHESIZING ROBUST NETWORKS FOR ENGINEERING APPLICATIONS WITH RESOURCE CONSTRAINTS

A Dissertation

by

HARSHA NAGARAJAN

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Swaroop Darbha
Co-Chair of Committee,	Sivakumar Rathinam
Committee Members,	K. R. Rajagopal
	Sergiy Butenko
Head of Department,	Andreas A. Polycarpou

May 2014

Major Subject: Mechanical Engineering

ABSTRACT

This dissertation deals with the following simpler version of an open problem in system realization theory which has several important engineering applications: Given a collection of masses and a set of linear springs with a specified cost and stiffness, a resource constraint in terms of a budget on the total cost, the problem is to determine an optimal connection of masses and springs so that the resulting structure is as stiff as possible, i.e., the structure is connected and its smallest non-zero natural frequency is as large as possible.

One often encounters variants of this problem in deploying Unmanned Aerial Vehicles (UAVs) for civilian and military applications. In such problems, one must determine the pairs of UAVs that must maintain a communication link so that constraints on resources and performance, such as a limit on the maximum number of communication links deployed, power consumed and maximum latency in routing information from one UAV to the other, are met and a performance objective is maximized. In this dissertation, algebraic connectivity, or its mechanical analog - the smallest non-zero natural frequency of a connected structure was chosen as a performance objective. Algebraic connectivity determines the convergence rate of consensus protocols and error attenuation in UAV formations and is chosen to be a performance objective as it can be viewed as a measure of robustness in UAV communication networks to random node failures.

Underlying the mechanical and UAV network synthesis problems is a Mixed Integer Semi-Definite Program (MISDP), which was recently shown to be a NP-hard problem. There has not been any systematic procedure in the literature to solve this problem. This dissertation is aimed at addressing this void in the literature. The

novel contributions of this dissertation to the literature are as follows: a) An iterative primal-dual algorithm and an algorithm based on the outer approximation of the semi-definite constraint utilizing a cutting plane technique were developed for computing optimal algebraic connectivity. These algorithms are based on a polyhedral approximation of the feasible set of MISDP, b) A bisection algorithm was developed to reduce the MISDP to a Binary Semi-Definite Program (BSDP) to achieve better computational efficiency, c) The feasible set of the MISDP was efficiently relaxed by replacing the positive semi-definite constraint with linear inequalities associated with a family of Fiedler vectors to compute *a tighter upper bound for algebraic connectivity*, d) Efficient neighborhood search heuristics based on greedy methods such as the k-opt and improved k-opt heuristics were developed, e) Variants of the problem occurring in UAV backbone networks and Air Transportation Management were considered in the dissertation along with numerical simulations corroborating the methodologies developed in this dissertation.

DEDICATION

To my parents, *S. Nagarajan* and *Usha Rajan*

ACKNOWLEDGEMENTS

As a graduate student at Texas A&M University, it has been a very exciting journey through times that sometimes have been highly rewarding and other times that have been difficult but certainly “different” from what I expected when I came in. However, looking back at the wonderful seven years I spent in TAMU, I am grateful for the support, encouragement, and friendship of a large number of people who have made my stay very productive and enjoyable.

First and foremost, I am deeply thankful to my thesis advisor, Dr.Swaroop Darbha, for the opportunity to be a member of his research group, and the freedom to grow as a researcher. He has always been a source of inspiration with his intuition and insight, and being part of his group has allowed me to interact with, and learn from, a number of knowledgeable and talented individuals. I am fortunate to have been his student and thank him for the countless hours that he has patiently spent trying to mold me into a better researcher, writer, and person.

I am also thankful to Dr.Sivakumar Rathinam for having served as a co-advisor on my thesis committee. Having chosen a topic of research which was of great theoretical significance but with little literature available on the practical aspects, many insightful discussions with Dr.Rathinam helped us greatly demystify the difficulties associated with the understanding of the problem.

I extend my deep sense of gratitude to Dr.K.R.Rajagopal, Distinguished Professor of Mechanical Engineering, for serving on my thesis committee. If not for my frequent interactions with him regarding the importance of research and general philosophy about learning during my stint as a masters student, I would not have taken the audacious decision to quit a lucrative job in the energy industry and pursue my doctoral studies at TAMU.

I also thank Dr.Sergiy Butenko for being a valuable member of my thesis committee, and Dr.Kalyanababu Nakshatrala, who taught me a great deal about computational methods.

I thank the many current and past members of the research group with whom I have had fruitful interactions. Some of them include Kaarthik Sundar, Satyanarayana Gupta, Myoungkuk Park and Navid Mohsenizadeh. I specially thank Kaarthik for spending numerous hours discussing various open problems in combinatorial optimization and also for helping me cross my initial learning curve in `C++` during the starting days of my PhD.

Special thanks to my roommate, Dr.Shriram Srinivasan, for supporting my extracurricular interests and putting up with my little idiosyncrasies at times.

I am fortunate to have spent a great deal of time with some of the very knowledgable and nicest faculty members at TAMU. Though there are many such professors, I am particularly grateful to Dr.N.Sivakumar (Mathematics), Dr.Shankar Bhattacharyya (Electrical Engineering), Dr.Mysore Mohan (Chemistry) and Dr.Krishna Narayanan (Electrical Engineering). Special thanks to Dr.Bhattacharyya for letting me stay at one of his properties - free of cost - when I did not have any financial support.

Thanks to Dr.P.J.Guruprasad for being an excellent host during the initial days of my stay in College Station and providing a smooth transition to the graduate life.

I am also blessed to have formed many friendships at TAMU that have immensely helped me explore my interests outside work. Life at College Station would have been very dull and much less fun without the wonderful Carnatic music sessions and SPIC-MACAY concerts. I have greatly enjoyed the weekend music sessions with Bhavana Iyer, Sriranjani Darbha, Srinidhi Narayanan, Aditya Nandakumar, Hariharan Bhagavatheeswaran, Vibhuti Chandra, Dr.N.Sivakumar, Karthik Krishnamurthy, Jagannath Vijayan, Adithya Ganapathi and Dr.Krishnakumar Ravikumar. Special thanks to Dr.Srikrishna Doraiswamy (also coffee mate at the lab) and Dr.Atul Narayan for being fine connoisseurs of Carnatic music and supporting our music sessions, and to Jayavel Arumugam, a good friend and a remarkable photographer. Thanks to the families of Dr.Swaroop Darbha, Dr.Krishna Narayanan, Dr.Srividya Ramasubramaniam and Dr.Pazhaniappan for their sustained support in hosting many music sessions followed by fine food. Dr.Krishna Narayanan's family needs a special mention for hosting my concerts regularly, turning this event into an "Aggie" tradition. It would be rather incomplete to think of my progress in the field of Indian classical music without all my revered *gurus* since my childhood. My humble salutations to Shri.T.K.V.Ramanujacharyulu, a top notch violinist from India, for patiently teaching me the nuances of Carnatic music through live and Skype sessions.

During my undergraduate studies at NIT Trichy, I was very lucky to have interacted with many wonderful friends and faculty members. Thanks to Dr.Hemalatha Thiagarajan for being an excellent Mathematics professor from whom I learnt a lot. Special thanks to Dr.A.R.Ganesan's family for accommodating my music practice sessions on a daily basis for the entire four years. Thanks to G.S.Sundar Rajan, Ganapathy Ramanthan and Trichy Krishnaswamy for being wonderful friends and making my stay in Trichy such a memorable one.

Finally, I thank a special group of people who are the real reasons for my continued good fortune: my parents, Shri.S.Nagarajan and Smt.Usha Rajan, my uncle Shri.M.N.Gururaj, and my brother Adarsh Nagarajan. Any success of mine would be impossible without their unconditional love and support.

NOMENCLATURE

UAVs	Unmanned Aerial Vehicles
MISDP	Mixed Integer Semi-Definite Program
BSDP	Binary Semi-Definite Program
PSD	Positive Semi-Definite
MILP	Mixed Integer Linear Program
ILP	Integer Linear Program
IP	Integer Program

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

This dissertation deals with the development of *novel* tools for addressing an *open* problem in system realization theory which has relevance to several important problems in biomedicine, altering the dynamic response of discrete and continuous systems, connectivity of Very Large Scale Integrated (VLSI) circuits, as well as the co-ordination of Unmanned Aerial/Ground vehicles. The simplest case of this open problem, referred to as the *Basic Problem* (or simply, **BP**) is the following: *Given a finite set of masses, a set of linear springs and dampers, a given subset of springs or dampers that may only be connected between a specified pair of masses, a transfer function to be realized with a subset of these components by connecting them appropriately, the decision problem is to determine if there is an interconnection which can accomplish this objective.* The resolution of **BP** is open and far from simple.

If we restrict ourselves to mechanical systems with springs and masses, and require further that the interconnections should be made so that the resulting structure is one-dimensional, the resulting problem has a nice connection to Graph Laplacians in graph theory. Graph Laplacians (or simply Laplacians) play an important role in assessing robustness of connectivity and are similar to stiffness matrices in discrete structural mechanical systems. The analogy may be made as follows: a mass serves the role of a node and a spring serves the role of an edge that connects two nodes in a graph. If one assigns the cost of the edge to be the stiffness of the corresponding spring, the resulting Graph Laplacian is the same as the stiffness matrix that one obtains for the corresponding structural mechanical system. A brief overview of Laplacian matrices is given in section 1.1.

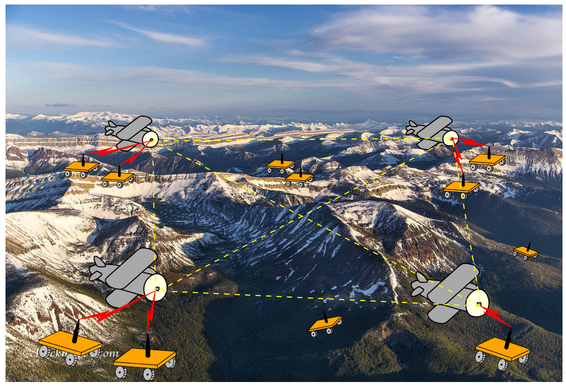
A typical transfer function in structural systems relates the input displacement or force acting on a mass to the displacement of another mass in the structural network. In the absence of any damping in the structural systems, only transfer functions that have purely imaginary zeros and poles can be realized. The poles of the system correspond to the natural frequencies of the interconnected system, the interconnections being the sought quantities. The zeros of the system can be thought of as the natural frequencies of an associated constrained system obtained by setting the output displacement to be identically zero, that is by constraining the appropriate mass to be stationary. Essentially, the fundamental problem of system realizability with a collection of springs and masses reduces to the following variant: Given a set V of masses and E of springs, and a set of bounds on natural frequencies $w_{1l}, w_{1u}, w_{2l}, w_{2u}, \ldots, w_{pl}, w_{pu}$, (with $p \leq |V|$), is there a connection of masses in which at most q springs are used which results in the interconnected structure having natural frequencies that lie between $[w_{1l}, w_{1u}], [w_{2l}, w_{2u}], \ldots, [w_{pl}, w_{pu}]$? This is a reasonable relaxation of the original problem which requires $w_{1l} = w_{1u},\ w_{2l} = w_{2u},\ \ldots, w_{pl} = w_{pu}$ in the following sense. The feasible set of the relaxed problem is bigger and the possibility of finding a solution should be better.

When $w_{il} = t,\ w_{iu} = \infty$ for all $i = 1, \ldots, p$, one obtains a decision problem for a related problem involving the maximization of *augmented* algebraic connectivity in graphs. The difference between the **BP** and the *augmented* algebraic connectivity maximization problem is as follows: In the **BP**, none of the masses are connected to any springs initially, whereas in the *augmented* algebraic connectivity problem, the masses may initially be connected partially and one is seeking *additional* edges to maximize the algebraic connectivity. This problem was only recently shown to be NP-hard [1] and may be stated as follows: Given an interconnected system of springs and masses, a prescribed number q and a positive number, t, the decision problem is to determine if one can find at most q *additional* springs that have not yet been used so as to make the second smallest natural frequency (which is also known as the algebraic connectivity for the associated Laplacian) to be greater than t. We will recall that the second smallest natural frequency is a measure of the "stiffness" of the structure and the smallest natural frequency is always zero corresponding to the rigid body mode admitted by the structure (a detailed discussion on the measure for stiffness of mechanical systems can be found in section 1.2.1). Since every instance of the maximum augmented algebraic connectivity problem is an instance of the system realization problem, NP-hardness of the former problem implies the NP-hardness of the latter problem and hence, non-trivial.

The problem of maximizing the augmented algebraic connectivity has applications to stiffening existing or damaged structures. The need for repair and strengthening of damaged or deteriorated structures subject to tight budget constraints has been an important challenge all over the world. As discussed in [2], there are many seismic resistant structures built before the 1970s which are still in service beyond their design life. These existing structures were designed with inadequate lateral load resistance because earlier building codes specified lower levels of seismic loads. Currently, it has been a topic of great interest to address the problem of deficiency in the structural system by adequately strengthening the structural system in order to attain the desired level of seismic resistance. Structural strengthening or rehabilitation, as defined in UNIDO (United Nations Industrial Development Organization) manual, may consist of modification of the existing structural members or addition of new structural members so that their structural strength, stiffness and/or ductility are improved. An improvement in the overall stiffness of a structure can be achieved through the addition of new structural members of known stiffness values to increase the respective characteristics of the structure like bracing in a frame or skeleton structure or new shear walls in a shear wall structure. One must also take into consideration the constraints on total budget/cost, while improving the overall stiffness of the structure. One may abstract the problem of strengthening structures as follows: Given a budget, a list of additional structural members to choose from along with their costs, the problem is to augment the structure so as to make it as strong (stiff) as possible by retrofitting additional structural members to the existing structure within the specified budget.

Algebraic connectivity, as the name indicates, is a measure of connectivity. In the structural context, a single dimensional structure is connected if a force applied at almost any point on the structure will influence the displacement of structure or the stress at almost all other locations and hence, cannot admit more than a single rigid body mode. Moreover, if the structure may be thought of as a linear discrete structural mechanical system, the structure is tightly connected if the stiffness of every spring is sufficiently high or equivalently, all its non-zero natural frequencies are sufficiently high. We may carry this analogy to applications involving a formation or collection of UAVs.

This dissertation is also motivated by a scenario as shown in the figure 1.1. In this scenario, there are clusters of ground robots moving in disparate regions that need to communicate their data and information amongst themselves. It is known a priori that the clusters will move slowly, and are known to be within a radius R_{max} of their centroid. The ground-to-ground communication between these clusters may be hampered by obstacles such as mountains or tall buildings that prevent line-of-sight communications. Since the power of a signal attenuates as the fourth power of distance in ground-to-ground communication, while it only decreases as the second power of the distance in ground-to-air and air-to-air communication [3, 4], an ad-hoc network of UAVs is envisioned. The UAVs serve as backbone



(a) Initial configuration

(b) Configuration after rigid body rotation

Figure 1.1: In this figure, part (a) represents an initial configuration of backbone UAVs communicating with ground robots in disparate regions. As shown with the coloring of robots, not all robots are able to maintain a ground-to-air communication link with the UAVs. But in part (b), after a rigid body rotation of the backbone network about the centroid, the remaining ground robots are able to maintain a ground-to-air communication link with the UAVs.

nodes and serve to establish communication between the clusters from ground. The robots use ground-to-air communication with the UAVs and the UAVs utilize air-to-air communication amongst themselves to reduce the overall power consumption for maintaining communication and transferring data.

The operational concept is as follows: The collection of UAVs maintain a fixed distance between them. The collection rotates about the centroid as a rigid body through an unit angle, stop at that configuration to facilitate communication with robots and then step through another unit angle and this procedure continues. Associated with each UAV, one may associate a circular footprint on the ground; robots in the footprint can utilize ground-to-air communication with the UAV. As the collection of UAVs rotates about the centroid, the footprints of UAVs sweep/cover the area of the footprint on the ground. Rotating the UAVs about the centroid helps in providing a time window for ground-to-air communication between the robots and the UAVs without the UAVs having to track the robots.

Maintaining a rigid formation of UAVs provides a convenient way of maintaining the backbone UAV network. An important problem of maintaining a rigid formation is the problem of determining the underlying information flow graph, i.e., the determination of the pairs of UAVs that are maintaining communication. It is well-known that with a given decentralized controller as in [5], the convergence rate of the error in maintaining a desired constant spacing with respect to other UAVs in the formation is influenced by the algebraic connectivity of the (unweighted) information flow graph; if the algebraic connectivity is higher, the convergence is faster. In essence, this problem is identical to **BP**.

Another variant of **BP** arises in the same scenario when we deal with the construction of an adhoc infrastructure network with UAVs, *i.e.*, the determination of the relative location of UAVs as well as the pairs of UAVs that must maintain air-to-air communication. Since UAVs have limited battery power on-board, power consumption is an important issue. We use the following model of power consumption: to maintain a connection (or a communication link) between the i^{th} and j^{th} UAVs, the power consumed is given by $\alpha_{ij} d_{ij}^2$, where d_{ij} is the distance between the UAVs and α_{ij} is the coefficient of proportionality and is dependent on the product of antenna gains of the transmitting and receiving UAVs. The coefficient α_{ij} may also be viewed as a strength of the communication link. If α_{ij} is higher, then the data rate that can be transmitted across the link is correspondingly higher. From the point of reducing interference in communication, there is an upper bound on the transmitted power by every UAV. This constraint limits communication between UAVs that are sufficiently far apart. The antennas and associated signal processing circuitry is typically powered by batteries on-board a UAV and this further limits the power that can be consumed in transmitting signals by every UAV. Instead of dealing with this constraint at the individual UAV level, we consider a surrogate constraint on the power consumption of the system as a whole. The total power consumed by the UAVs for maintaining air-to-air connectivity (or simply connectivity) is the sum of the power consumption associated with all the employed communication links. The total power consumption affects the cost of operation of the network and hence, can be treated as a resource.

One can naturally associate a graph with the network of backbone UAVs, with the UAVs serving as nodes, communication links being edges and a weight, α_{ij} associated with the communication link between the i^{th} and j^{th} UAVs. The desirable attributes of a communication network are: lower diameter so as to minimize latency in communicating data/information across the network, high isoperimetric number so that the bottlenecking in a network can only occur at higher data rates and robustness to node and link failures. It is known that a higher value of algebraic connectivity of a network is associated with a network with the previously mentioned desirable attributes[6]. In relation to graph theory, algebraic connectivity provides a measure of how weakly any subset of vertices is connected to the remaining graph. In this measure, a subset of vertices is considered to be weakly connected if a normalized cut (sum of the number of edges leaving the subset) of the subset has a low value. Essentially, a tightly connected network with a larger normalized cut corresponds to a network with a higher algebraic connectivity. Algebraic connectivity as a measure of network connectivity is also superior to other measures such as the node or the link connectivity of a network ; for example, any (unweighted) spanning tree has a node or a link connectivity of one. On the other hand, it is known that a star network has a higher algebraic connectivity compared to that of any (Hamiltonian) path in the network. A star network, for instance, is considered to be more robust against a random removal of a node in the network as opposed to a path which gets disconnected upon the removal of any intermediate node [6]. For this reason, we pose the network synthesis problem as that of determining the network with the maximum algebraic connectivity over all possible networks satisfying the given resource and operational constraints.

Simply put, a variant of the **BP** that arises in this application is as follows: Given a collection of UAVs which can serve as backbone nodes, how should they be arranged and connected so that

(i) the convex hull of the projections of their locations on the ground spans a prespecified area of coverage,

(ii) the resources such as the total UAV power consumption for maintaining connectivity and the total number of communication links employed are within their respective prescribed bounds, and

(iii) algebraic connectivity of the network is maximum among all possible networks satisfying the constraints (i) and (ii).

Variants of **BP** have recently received attention in the UAV literature, for example, a few of the relevant references are [7], [8], [9], [5], [10]. However, prior to this dissertation, a systematic and computationally efficient method for solving the problem exactly was lacking.

Apart from mechanical systems, similar problems appear in disparate research areas including biomedicine and VLSI circuit design. In biomedicine, of particular relevance is the field of systems biology which aims to study the interplay between proteins, nucleic acids and other cellular components at the global level. In this research area, one is interested in engineering and achieving a desired output by either allowing certain new interactions or disallowing some interactions from taking place. In the simplest form, these interactions may be modeled by systems of coupled ordinary differential equations and in more complicated situations such as cascades of biochemical reactions that need to be controlled, the interactions can be modeled by a system of coupled partial differential equations.

A similar problem is also encountered in VLSI circuit design[11]. Due to steady miniaturization of VLSI devices and a quest for faster communication rates, there are critical performance objectives placed on the design of interconnects [12],[13] between the components of a VLSI circuit including minimization of interconnect delays and signal distortion, minimization of signal delays between time-critical components, minimization of total wire length etc. A fundamental problem in VLSI circuit design [11] deals with designing a suitable network topology (*i.e.*, the interconnects between the components) such that the specified performance objective is realized. The same problem also appears in disparate disciplines such as coding theory[14], image webs

[15], air traffic management [16, 17] and free space optical and communication networks [18],[19].

Solving the **BP**, i.e., finding the optimal network corresponding to the maximum value of augmented algebraic connectivity is non-trivial. It is further compounded by the rapid increase in the size of the problem with an increase in the number of nodes (for example, masses). Even for instances of moderate size involving 8 identical masses, if one were asked to pick only 7 springs to form a connected structure, there are $8^6 \approx 262144$ combinatorial possibilities (for a graph with n masses, there are n^{n-2} connected structures with $n-1$ springs). The difficulty is further accentuated by the non-smooth and nonlinear nature of the objective function. *The focus of the dissertation is to develop numerical algorithms for computing optimal networks as well as for computing sub-optimal networks along with a bound on their suboptimality.*

1.1 A note on Laplacian matrix

A graph G is specified by a set of vertices V, a set of edges $E \subset V \times V$ and a cost function $c : E \to \Re_+$. A graph G is compactly represented as $G(V, E, c)$. Let n denote the cardinality of V and let I_n be the identity matrix of dimension n. Without any loss of generality, we can arbitrarily number the vertices and associate the numbers with the vertices. Let $i, j \in V$ and let e_i, e_j correspond to the i^{th} columns of I_n. If $a, b \in \Re^n$, let $a \otimes b$ denote the tensor product of a and b. Let c_{ij} denote the cost of the edge $\{i, j\}$.

The graph Laplacian of $G(V, E, c)$ is defined as:

$$L := \sum_{e=\{i,j\} \in E} c_{ij}(e_i - e_j) \otimes (e_i - e_j).$$

The component of L in the i^{th} row and j^{th} column is given by L_{ij} and is as follows:

$$L_{ij} = \begin{cases} -c_{ij}, & \text{if } i \neq j,\ \{i,j\} \in E, \\ \sum_{j:\{i,j\}\in E} c_{ij}, & \text{if } i = j, \\ 0, & \text{otherwise} \end{cases}$$

As an example, Laplacian matrix for the graph shown in Figure 1.2(a) is as follows:

$$L = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix} \tag{1.1}$$

There are other variants of Laplacians that are used; this dissertation primarily focuses on the graph Laplacian.

1.1.1 Relationship between Laplace's equation and graph Laplacian

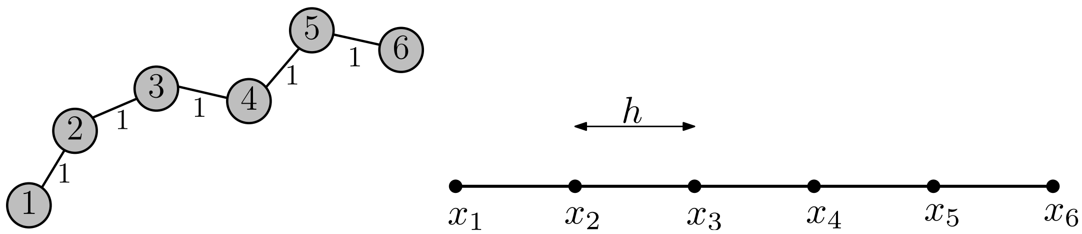
Consider the following one-dimensional Laplace's equation:

$$-\frac{d^2u}{dx^2} = f(x), \tag{1.2}$$

where $f(x)$ is the source, $u(x)$ is the response and $\frac{d^2}{dx^2}(\cdot)$ is the Laplacian operator. As shown in Figure 1.2(b), consider a discretized space such that the given domain $\Omega = [0, X]$ is discretized with equally spaced points $x_i, i = 1 \ldots 6$ and the grid size $h = 1$. Hence, a one-dimensional stencil using a second order central differencing which approximates the Laplacian operator at point x_i is given as follows:

$$\frac{d^2u}{dx^2}\bigg|_{x=x_i} \approx \frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} = u_{i+1} - 2u_i + u_{i-1}$$

where, $u_i \approx u(x_i)$.



(a) A graph with unit weights (b) Finite difference grid

Figure 1.2: A graph with unit weights and its equivalent finite difference grid.

In order to represent the discretized Laplace's equation in the matrix form, we consider the following cases:

Dirichlet boundary condition: For the Laplace's equation in (1.2), let the Dirichlet boundary conditions be as follows:

$$u_1 = u(x_1) = \alpha_1, u_6 = u(x_6) = \alpha_6.$$

Under these boundary conditions, discretizing the Laplace's equation in (1.2) over the grid shown in Figure 1.2(b), we obtain the following matrix form:

$$
\begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{pmatrix} \begin{pmatrix} u_2 \\ u_3 \\ u_4 \\ u_5 \end{pmatrix} = \begin{pmatrix} f_2 - \alpha_1 \\ f_3 \\ f_4 \\ f_5 - \alpha_6 \end{pmatrix} \tag{1.3}
$$

It is evident that the square sub-matrix obtained by dropping the first and the last rows and columns of the Laplacian matrix in equation (1.1) is the same as the coefficient matrix in equation (1.3).

A combination of Neumann and Dirichlet boundary conditions:

For the Laplace's equation in (1.2), let the Neumann boundary condition be:

$$
\left.\frac{du}{dx}\right|_{x=x_1} = \beta_1
$$

and the Dirichlet boundary condition be

$$
u_6 = u(x_6) = \alpha_6.
$$

A natural first order approximation to the derivative at x_1 is a one sided difference

$$
\left.\frac{du}{dx}\right|_{x=x_1} \approx \frac{u_1 - u_2}{h} = \beta_1.
$$

Under these boundary conditions, discretizing the Laplace's equation in (1.2) over the grid shown in Figure 1.2(b), we obtain the following matrix form:

$$\begin{pmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{pmatrix} \begin{pmatrix} u_2 \\ u_3 \\ u_4 \\ u_5 \end{pmatrix} = \begin{pmatrix} f_2 - \beta_1 \\ f_3 \\ f_4 \\ f_5 - \alpha_6 \end{pmatrix} \tag{1.4}$$

It is evident that the square sub-matrix obtained by dropping the the last two rows and columns of the Laplacian matrix in equation (1.1) is the same as the coefficient matrix in equation (1.4).

The graph interpretation of the discretized problem is shown in Figure 1.2(a). In this interpretation, every graph vertex in Figure 1.2(a) can be treated as a grid point; the edges of the graph shown in Figure 1.2(a) have a cost of 1 unit. The finite difference stencil at the grid point can be treated as the local Laplacian matrix and the unit edge cost corresponds to the homogeneous material with a unit thermal conductivity in the case of heat conduction equation.

1.1.2 Graph Laplacian and electrical systems

Consider a simple electrical network with four resistors and five junctions as shown in Figure 1.3. In graph theoretic terms, the junctions represent the vertices of the graph, the resistors represent the edges in the graph and the corresponding conductance values represent the edge weights. For the convenience in the notation, we describe each resistor by it's conductance values, which is the inverse of its resistive values. As an example, if the resistance between vertices two and five is $\frac{1}{c_{25}}\Omega$, then it's conductance value is equal to $c_{25}\mho$.

The problem of interest is to find the voltages V_1, V_2, V_3, V_4 and V_5 at all the vertices in the electrical resistive network given that I_1 and I_5 units of current enters

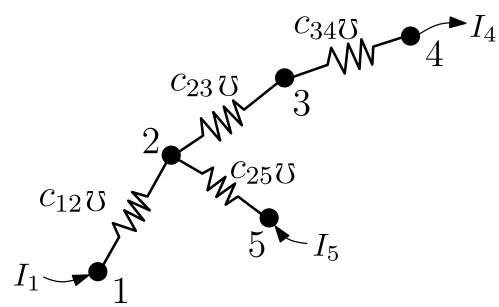


Figure 1.3: An electric network with resistors labeled by their conductance values (℧)

at vertices one and five respectively and I_4 units of current leaves from the fourth vertex.

From Ohm's law, we know that the current flowing across the edges $1 \to 2$, $2 \to 3$, $5 \to 2$ and $3 \to 4$ are $c_{12}(V_1 - V_2)$, $c_{23}(V_2 - V_3)$, $c_{25}(V_5 - V_2)$ and $c_{34}(V_3 - V_4)$ respectively. By applying Kirchoff's current balance law at all the vertices, we have the following set of linear equations:

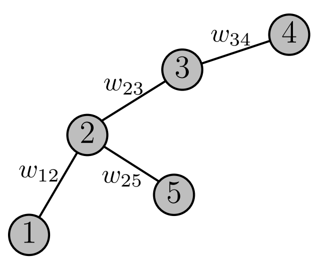
$$
\begin{aligned}
c_{12}(V_1 - V_2) &= I_1, \\
-c_{12}(V_1 - V_2) + c_{25}(V_2 - V_5) + c_{23}(V_2 - V_3) &= 0, \\
-c_{23}(V_2 - V_3) + c_{34}(V_3 - V_4) &= 0, \\
-c_{34}(V_3 - V_4) &= -I_4, \\
-c_{25}(V_2 - V_5) &= I_5.
\end{aligned}
$$

The above system can be expressed in matrix form as follows:

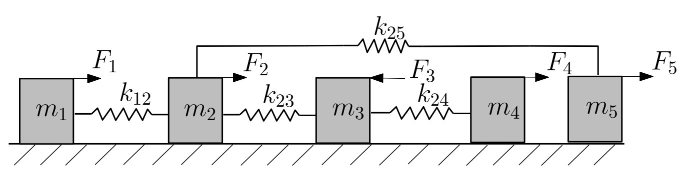
$$\underbrace{\begin{pmatrix} c_{12} & -c_{12} & 0 & 0 & 0 \\ -c_{12} & c_{12}+c_{23}+c_{25} & -c_{23} & 0 & -c_{25} \\ 0 & -c_{23} & c_{23}+c_{34} & -c_{34} & 0 \\ 0 & 0 & -c_{34} & c_{34} & 0 \\ 0 & -c_{25} & 0 & 0 & c_{25} \end{pmatrix}}_{\text{Admittance matrix}} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{pmatrix} = \begin{pmatrix} I_1 \\ 0 \\ 0 \\ -I_4 \\ I_5 \end{pmatrix} \tag{1.5}$$

It can be noted from the above admittance matrix that the $(i,j)^{th}$ entry is the negation of the conductance between the vertices i and j and the i^{th} diagonal entry is the sum of the conductances of all the resistors incident at i^{th} vertex. Hence, this matrix is same as the Laplacian matrix of the weighted graph shown in 1.4(a) and also the stiffness matrix shown in (1.6).

1.1.3 Graph Laplacian and discrete mechanical systems



(a) Weighted graph



(b) Spring mass system

Figure 1.4: A weighted graph and its equivalent form as a spring-mass system.

Consider a simple five degree of freedom vibratory system shown in Figure 1.4(b). At equilibrium, the springs are unstretched. The springs are assumed to be linear with a stiffness coefficient k_{ij} if the spring connects masses m_i and m_j. An application of Newton's laws yields the following governing equations of motion:

$$\begin{pmatrix} m_1 & 0 & 0 & 0 & 0 \\ 0 & m_2 & 0 & 0 & 0 \\ 0 & 0 & m_3 & 0 & 0 \\ 0 & 0 & 0 & m_4 & 0 \\ 0 & 0 & 0 & 0 & m_5 \end{pmatrix} \begin{pmatrix} \ddot{x}_1 \\ \ddot{x}_2 \\ \ddot{x}_3 \\ \ddot{x}_4 \\ \ddot{x}_5 \end{pmatrix} +$$

$$\underbrace{\begin{pmatrix} k_{12} & -k_{12} & 0 & 0 & 0 \\ -k_{12} & k_{12}+k_{23}+k_{25} & -k_{23} & 0 & -k_{25} \\ 0 & -k_{23} & k_{23}+k_{34} & -k_{34} & 0 \\ 0 & 0 & -k_{34} & k_{34} & 0 \\ 0 & -k_{25} & 0 & 0 & k_{25} \end{pmatrix}}_{\text{Stiffness matrix}} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} F_1 \\ F_2 \\ -F_3 \\ F_4 \\ F_5 \end{pmatrix} \tag{1.6}$$

Hence, for the example shown in Figure 1.4(a), the corresponding Laplacian matrix would be:

$$L = \begin{pmatrix} k_{12} & -k_{12} & 0 & 0 & 0 \\ -k_{12} & k_{12}+k_{23}+k_{25} & -k_{23} & 0 & -k_{25} \\ 0 & -k_{23} & k_{23}+k_{34} & -k_{34} & 0 \\ 0 & 0 & -k_{34} & k_{34} & 0 \\ 0 & -k_{25} & 0 & 0 & k_{25} \end{pmatrix} \tag{1.7}$$

Clearly, the stiffness matrix in equation (1.6) is the same as the Laplacian matrix in equation (1.7).

1.2 Algebraic connectivity as an objective of maximization

Since algebraic connectivity is chosen as an objective of maximization in this dissertation, a motivation for the choice of the objective is in order. In this section, the motivation is provided through three different applications where maximizing algebraic connectivity is meaningful.

1.2.1 Linear mechanical systems

Let the mechanical system consist of n identical masses and $|E|$ springs. If masses were to be treated as nodes, the linear springs as edges and stiffness coefficients of the springs as the "weight" associated with each edge (spring), then the algebraic connectivity of the graph corresponds to the smallest non-zero natural frequency of the discrete mechanical system. Let M, L respectively represent the mass and stiffness matrices respectively. The components of L depend on the topology, x, of connections of masses with the aid of springs. Let e_0 denote a vector with every component being unity. If δ, f represent respectively the vectors of displacements and forces acting on the masses, then the governing equations corresponding to a given topology x may be compactly expressed as

$$M\ddot{\delta} + L(x)\delta = f.$$

Let $\|\delta\|_2, \|f\|_2$ represent the 2-norm of δ and f respectively. Let $\mathcal{F} = \{f : \|f\|_2 \leq 1, \ f \cdot e_0 = 0.\}$ The condition $f \cdot e_0 = 0$ implies that the net force acting on the system of masses is zero and hence, the centroid of the masses remains stationary if it is stationary initially.

For a given topology x, let $v_1, v_2, \ldots v_n$ be the unit eigenvectors of $L(x)$ corresponding to eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \lambda_n$.

$$L(x) = \sum_{i=1}^{n} \lambda_i v_i \otimes v_i.$$

Since $L(x)e_0 = 0$, it implies that $\lambda_1 = 0, v_1 = \frac{e_0}{\sqrt{e_0 \cdot e_0}}$. Clearly, when the displacements of all the masses are the same, the deflections in the springs are zero and this eigenvector corresponds to a "rigid-body" mode. If the set of masses is connected, then it will admit only one rigid body mode; in this case $\lambda_2 > 0$. Otherwise, $\lambda_2 = 0$ suggesting that there is another rigid body mode; in this case, one can find two disjoint sets of masses S_1 and S_2 that are not connected by any spring and correspondingly, the governing equations of motion in this case can be recast as:

$$\underbrace{\begin{bmatrix} M_1 & 0 \\ 0 & M_2 \end{bmatrix}}_{M} \underbrace{\begin{bmatrix} \ddot{\delta}_1 \\ \ddot{\delta}_2 \end{bmatrix}}_{\ddot{\delta}} + \underbrace{\begin{bmatrix} L_1(x) & 0 \\ 0 & L_2(x) \end{bmatrix}}_{L(x)} \underbrace{\begin{bmatrix} \ddot{\delta}_1 \\ \ddot{\delta}_2 \end{bmatrix}}_{\delta} = \underbrace{\begin{bmatrix} f_1 \\ f_2 \end{bmatrix}}_{f}.$$

We can construct a force $f \in \mathcal{F}$ as follows: the forces on the masses in S_1 and S_2 are $\frac{1}{\sqrt{n}}\sqrt{\frac{|S_2|}{|S_1|}}$ and $-\frac{1}{\sqrt{n}}\sqrt{\frac{|S_1|}{|S_2|}}$ units respectively. Clearly $\|f\|_2 = 1$ and the sum of the forces acting on the masses is zero. However, the masses in S_1 and S_2 move as if they are independent masses with a constant acceleration. Hence, in this case, the difference in the steady state displacements among the masses is unbounded. Since such a topology of connections is not desirable, let $\mathcal{X}$ denote the topology of connection of masses with springs which admits only a single rigid body mode.

Lemma 1 *Let δ_s be the vector of displacements of masses of the mechanical system due to the forcing function f. If $x \in \mathcal{X}$, and the initial value of average displacement* *and velocity of all masses is zero, then*

$$\max_{f\in\mathcal{F}} \ \|\delta_s\|_2 = \frac{1}{\lambda_2(L(x))}.$$

Proof Since f is a constant force, δ_s is a vector of constants and hence satisfies

$$L(x)\delta_s = f.$$

Let f be decomposed along the eigenvectors $v_2, \ldots, v_n$ as

$$f = \sum_{i=2}^{n} \alpha_i v_i,$$

so that

$$\alpha_j = v_j \cdot f = v_j \cdot L(x)\delta_s = L(x)v_j \cdot \delta_s = \lambda_j v_j \cdot \delta_s.$$

From the assumption that the initial average displacement and velocity of all masses is zero, it follows that the average displacement and velocity of masses is zero throughout as:

$$e_0 \cdot [M\ddot{\delta} + L(x)\delta] = e_0 \cdot f = 0, \Rightarrow e_0 \cdot \ddot{\delta} = 0.$$

Hence, δ_s cannot have a component along v_1 or equivalently along e_0. Since $x \in \mathcal{X}$,

$$\delta_s = \sum_{j=2}^{n} \frac{v_j \cdot f}{\lambda_j} v_j \Rightarrow \|\delta_s\|_2^2 = \sum_{j=2}^{n} \frac{\alpha_j^2}{\lambda_j^2} \leq \frac{\sum_{j=2}^{n} \alpha_j^2}{\lambda_2^2} = \frac{1}{\lambda_2^2}.$$

Since the maximum is achieved when $f = v_2$, it follows that

$$\max_{f\in\mathcal{F}} \|\delta_s\|_2 = \frac{1}{\lambda_2}.$$

■

Clearly, the maximum value of the 2-norm of forced response of the mechanical system can be minimized when $\lambda_2(L(x))$ is a maximum. It is for this reason that algebraic connectivity (or the second smallest eigenvalue of $L(x)$ is maximized.

1.2.2 Application to rigid formations

Consider a formation of n identical UAVs in a single dimension trying to maintain a fixed distance from each other throughout their motion. Suppose the motion of the i^{th} UAV is given by:

$$X_i(s) = \frac{1}{s^2}[P(s)U_i(s) - D_i(s)], \tag{1.8}$$

where $P(s)$ is a proper, rational transfer function, $X_i(s), U_i(s)$ and $D_i(s)$ are respectively the Laplace transformation of the position of, control input to and disturbing force acting on the i^{th} UAV. The term $P(s)$ represents the actuator transfer function and relates the control input to a UAV with the actuation force generated by the UAV.

Suppose the UAVs desire to maintain a constant relative separation with respect to each other with the help of an identical on-board controller represented by the transfer function $C(s)$. Aiding the UAVs in accomplishing this task is a set of communication and sensing devices. An underlying information flow graph indicates the information available to each UAV. For example, if the i^{th} UAV has the position and velocity information of the j^{th} UAV in the collection, the i^{th} and j^{th} UAVs are considered adjacent or neighbors in the information flow graph. For the sake of simplicity of exposition, if i^{th} UAV has the information of j^{th} UAV, it will be assumed that the converse also holds. Let $\mathcal{S}_i$ be the set of neighbors of the i^{th} UAV.

With this set up, one may associate a Laplacian with the information flow graph. For $i \neq j$, the component of Laplacian in the i^{th} row and j^{th} column is -1 if the i^{th} and j^{th} UAVs are neighbors and is zero otherwise. For each i, the i^{th} diagonal element is the number of UAVs, $|S_i|$, that are neighbors of the i^{th} UAV. Clearly, the sum of the components of the corresponding row (and column) is zero.

Let $\bar{X}(s)$ denote the Laplace transformation of the position of the centroid of the formation and let the error in spacing $E_i(s) := X_i(s) - \bar{X}(s) - \frac{L_i}{s}$, where L_i is the desired position of the i^{th} UAV from the centroid. The input to the controller is the aggregate error in maintaining a desired spacing relative to its neighbors and may be described as:

$$U_i(s) = -C(s) \sum_{j \in S_i} [X_i(s) - X_j(s) - \frac{L_i - L_j}{s}] = -C(s) \sum_{j \in S_i} [E_i(s) - E_j(s)]. \quad (1.9)$$

In this case, the error evolution equation can be described by:

$$[s^2 I_n + P(s)C(s)L]E(s) = -D(s) - s^2 \bar{X}(s) - s\ell, \quad (1.10)$$

where $E(s), D(s)$ are respectively the Laplace transformation of the vector of errors in spacing and disturbing forces acting on the UAVs. The term ℓ represents the vector of desired distances of the UAVs from the centroid and L represents the Laplacian associated with the information flow graph.

The associated characteristic equation is given by

$$\Pi_{i=2}^{n} (s^2 + \lambda_i(L)P(s)C(s)) = 0.$$

The stability of motion of the formation of UAVs is governed by the eigenvalues of the Laplacian. Clearly, $P(0)C(0) \neq 0$; otherwise, the errors do not decay to zero as 0 is one of the roots of the characteristic equation. The term $P(0)C(0) > 0$ indicating that the steady state gain from the aggregate error to the force supplied by the actuation system is positive. Since λ_i are the non-zero eigenvalues of the Laplacian, $\lambda_i > 0$. The high frequency gain is positive as it is the coefficient of s^2. If $P(0)C(0) < 0$, then for $\lambda > 0$, the highest and lowest degree terms of the characteristic polynomial will be of opposite signs indicating instability of motion. Physically, if $P(0)C(0) < 0$ then a UAV lagging behind will lag further behind and the motion of the UAVs will be unstable. Since $P(0)C(0) > 0$, the sensitivity to low frequency disturbances is higher if λ is lower. This can be seen as follows: Let $E_v(s) := v \cdot E(s), D_v(s) := v \cdot D(s)$, where v is an eigenvector of Laplacian corresponding to one its non-zero eigenvalues, λ. Then:

$$v \cdot [s^2 I_n + P(s)C(s)E(s)] = -v \cdot D(s) - s(v \cdot \ell),$$

which simplifies to

$$(s^2 + \lambda P(s)C(s))E_v(s) = -D_v(s) - s(v \cdot \ell).$$

Correspondingly, the disturbance attenuation transfer function is given by $\frac{-1}{s^2+\lambda P(s)C(s)}$. At a low frequency, w, the low frequency attenuation is governed by $|\frac{1}{-w^2+\lambda P(0)C(0)}|$ since $P(0)C(0) > 0$. Clearly, higher the value of λ, the better is the attenuation. Since disturbance attenuation at low frequencies is important, it is reasonable to maximize λ_2, the lowest non-zero eigenvalue of the Laplacian.

1.2.3 Application to UAV network synthesis

Earlier in this section, a variant of **BP** involving the construction of an adhoc infrastructure network with UAVs has been alluded to. It required the maintenance of a rigid formation and a control law for maintaining such a formation can be constructed along the lines mentioned in the earlier subsection. The networking aspect of this problem also has relevance to the maximization of algebraic connectivity of the underlying communication graph. Since UAVs form a backbone network, each UAV must be able to transmit data at a constant rate, say $r\ bits/sec$ to every other UAV in the network; however, each UAV may be receiving data at a rate of $R\ bits/sec$ to be transmitted to other UAVs. In order to maintain a desirable quality of service, one may be interested in finding out the maximum value of R that is allowable so that the network is not congested or "bottlenecked"; from the viewpoint of designing a network, one would be interested in designing a UAV adhoc network so that R is maximized subject to other constraints on resources.

One may associate a vertex, v, with each UAV, an edge, e with a communication link and a "weight" α_{ij} associated with the edge (communication link) connecting the i^{th} and j^{th} UAVs. Let V, E represent the set of vertices and edges and let the underlying communication graph be represented as $G(V, E, \alpha)$. The term α_{ij} is proportional to the product of the antenna gains of the i^{th} and j^{th} UAVs and is reflective of the data rate that can be communicated across the link. An important concept in addressing this issue is the value of a cut. A cut may be identified by a set $S \subset V,\ S \neq V$. The cut $\delta(S)$ is the set of edges with exactly one end in S. The value of the cut is the sum of the weights of the edges in $\delta(S)$ and is represented by $w(\delta(S))$. Let $\bar{S}$ be the complement of S in V; i.e., $v \notin S \iff v \in \bar{S}$. Clearly, $\delta(S) = \delta(\bar{S})$ and the value of the cut is the same.

Suppose the network must be designed so that there is a guaranteed data rate of R bits/second that every UAV can transmit without the network getting congested. Given a set $S \subset V$ and its complement $\bar{S} \subset V$, let $r(S)$ be the pairwise data exchange between every pair of nodes in S and $\bar{S}$. Then, the total data transmission across the cut of S is $r(S)|S||\bar{S}|$ and must be no more than the value of the cut, $w(\delta(S))$ and hence:

$$r(S) \leq \frac{w(\delta(S))}{|S| \cdot |\bar{S}|}.$$

If S is of lower cardinality than $\bar{S}$, then nodes in S can transmit data at a rate of $r(S)$ to each node in $\bar{S}$ and consequently,

$$r(S)\max\{|S|, |\bar{S}|\} \leq \frac{w(\delta(S))}{|S| \cdot |\bar{S}|}\max\{|S|, |\bar{S}|\} = \frac{w(\delta(S))}{\min\{|S|, |\bar{S}|\}}.$$

Since

$$R = \min_{S \subset V} \ r(S)\max\{|S|, |\bar{S}|\},$$

it follows that

$$R \leq \min_{S \subset V} \frac{w(\delta(S))}{\min\{|S|, |\bar{S}|\}}.$$

In fact, R can be set to the minimum value of the right-hand side, which is referred to as the Cheeger constant or Cheeger number, h, or the isoperimetric number of a graph. If a link (or an edge) in $\delta(S)$ were to fail, the capacity of a cut decreases and the above inequality relates how the guaranteed data rate of transmission is reduced for every UAV in the network. In the problem of UAV network synthesis, it is desirable to maximize R over all allowable ways of connecting the UAVs.

The Cheeger number is difficult to compute for a sufficiently large size graph; for this reason, algebraic connectivity of a graph is used as its surrogate. The justification for using algebraic connectivity, λ_2, of a graph as a surrogate stems from the following inequality connecting Cheeger constant and algebraic connectivity:

$$h \leq \lambda_2 \leq \frac{h^2}{4}.$$

Clearly, if h is small, λ_2 is small because of the upper bounding inequality and if h is large, λ_2 is also large because of the lower bounding inequality.

1.2.4 Application to air transportation networks

The fundamental objective of an air transportation network is to transport passengers from one airport to another in as efficient a manner as possible while meeting quality of service requirements of the passengers. In this dissertation, a very simple model of an air transportation network will be considered, where every airport serves the role of a node, a *direct* route between two airports serves the role of an edge between the two nodes. The main issue of interest in this dissertation is the sensitivity of "connectivity" of air transportation network to some edges being not operational due to a variety of factors such as weather etc. Here, the term "connectivity" is used loosely. Connectivity is meant to mean the ability to transport the passengers from their respective origin to their intended destination. Connectivity can be affected by weather resulting in an edge connecting two nodes being deleted (i.e., the route being out of operation temporarily or a flight being cancelled). For example, a reduction in connectivity can result in passengers being stranded at an airport and the undesirable consequence of reduction in the quality of service to the passengers.

Given that the motivation is to address connectivity, each edge will be "weighed" according to the passengers that *can be* transported across that edge or even more simply as the number of passenger flights flying that *can be flown* on the route in a day. This is a gross simplification; however, this is a first step towards a more complicated model of an air transportation network. The underlying assumption is that each flight carries the same number of passengers. Since the quality of service is associated with the passengers transported from a node, let $R(i)$ be the total number of passengers transported from node i. It is assumed that the travel demand from node i to all other nodes is the same, i.e., the number of passengers desiring to travel from node i to any other node is exactly the same. The *node capacity* of the network may be defined to the maximum value of $\min_i R(i)$ for which the network is congested, i.e., for some cut, $S \subset V$, the value of the cut equals the travel demand across S, that is, the number of passengers starting from S and intending to reach some node in $\bar{S}$.

A preliminary design of an air transportation network can be posed as follows: Suppose a graph, $G(V, E, w)$ of the nodes/vertices (airports), the set of edges E (routes connecting a pair of airports) and the associated weights. Suppose further that the cost of operating a route is known a priori; one may even associate a priority/importance of the route as a cost. The problem is to find a network so that the minimum serving capacity (i.e., the number of passengers that can be transported from any node in the network) is maximized without the network getting congested and the sum of cost of operation of the routes is within a specified budget.

This problem is analogous to the UAV adhoc infrastructure network design problem, where the objective is to maximize the Cheeger number of the network subject to resource constraint. Since Cheeger number is difficult to deal with, one can pose the closely related problem of maximizing algebraic connectivity subjected to resource constraints.

1.3 Literature review

1.3.1 Relation to current state of knowledge in system theory

The problem of system realization considered in this dissertation was the topic of a plenary talk by Kalman in an IFAC meeting [20] in 2005. While the relevant reference to this work is [21], there has not been a formally written problem statement to this effect to the best of the knowledge of the author. Neither has there been a resolution of the problem.

It is known that the algebraic connectivity of a structure is non-zero if and only if the structure or graph is connected [22], [23], [24]. The problem of maximum algebraic connectivity has been considered in [25] for reducing the heights of the water columns at the junction in a network of pipes connecting them. The relevance of the maximum algebraic connectivity to mixing of Markov Chains is shown in [26]. The work in [26] is concerned only with unweighted graphs and provides bounds on the maximum algebraic connectivity by exploiting the symmetry of the Laplacian under the action of permutations. This problem is also relevant to information flow and motion planning of UAVs as considered in the works [7], [9]. However, none of them solve the mixed integer semi-definite program. Recently, for the special case of the maximum augmented algebraic connectivity problem where only one edge must be added, a bisection algorithm has been presented in [27].

The problem of maximizing augmented algebraic connectivity was considered by Maas [25]; however, a systematic procedure to solving for the maximum augmented algebraic connectivity is still lacking. It may be posed compactly as a mixed-integer, semi-definite program; initial efforts to compute the upper bounds of the maximum algebraic connectivity (which is the second smallest natural frequency in structural systems) may be found in [26] and also in the recent work of the authors [10].

1.3.2 Relation to current state of knowledge in discrete optimization

The problem of determining whether one can construct a constant factor approximation algorithm for this problem is still open. From the viewpoint of constructing cuts for the semi-definite integer programs, Atamturk and Narayanan recently developed non-linear cuts for conic programs[28], [29]. Since conic programs are special instances of semi-definite programs, the general problem of constructing efficient cuts for semi-definite programs is still open. The recent work in [30] develops efficient interior-point algorithms for infinite linear programs. Their work was motivated by the need to solve mixed-integer, semi-definite programs through polyhedral approximations. The book on convex optimization [31] provides an excellent overview of the algorithms required to solve linear semi-definite programs.

1.4 Summary of contributions

In the context of the problem of maximizing the algebraic connectivity of networks under resource constraints, our contributions, as presented in sections 2 and 3, are as follows:

a) Understanding the relevance of **BP** in the context of disparate fields of research and providing algorithms for solving **BP** to optimality, methods to obtain upper bounds and quick heuristics to obtain sub-optimal solutions.

b) Providing algorithms for solving a variant of **BP** that arises in the synthesis of robust UAV communication networks under resource constraints such as the total number of communication links and the diameter of the network.

c) Providing algorithms for solving a second variant of **BP** that arises in synthesizing robust UAV communication networks under resource constraints such as the total number of communication links and the power consumption constraint.

1.4.1 Organization of the dissertation

This dissertation is organized as follows:

In section 2, we pose the problem of maximizing the algebraic connectivity as three equivalent formulations; Mixed Integer Semi-Definite Program (MISDP), MISDP with connectivity constraints and the Fiedler vector formulation as an MILP and discuss the relative strengths and useful features of the proposed formulations.

Further, we study the importance of the choice of an appropriate family of finite number of vectors used to relax the semi-definite constraint and discuss the quality of the associated upper bounds due to the relaxation.

In section 2, we propose three cutting plane based algorithms to solve the proposed MISDP to optimality, namely: an algorithm based on the polyhedral approximation of the semi-definite constraint, an iterative primal-dual algorithm that considers the Lagrangian relaxation of the semi-definite constraint and an algorithm based on the Binary Semi-Definite Program (BSDP) approach in conjunction with cutting plane and bisection techniques. Further, by an improved relaxation of the semi-definite constraint, we discuss the computational efficacy of the cutting plane algorithm in comparison with the state-of-the-art MISDP solvers in Matlab. Also, by adopting the BSDP approach and implementing the algorithm in CPLEX, we discuss the gain in the computation time. Section 2 concludes with heuristics to synthesize feasible solutions for the **BP**. The proposed heuristics are based on neighborhood search, namely k-opt and an improved k-opt heuristic with a reduced search space. We corroborate the quality of the heuristic solutions with respect to optimal solutions for small instances and present the numerical results for large instances (up to sixty nodes).

In section 3, we mathematically formulate various resource constraints such as the diameter constraint and the power consumption constraint. For the problem of maximizing algebraic connectivity under these resource constraints, we propose modified versions of the cutting plane algorithms and discuss their computational performance for relatively small instances. In the context of the problem with power consumption constraint, we extend the BSDP approach to obtain feasible solutions and discuss the quality of the associated lower bounds for relatively large instances (up to ten nodes). Finally, we discuss the performance of the k-opt heuristic in the context of solving **BP** under resource constraints.

Lastly, in section 4, we summarize the results of the work and discuss possible directions to further develop this field of research.

2. *ALGORITHMS FOR THE MAXIMIZATION OF ALGEBRAIC CONNECTIVITY

In this section, the **BP** of maximizing the algebraic connectivity of graphs is considered. The rationale for considering algebraic connectivity of graphs as an objective for optimization is illustrated via various applications, including a discrete mechanical system, air transportation network and UAV ad-hoc networks. Since the computation of solutions for combinatorial problems can be sensitive to mathematical formulation of the problem, different mathematical formulations of **BP** are presented along with their features. The rest of the section is focused on (1) developing upper bounds for the optimal value of algebraic connectivity using relaxation and cutting plane techniques, (2) developing techniques for computing the optimal value of algebraic connectivity and (3) to provide heuristic techniques for synthesizing suboptimal graphs along with the percentage deviation of their algebraic connectivity from the optimal value.

2.1 Problem formulation for the basic problem of maximizing algebraic connectivity

Let (V, E, w) represent a graph. Without any loss of generality, we will simplify the problem by allowing at most one edge to be connected between any pair of nodes

*Reprinted with permission from 1) Algorithms for synthesizing mechanical systems with maximal natural frequencies by H.Nagarajan, S.Rathinam, S.Darbha and K.R.Rajagopal, 2012. *Nonlinear Analysis: Real World Applications*, 13(5):2154-2162, Copyright 2012 by Elsevier Ltd.
2. Synthesizing robust communication networks for UAVs by H.Nagarajan, S.Rathinam, S.Darbha and K.R.Rajagopal, 2012. *American Control Conference, 2012.* 3730–3735, Copyright 2012 by IEEE.
3. Heuristics for synthesizing robust networks with a diameter constraint by H.Nagarajan, P.Wei, S.Rathinam and D.Sun, 2014. *Mathematical Problems in Engineering*, Copyright 2014 by Hindawi Publishing Corporation.

in the graph. Let w_{ij} represent the edge weight with the edge $e = \{i, j\} \in E$ and let $x_{ij} \in \{0, 1\}$ represent the choice variable for every $\{i, j\} \in E$. Let x be the vector of choice variables, x_{ij}. If $x_{ij} = 1$, it implies that the edge is chosen in the construction of the network; otherwise, it is not. In the context of UAVs, vertices of the graph correspond to UAVs and the edges correspond to the communication links between them. The edge weight corresponds to the strength of the communication link between a given pair of UAVs.

If v_1, v_2 are two vectors in the same vector space, we denote their tensor product by $v_1 \otimes v_2$ and their scalar or dot product by $v_1 \cdot v_2$. If $e = \{i, j\}$ connects UAVs i and j, then the effective communication between that pair of UAVs may be expressed as $w_{ij}x_{ij}$. Let e_i denote the i^{th} column of the identity matrix I_n of size $|V| = n$. We may define $L_{ij} = w_{ij}(e_i - e_j) \otimes (e_i - e_j)$, and correspondingly, the weighted Laplacian matrix (in the remainder of this dissertation, the usage of Laplacian matrix implies weighted Laplacian matrix unless specified) may be expressed as:

$$L(x) = \sum_{i<j, \{i,j\} \in E} x_{ij} L_{ij}.$$

Note that, for a given connected network, $L(x)$ is a symmetric, positive semi-definite matrix, that is,

$$v \cdot L(x)v \geq 0 \ \forall v.$$

Let $(\lambda_1(L(x)) = 0) < \lambda_2(L(x)) \leq \lambda_3(L(x)) \ldots \leq \lambda_n(L(x))$ be the eigenvalues of $L(x)$ and let $v_1, v_2, \ldots v_n$ be the corresponding eigenvectors of $L(x)$.

The **BP** can be expressed as:

$$\begin{aligned} \gamma^* = \quad & \max \lambda_2(L(x)), \\ \text{s.t.} \quad & \textstyle\sum_{i<j,\ \{i,j\} \in E} x_{ij} \leq q, \\ & x_{ij} \in \{0,1\}^{|E|} \end{aligned} \tag{2.1}$$

where q is some positive integer which is an upper bound on the number of edges to be chosen. Since this is a non-linear binary program, it is paramount to develop efficient ways of formulate this problem. In the remainder of this section, we shall focus on developing various equivalent formulations for **BP**.

2.1.1 Mixed integer semi-definite program

The **BP** formulation in (2.1) may be equivalently expressed as a Mixed Integer Semi-Definite Program (MISDP) as follows: let $e_0 = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} e_i$ so that $e_0 \cdot e_0 = 1$. Then, formulation (2.1) may be expressed as:

$$\begin{aligned} \gamma^* = \quad & \max \gamma, \\ \text{s.t.} \quad & \textstyle\sum_{i<j,\ \{i,j\} \in E} x_{ij} L_{ij} \succeq \gamma (I_n - e_0 \otimes e_0), \\ & \textstyle\sum_{i<j,\ \{i,j\} \in E} x_{ij} \leq q, \\ & x_{ij} \in \{0,1\}^{|E|}. \end{aligned} \tag{2.2}$$

We will refer to it as the formulation $\mathcal{F}_1$. We first show that this formulation correctly solves the algebraic connectivity problem.

Lemma 2 *Let an optimal solution corresponding to the formulation $\mathcal{F}_1$ be γ^* and x^*. Then, x^* is a network that solves* **BP** *to optimality with γ^* being the second eigenvalue of $L(x^*)$.*

Proof Let the eigenvalues of the positive semi-definite matrix, $L(x)$ be given by $(0 = \lambda_1(L(x))) < \lambda_2(L(x)) \leq \ldots \leq \lambda_n(L(x))$. We first show that for any connected network x, $L(x)$ and $\gamma = \lambda_2(L(x))$ satisfy the constraints in the formulation $\mathcal{F}_1$. Let e_0 be an eigenvector corresponding to $\lambda_1(L(x)) = 0$. Then, $L(x)$ admits a spectral decomposition of the form

$$L(x) = \sum_{i=1}^{n} \lambda_i(L(x)) \ (v_i \otimes v_i), \tag{2.3}$$

where v_i is the eigenvector corresponding to eigenvalue, $\lambda_i(L(x))$. Since $\lambda_1(L(x)) = 0$, and $v_1 = e_0$, the equation (2.3) reduces to

$$L(x) = \sum_{i=2}^{n} \lambda_i(L(x)) \ (v_i \otimes v_i). \tag{2.4}$$

Adding $\lambda_2(L(x)) \ (e_0 \otimes e_0)$ to both sides of the equation (2.4),

$$L(x) + \lambda_2(L(x)) \ (e_0 \otimes e_0) = \lambda_2(L(x)) \ (e_0 \otimes e_0) + \sum_{i=2}^{n} \lambda_i(L(x)) \ (v_i \otimes v_i). \tag{2.5}$$

Since $\lambda_i(L(x)) \geq \lambda_2(L(x)), \ \forall \ i \geq 2$, equation (2.5) reduces to the following inequality:

$$L(x) + \lambda_2(L(x)) \ (e_0 \otimes e_0) \succeq \lambda_2(L(x)) \underbrace{\left(\sum_{i=1}^{n} (v_i \otimes v_i) \right)}_{I_n}, \tag{2.6}$$

$$L(x) \succeq \lambda_2(L(x)) \ (I_n - e_0 \otimes e_0). \tag{2.7}$$

Therefore, for any connected network x, $L(x)$ and $\gamma = \lambda_2(L(x))$ satisfy the constraints in the formulation $\mathcal{F}_1$. Now, to show that $\gamma^* = \lambda_2(L(x^*))$, it is enough to prove that $\gamma^* \geq \lambda_2(L(x^*))$ and $\gamma^* \leq \lambda_2(L(x^*))$.

Proof for $\gamma^* \geq \lambda_2(L(x^*))$: We know that x^* is a feasible solution to formulation $\mathcal{F}_1$ with second eigenvalue, $\lambda_2(L(x^*))$. Since this is a maximization problem, γ^* must be an upper bound on $\lambda_2(L(x))$ for all possible feasible solutions. Hence, $\gamma^* \geq \lambda_2(L(x^*))$.

Proof for $\gamma^* \leq \lambda_2(L(x^*))$: Since (x^*, γ^*) is a feasible solution, we have

$$L(x^*) \succeq \gamma^*(I_n - e_0 \otimes e_0). \tag{2.8}$$

Let $\hat{v}$ be any unit vector perpendicular to e_0. Then

$$\hat{v} \cdot L(x^*)\hat{v} \geq \gamma^*. \tag{2.9}$$

Hence, from the Rayleigh quotient characterization of the second eigenvalue, it follows that $\lambda_2(L(x^*)) \geq \gamma^*$. ■

Approximations of the feasible set of $\mathcal{F}_1$: One can approximate the feasible set of $\mathcal{F}_1$ in at least two different ways:

(a) *Binary relaxation*: In this type of relaxation, the feasible set of the formulation $\mathcal{F}_1$ is expanded by replacing the integer constraint, $x_{ij} \in \{0,1\}^{|E|}$ with $0 \leq x_{ij} \leq 1, \quad \forall i < j, \{i,j\} \in E$.

(b) *Relaxation of the semi-definite constraint*: The semi-definite constraint can be equivalently expressed as a family of linear inequalities parameterized as follows:

$$v \cdot \left(\sum_{i<j,\ \{i,j\} \in E} x_{ij} L_{ij} - \gamma (I_n - e_0 \otimes e_0) \right) v \geq 0 \ \ \forall v.$$

where v is any unit vector. One can relax the semi-definite constraint by picking a finite number of unit vectors, say $v_1, v_2, \ldots, v_N$, and replacing the semi-definite constraint with the following linear inequalities:

$$v_k \cdot \left(\sum_{i<j,\ \{i,j\} \in E} x_{ij} L_{ij} - \gamma (I_n - e_0 \otimes e_0) \right) v_k \geq 0 \ \ \forall k = 1, \ldots, N.$$

Naturally, by solving formulation $\mathcal{F}_1$ with either of these relaxations, we are guaranteed to obtain an upper bound on the maximum algebraic connectivity. Hence, in the remainder of this section, we discuss the quality of the upper bounds obtained by considering the binary relaxation of formulation $\mathcal{F}_1$ and its variants. Later, in section 2.2, we discuss the quality of the upper bounds based on the relaxation of the semi-definite constraint with a finite number of unit vectors without relaxing the binary constraints.

Performance of formulation $\mathcal{F}_1$: For the purposes of implementation, we restrict our feasible solutions to a set of undirected spanning trees since they serve as minimally connected structures. Hence, we solve the following version of formulation $\mathcal{F}_1$.

$$\begin{aligned} \gamma^* = \quad & \max \gamma, \\ \text{s.t.} \quad & \textstyle\sum_{i<j,\ \{i,j\} \in E} x_{ij} L_{ij} \succeq \gamma (I_n - e_0 \otimes e_0), \\ & \textstyle\sum_{i<j,\ \{i,j\} \in E} x_{ij} = n - 1, \\ & x_{ij} \in \{0, 1\}^{|E|}. \end{aligned} \tag{2.10}$$

From here on, we will prefix a formulation with $\mathcal{R}$ to indicate the relaxation of binary constraints (i.e., replacing the constraint $x_{ij} \in \{0,1\}^{|E|}$ with $0 \leq x_{ij} \leq 1, \quad \forall i < j, \{i,j\} \in E$) associated with the formulation. Note that the feasible solutions of $\mathcal{F}_1$ are also feasible for $\mathcal{RF}_1$; the optimal value of $\mathcal{RF}_1$ is an upper bound represented by $\gamma^*_{RF_1}$. A summary of $\mathcal{RF}_1$'s solutions for various problem sizes is shown in Table 2.1. It is clear from the table that the percentage deviation of the upper bound ($\gamma^*_{RF_1}$) from the best known feasible solution is unsatisfactory even for problems with small sizes (103.2% gap for five nodes problem). Also, one can observe that the percent deviation of the upper bound increases with the size of the problem (maximum gap up to 181.9% for twelve nodes problem), which is an undesirable feature. However, having formulations with better upper bounds due to binary relaxations are useful which can in turn reduce the computational time of the Branch and Bound (B&B) solver for solving the problem to optimality. For example, any B&B solver requires upper bounds on the optimal γ^* and one of the ways it generates this bound is by relaxing the binary constraint on x_e.

Also, since the binary relaxation of $\mathcal{F}_1$ allows for fractional values of x_e, it can violate the following fundamental property of connectivity: If S is a strict subset of V, then there must be at least one edge between the set of nodes in S and $V-S$. The relaxation sometimes allows the sum of the fractional values of the edges between S and $V-S$ to be less than unity. As an example, for a random cost matrix (Appendix A), a support graph constructed based on the binary relaxation solution of the $\mathcal{RF}_1$ is shown in Figure 2.1. It is clear from the figure that the min-cut ($x_{38} + x_{78}$) value is equal to 0.805 and hence violates the connectivity property.

A natural way to incorporate the connectivity constraints into the formulation is

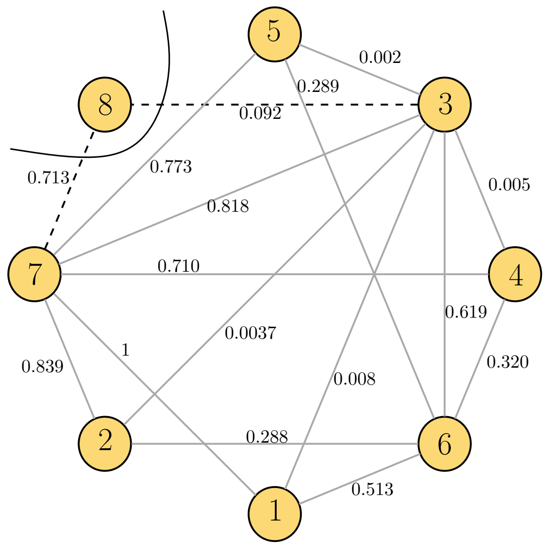


Figure 2.1: Support graph for a random instance where the connectivity constraints are violated. The edges in the violated cutset are shown in dashed lines.

through the augmentation of flow cuts for the violated cutsets of the following form:

$$\sum_{i<j\{i,j\}\in\delta(S)} x_{ij} \geq 1,$$

where $\delta(S)$ represents the edges in the cutset for the which the connectivity requirement is not satisfied. From the implementation point of view, this may not be an efficient way of enforcing connectivity since the number of the violated cutsets may be exponential for large problems. Alternatively, one can also use the flow formulation of Magnanti and Wong [32] to obtain an equivalently strong lifted formulation with a polynomial number of constraints.

In the next subsection, we discuss a compact representation of connectivity using the flow formulation and study the performance of the MISDP with connectivity constraints.

2.1.2 Mixed integer semi-definite problem with connectivity constraints

As we discussed in the earlier subsection, though the formulation $\mathcal{F}_1$ with binary requirements on x_{ij} enforces connectivity, the relaxed problem $\mathcal{RF}_1$ does not ensure connectivity due to the fractional solutions.

MISDP formulation with cutset constraints which enforces the requirement of spanning trees as feasible solutions is as follows:

$$\begin{aligned} \gamma^* = \quad & \max \gamma, \\ \text{s.t.} \quad & \textstyle\sum_{i<j,\ \{i,j\} \in E} x_{ij} L_{ij} \succeq \gamma(I_n - e_0 \otimes e_0), \\ & \textstyle\sum_{i<j,\ \{i,j\} \in E} x_{ij} = n - 1, \\ & \textstyle\sum_{i<j,\ \{i,j\} \in \delta(S)} x_{ij} \geq 1 \quad \forall S \subset V, \\ & x_{ij} \in \{0,1\}^{|E|}. \end{aligned} \tag{2.11}$$

Clearly, if we ignore the integrality restrictions on x_{ij} variables, the fractional solutions still satisfy the connectivity requirements. However, the main drawback of formulation (2.11) is that the number of cutset constraints are exponential in the number of the nodes in the network. Hence, we discuss an alternative formulation based on multicommodity flow model proposed by Magnanti and Wolsey in [32].

In summary, in order to impose the connectivity constraints, the idea of the multicommodity flow model is as follows: Fix any vertex in the graph as a source vertex s. Then construct the network x_{ij} such that a distinct unit commodity is shipped from s to each of the vertices in V while satisfying the flow and capacity constraints. Flow constraints ensure that every distinct commodity indeed reaches its terminal vertex by satisfying the mass balance at every intermediate vertex. Capacity constraints ensure that sa the flow of commodity across an edge occurs only if the capacity of the edge is greater than or equal to the amount of the commodity shipped.

In the multicommodity flow formulation, let f_{ij}^k be the the k^{th} commodity flowing from i to j. In this formulation, although the edge variables are undirected, the flow variables will be directed. Then, the MISDP formulation with multicommodity flow constraints, which we shall refer as $\mathcal{F}_2$ is as follows:

$$\begin{aligned}
\gamma^* = \quad & \max \gamma, \\
\text{s.t.} \quad & \textstyle\sum_{i<j,\ \{i,j\} \in E} x_{ij} L_{ij} \succeq \gamma (I_n - e_0 \otimes e_0), \\
& \textstyle\sum_{j \in V \setminus \{s\}} (f_{ij}^k - f_{ji}^k) = 1, \quad \forall k \in V \text{ and } i = s, \\
& \textstyle\sum_{j \in V} (f_{ij}^k - f_{ji}^k) = 0, \quad \forall i, k \in V \text{ and } i \neq k, \\
& \textstyle\sum_{j \in V} (f_{ij}^k - f_{ji}^k) = -1, \quad \forall i, k \in V \text{ and } i = k, \\
& f_{ij}^k + f_{ji}^k \leq x_{ij}, \quad \forall \ \{i,j\} \in E, \forall k \in V, \\
& 0 \leq \ f_{ij}^k \ \leq 1, \quad \forall i, j \in V, \forall k \in V, \\
& \textstyle\sum_{i<j,\ \{i,j\} \in E} x_{ij} = n - 1, \\
& x_{ij} \in \{0,1\}^{|E|}.
\end{aligned} \tag{2.12}$$

Performance of formulation $\mathcal{F}_2$: Although the MISDP formulation with multicommodity flow constraints ($\mathcal{F}_2$) circumvents the enumeration of exponential number of cutset constraints, the computational performance of this formulation is very poor. With the integrality constraints, state-of-the-art MISDP solvers like Sedumi in Matlab [33] on a reasonably powerful workstation could not handle problems with five vertices and ten edge variables. One of the main reasons for the poor performance is the addition of $O(|V|^3)$ flow variables in addition to the $O(|V|^2)$ edge variables.

By relaxing the integrality constraints on $\mathcal{F}_2$ and solving $\mathcal{RF}_2$, the computational results are summarized in Table 2.1. Again, the performance of $\mathcal{RF}_2$ was very poor and the solvers in Matlab crashed for instances with more than six vertices and fifteen edge variables. For the case of five vertices, there is a slight improvement in the upper bound in comparison with the solution for $\mathcal{RF}_1$.

2.1.3 Fiedler vector formulation

There has been a great deal of interest in developing high performance solvers for solving LPs and MILPs to optimality. Recently, there has also been a progress in the development of efficient programs for solving semi-definite problems and its variants with additional constraints such as polynomial constraints, second order conic constraints, etc. Here is a link to a comprehensive list of state-of-the-art semi-definite solvers: `http://www-user.tu-chemnitz.de/~helmberg/sdp_software.html`. However, there has not been much focus on developing generic solvers for solving MISDP problems. In order to utilize the available high performance MILP solvers, we present an equivalent formulation for $\mathcal{F}_1$ in the form of an MILP and study its performance in this subsection.

We define the following notation before discussing the formulation based on the Fiedler vectors of feasible solutions: Let Γ represent the set of all feasible solutions to formulation $\mathcal{F}_1$ and

$$V_f := \{v \in \mathbb{R}^n : v \text{ is a Fiedler vector for a feasible solution, } x \in \Gamma\}.$$

For the case of spanning trees as feasible solutions, V_f contains the Fiedler vectors corresponding to the n^{n-2} spanning trees.

The Fiedler vector formulation, which we shall refer as $\mathcal{F}_3$ is as follows:

$$\begin{aligned} \gamma^* = \quad & \max \gamma, \\ \text{s.t.} \quad & v \cdot (\textstyle\sum_{i<j,\ \{i,j\} \in E} x_{ij} L_{ij}) v \succeq \gamma, \quad \forall v \in V_f, \\ & \textstyle\sum_{i<j,\ \{i,j\} \in E} x_{ij} \leq q, \\ & x_{ij} \in \{0,1\}^{|E|}. \end{aligned} \tag{2.13}$$

We prove the following lemma to show the equivalence of formulations $\mathcal{F}_1$ and $\mathcal{F}_3$.

Lemma 3 *Let $(x^*_{F3}, \gamma^*_{F3})$ be an optimal solution to $\mathcal{F}_3$ and let $(x^*_{F1}, \gamma^*_{F1})$ be an optimal solution to $\mathcal{F}_1$. Then, $\gamma^*_{F1} = \gamma^*_{F3}$.*

Proof Clearly, the feasible set for $\mathcal{F}_1$ is a subset of the feasible set for $\mathcal{F}_3$ since we replace the original semi-definite constraint with a finite number of constraints in $\mathcal{F}_3$. Hence, we have

$$\gamma^*_{F3} \geq \gamma^*_{F1} = \lambda_2(L(x^*_{F1})).$$

Let v_{F3} represent the Fiedler vector of x^*_{F3}. From the definition of V_f, we know that v_{F3} belongs to the set V_f. However, x^*_{F3} is feasible for $\mathcal{F}_3$. Hence, we have

$$v_{F3} \cdot L(x^*_{F3}) v_{F3} \geq \gamma^*_{F3}.$$

that is,

$$\lambda_2(L(x^*_{F3})) \geq \gamma^*_{F3}.$$

Combining all the inequalities, we have

$$\gamma^*_{F1} = \lambda_2(L(x^*_{F1})) \geq \lambda_2(L(x^*_{F3})) \geq \gamma^*_{F3} \geq \lambda_2(L(x^*_{F1})) = \gamma^*_{F1}.$$

It follows that $\gamma^*_{F1} = \gamma^*_{F3}$. ■

Performance of Fiedler vector formulation: For the implementation purposes, we restrict the feasible solutions of formulation $\mathcal{F}_3$ to undirected spanning trees. Solving $\mathcal{F}_3$ with binary constraints is computationally very inefficient since the number of the constraints are $8^6 + 1$ (262,145) even for the case of eight nodes. Hence, we solve $\mathcal{RF}_3$ by relaxing the binary constraints on x_{ij}. From table 2.1, it is clear that the upper bounds obtained are orders of magnitude higher than the optimal solutions.

Relative strengths of the proposed formulations: Understanding the relative strengths of the formulations is easier by fixing the continuous variable γ to a constant non-negative value, which shall be $\bar{\gamma}$ in all the three formulations. Since $\bar{\gamma}$ is chosen arbitrarily, the results hold true for any γ.

For a given complete graph $G = (V, E)$, let S denote the set of incidence vectors of spanning trees $s^j,\ j = 1, \ldots, (N = |V|^{|V|-2})$. Let $conv(S)$ denote the convex hull of S, that is,

$$conv(S) := \{\sum_{j=1}^{N} \mu_j s^j : \sum_{j=1}^{N} \mu_j = 1, \mu_j \geq 0\ \forall j = 1, \ldots, N\}$$

Since we are interested in undirected edges, we use the following notation for simplicity where x_e represents an edge variable corresponding to edge $e := \{i, j\}$.

Based on our earlier discussion, S can be defined for each of the formulations as follows:

$$S_{F1} := \{x_e \in \{0,1\}^{|E|} : \sum_{e \in E} x_e L_e \succeq \bar{\gamma}(I_n - e_0 \otimes e_0), \sum_{e \in E} x_e = n - 1\},$$

$$S_{F2} := \{x_e \in \{0,1\}^{|E|} : \sum_{e \in E} x_e L_e \succeq \bar{\gamma}(I_n - e_0 \otimes e_0), \sum_{e \in E} x_e = n-1,\ \sum_{e \in \delta(S)} x_e \geq 1\ \forall S \subset V\},$$

$$S_{F3} := \{x_e \in \{0,1\}^{|E|} : v \cdot (\sum_{e \in E} x_e L_e) v \succeq \bar{\gamma}, \sum_{e \in E} x_e = n - 1\}.$$

We have shown that

$$conv(S_{F1}) = conv(S_{F2}) = conv(S_{F3}).$$

Let P_{F1}, P_{F2} and P_{F3} denote the polyhedrons obtained by relaxing the binary constraints on formulations $\mathcal{F}_1$, $\mathcal{F}_2$ and $\mathcal{F}_3$ respectively.

Lemma 4 $P_{F2} \subseteq P_{F1} \subseteq P_{F3}$

Proof A simple argument to prove this lemma is as follows.

From the definition of P_{F1} and P_{F2}, we know that P_{F2} has all the constraints of P_{F1} in addition to the cutset constraints. Hence $P_{F2} \subseteq P_{F1}$.

Based on the definition of positive semi-definite matrices, the semi-definite constraint defining P_{F1} can be replaced with infinite linear constraints, that is,

$$v \cdot \left(\sum_{e \in E} x_e L_e - \bar{\gamma}(I_n - e_0 \otimes e_0) \right) v \geq 0 \quad \forall v.$$

Since this representation is true for any $v \in \mathbb{R}^n$, the set P_{F1} can be relaxed by picking only a finite number of vectors, particularly $v \in V_f$. However, based on the definition, this relaxed set is also P_{F3}. Hence $P_{F1} \subseteq P_{F3}$.

Combining the above two results, we have $P_{F2} \subseteq P_{F1} \subseteq P_{F3}$. ■

The computational results summarizing the strengths of the formulations shown in Table 2.1 also matches well with the above lemma.

Useful features of the proposed formulations:

- Formulation $\mathcal{F}_1$ provides a compact representation of maximizing the algebraic connectivity via it's semi-definite constraint. Since maximizing algebraic connectivity automatically ensures connectedness in graphs, additional connectivity constraints can be omitted.

Table 2.1: Summary of the binary relaxation solutions of proposed formulations. The entries in the table represent the upper bounds due to binary relaxations and their corresponding percent gaps from the best known feasible solution (Best FS). N/A implies that the Matlab's MISDP/MILP solver could not handle those instances. For every n, the values shown are averaged over ten random instances.

		MISDP ($\mathcal{F}_1$)		**MISDP with flow ($\mathcal{F}_2$)**		**Fiedler vector formulation ($\mathcal{F}_3$)**	
n	Best FS	$\gamma^*_{RF_1}$	% gap	$\gamma^*_{RF_2}$	% gap	$\gamma^*_{RF_3}$	% gap
5	10.9751	22.1131	103.2	22.0626	102.8	23.6332	116.5
6	15.7173	33.3133	113.6	N/A	N/A	33.8767	118.1
7	17.6994	42.7876	144.5	N/A	N/A	43.2787	147.4
8	25.5552	56.9862	123.1	N/A	N/A	58.2514	128.1
9	28.0676	77.5151	177.8	N/A	N/A	N/A	N/A
10	38.1984	105.7051	178.5	N/A	N/A	N/A	N/A
12	52.0502	146.2322	181.9	N/A	N/A	N/A	N/A

- As it is, solving $\mathcal{F}_1$ to optimality is computationally inefficient since the available MISDP solvers have limited features. However, by relaxing the semi-definite constraint in $\mathcal{F}_2$ using a finite number of vectors, one can readily have a tighter upper bound by solving the corresponding MILP. Of course, the quality of the upper bound depends on the number and the type of the vectors chosen.

- The multicommodity flow constraints in $\mathcal{F}_2$ come in handy to enforce connectedness in feasible solutions while solving MILP with relaxed semi-definite constraints.

- The Fiedler vectors of feasible solutions used in formulation $\mathcal{F}_3$ can be readily used to relax the semi-definite constraint by choosing a few of the many vectors from the set V_f.

- The solution to $\mathcal{F}_2$ with relaxed semi-definite constraints need not be feasible for

$\mathcal{F}_1$, that is, the solution $(x_e^*, \gamma_{RF_1}^*)$ need not satisfy the semi-definite constraint. This implies that the matrix $\sum_{e \in E} x_e^* L_e - \gamma_{RF_1}^* (I_n - e_0 \otimes e_0)$ will have a negative eigenvalue. Hence, based on the eigenvector corresponding to the negative eigenvalue, one can develop a cutting plane[1] to eliminate the current solution and possibly many other non-optimal solutions. Similarly, a sequence of cutting planes can be generated until an optimal solution is obtained. This summarizes the basic idea of the cutting plane algorithms which are discussed in detail in section 2.3.

In summary, this section has basically dealt with development of three equivalent formulations for the **BP** and summarized the quality of upper bounds and the strengths of formulations obtained by considering their respective binary relaxations. In the next section, we utilize the various features of these formulations as discussed to develop tighter upper bounds and ultimately obtain optimal solutions asymptotically.

2.2 Upper bounds on algebraic connectivity

We discussed in the earlier section that the quality of the binary relaxations for all the three formulations was poor and became worse with an increase in the problem size. In this section, we mainly focus on developing techniques to obtain tight upper bounds for the **BP**. For any spanning tree as a feasible solution, we first develop a method to relax the semi-definite constraint using Fiedler vectors; this relaxation seems to provide better bounds than binary relaxations.

2.2.1 Relaxation of the semi-definite constraint using Fiedler vectors

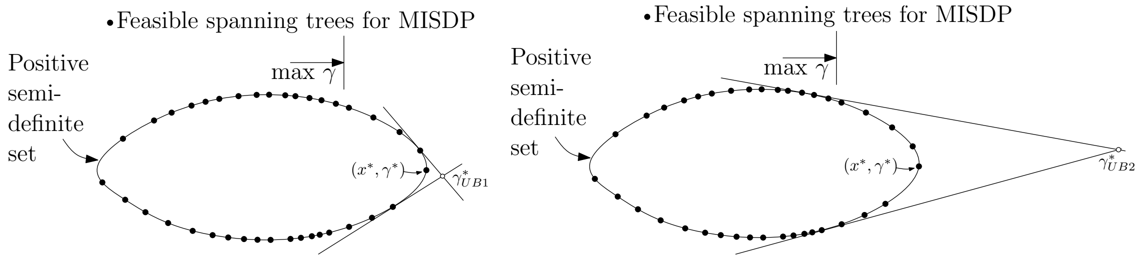
From our earlier discussion in section 2.1.3, we know that the MILP formulation in $\mathcal{F}_3$ is equivalent to solving the MISDP formulation in $\mathcal{F}_1$. However, even for problems

[1]A brief discussion on the concept of cutting planes can be found in section 2.3.

of moderate sizes ($n \geq 8$), it would be impractical to enumerate all the Fiedler vectors of feasible solutions in $\mathcal{F}_3$. However, by considering only a few vectors from the many Fiedler vectors of the set V_f and maintaining the integrality constraints, one can readily obtain upper bounds on the algebraic connectivity due to the relaxation of the feasible set. Earlier, in section 2.1.1, we had briefly alluded to the concept of approximating the feasible set of $\mathcal{F}_1$ by relaxing the semi-definite constraint with finite set of linear inequalities with each inequality identified with an appropriate unit vector. However, in this section, we restrict the relaxation of the feasible set using the Fiedler vectors and discuss the quality of the bounds obtained from such relaxations.

The quality of the upper bound from relaxing the semi-definite constraint using Fiedler vectors depends on the following two factors: 1) Type of feasible solutions whose Fiedler vectors are considered, 2) The number of Fiedler vectors considered. Hence, the main focus of this section will be to construct appropriate Fiedler vectors which provide tight upper bounds and study their quality.

Choosing the type of Fiedler vectors to relax the semi-definite constraint: We observed that the relaxation of the semi-definite constraint with the Fiedler vectors of spanning trees with higher value of algebraic connectivity gives very good upper bounds. A simple, but a rough geometric interpretation of this hypothesis is as follows: From Figure 2.2, it is clear that the relaxation of the feasible semi-definite set with the Fiedler vectors corresponding to the spanning trees with higher algebraic connectivity (γ) gives better upper bound. However, it can also be observed that, without the Fiedler vector corresponding to the optimal solution, γ^*_{UB1} will be strictly greater than γ^* irrespective of the number of Fiedler vectors used for the relaxation. This can also be easily deduced from Lemma 3.



(a) Tighter relaxation

(b) Weaker relaxation

Figure 2.2: A geometric interpretation of the relaxation of the semi-definite constraint using Fiedler vectors of feasible solutions. In (a), the upper bound obtained (γ^*_{UB1}) from Fiedler vectors of spanning trees with higher algebraic connectivity is tighter than the upper bound obtained (γ^*_{UB2}) from Fiedler vectors of spanning trees with lower algebraic connectivity.

Constructing good feasible solutions: A priori, for a given complete weighted graph, we neither know the optimal spanning tree with maximum algebraic connectivity nor the sub-optimal spanning trees with higher algebraic connectivities. However, by enumerating all the spanning trees for small instances, one can observe that the spanning trees with higher algebraic connectivity tend to have larger values of the sum of the weights of the edges in the tree. This trend can be clearly observed in Figure 2.3 for instances with six and seven nodes. It can also be noted from the figure that the maximum spanning tree is not necessarily the spanning tree with maximum algebraic connectivity. However, from Table 2.2, we can see that the tree with maximum algebraic connectivity occurs in the first few thousands (up to eight nodes) while enumerating all the spanning trees in the decreasing order of the sum of the weights of the edges in the tree. The enumeration of all the spanning trees in Figures 2.3(a) and 2.3(b) corresponds to the third and the first instance in Table 2.2 respectively. These are the worst case instances where the optimal spanning tree is farthest from the maximum spanning tree.

Based on these ideas, we now present a systematic procedure to construct the

Fiedler vectors used for relaxation of the semi-definite constraint and discuss the quality of the upper bounds obtained.

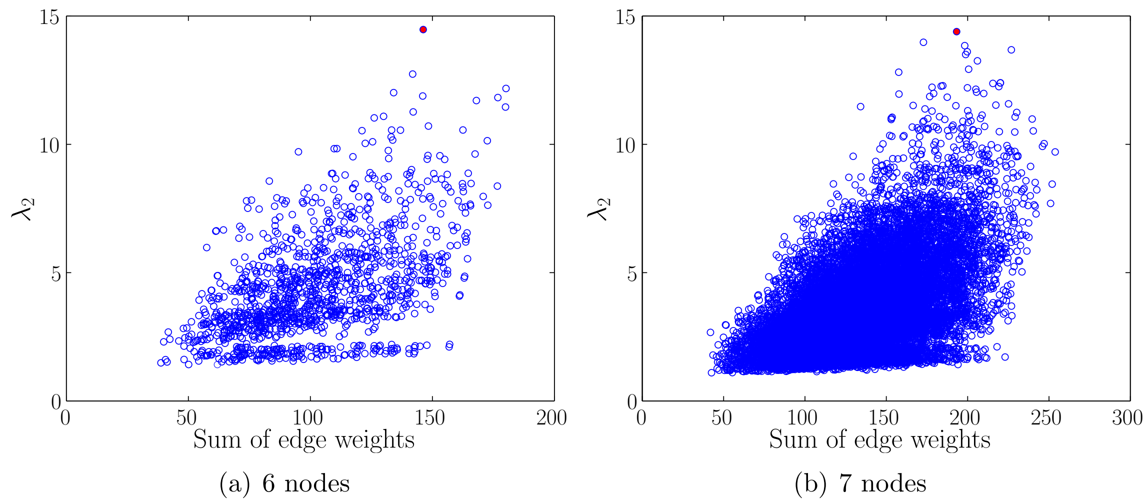


Figure 2.3: Graphical representation of the distribution of algebraic connectivity (λ_2) values for all the spanning trees over a random complete graph. The tree with maximum λ_2 is indicated by the circle filled with red color. It can be observed that the trees with larger values of λ_2 tend to have larger sum of the edge weights.

(a) Enumerate a fixed number of spanning trees in the decreasing order of the sum of the weights of the edges in the tree, where the first tree in the enumerated list will be a maximum spanning tree.

(b) Rank the enumerated spanning trees in the decreasing order of their algebraic connectivity values, that is, the tree with rank one will have the maximum value of algebraic connectivity among the enumerated spanning trees.

(c) Pick a fixed number of the first few ranked spanning trees in the decreasing order of the algebraic connectivity values. Their Fiedler vectors can be used to relax the semi-definite constraint.

Table 2.2: The entries of this table represent the position of the spanning tree with maximum algebraic connectivity (optimal solution) in the enumerated list of spanning trees, where the enumeration is in the decreasing order of the sum of the edge weights.

Instances	**6 nodes**	**7 nodes**	**8 nodes**
1	39	819	5126
2	21	47	530
3	97	48	81
4	2	466	1058
5	3	109	704
6	30	10	9312
7	11	243	398
8	2	3	12805
9	92	189	11991
10	2	312	1225
Average	30	225	4323

Quality of the upper bounds: In order to enumerate a fixed number of spanning trees from the maximum spanning tree, a standard enumeration algorithm for weighted graphs as given in [34] was implemented in Matlab. The optimal spanning tree for up to eight nodes was within 4,323 trees while averaged over ten instances and the worst case being 12,805 as shown in Table 2.2. Hence, from the maximum spanning tree, we enumerated 15,000 spanning trees for every random instance up to twelve nodes. The computation time for enumerating up to 15,000 spanning trees for graphs of sizes up to twelve nodes was less than ten minutes.

The performance of the relaxation of formulation $\mathcal{F}_3$ with various number the Fiedler vectors used for relaxation is shown in Figure 2.4. The percent gap shown in Figure 2.4 is defined as follows:

$$\text{percent gap} = \frac{\gamma^*_{UB} - \gamma_{bfs}}{\gamma_{bfs}} * 100,$$

where γ^*_{UB} is the upper bound obtained by solving the relaxed formulation $\mathcal{F}_3$ and γ_{bfs} is the algebraic connectivity of the best feasible solution known. The best feasible solution in this case will be the spanning tree with maximum algebraic connectivity among the 15,000 enumerated trees.

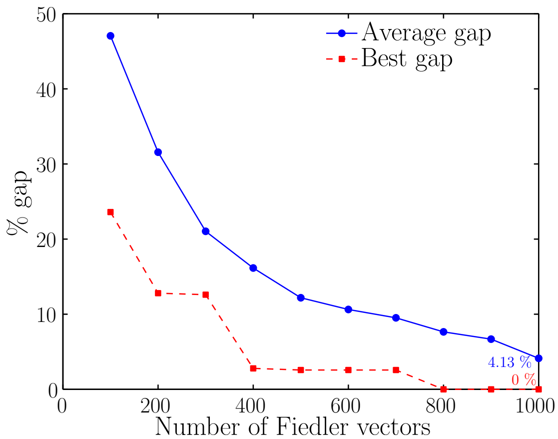
In Figure 2.4, it can be noted that the average percent gap obtained by relaxing with thousand Fiedler vectors for instances of eight, nine, ten and twelve nodes are 4.13%, 42.9%, 65.7% and 116.1% respectively. Though the gaps grow with the increase in problem size, they are orders of magnitude better than the binary relaxation gaps shown in Table 2.1. Also, for the best instances shown in Figure 2.4, the optimal solution was obtained with just eight hundred Fiedler vectors for the case of eight nodes.

2.3 Algorithms for determining maximum algebraic connectivity

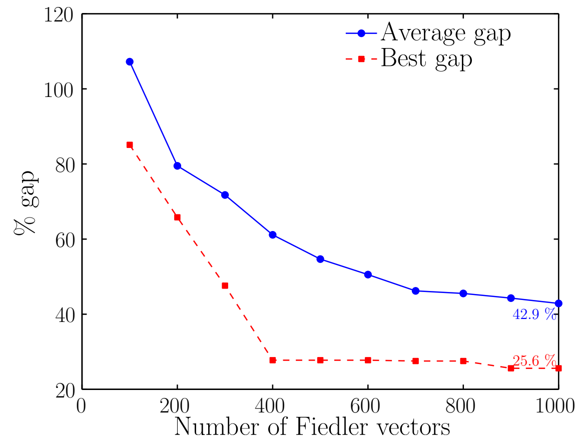
In this section, we focus on developing algorithms based on cutting plane techniques to obtain optimal solutions for the problem of maximizing algebraic connectivity (**BP**).

In principle, the available MISDP solvers in Matlab can be employed to solve the problem $\mathcal{F}_1$ in (2.2). A well known state-of-the-art solver employed for solving MISDPs is the SEDUMI [33] toolbox which can be accessed with the YALMIP user interface [35]. However, even on a powerful workstation, the time to compute an optimal solution using these solvers was in the order of hours for instances involving at most eight nodes and couldn't handle instances involving nine nodes or more. (for 8 and 9 nodes, the number of feasible solutions are 262,144 and 4,782,969 respectively).

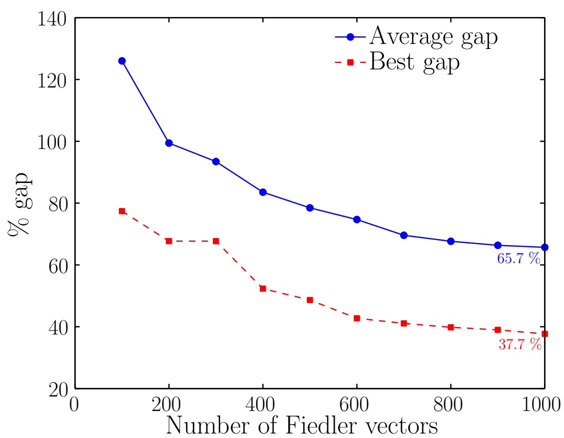
There is a need for developing algorithms that provide tight upper bounds, in case the optimal solution cannot be computed efficiently. The cutting plane technique can be used to provide a monotonically decreasing sequence of upper bounds that



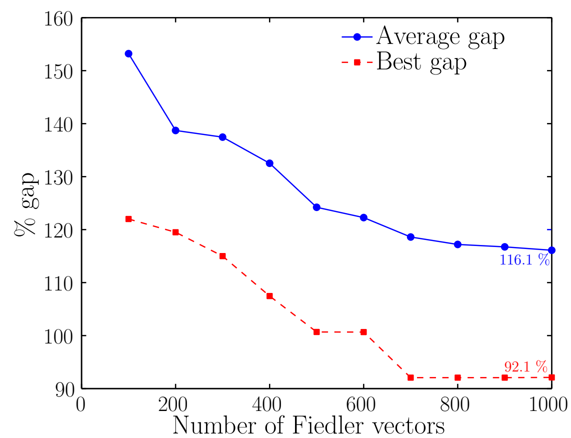
(a) 8 nodes



(b) 9 nodes



(c) 10 nodes



(d) 12 nodes

Figure 2.4: Plot of the percent deviation of the upper bounds obtained by relaxing the semi-definite constraint using the Fiedler vectors of good solutions from the best known feasible solution. Average gap corresponds to the average value evaluated over ten random instances and the best gap corresponds to the instance for which the percent gap was minimum.

converge to the optimal value of algebraic connectivity. In the previous section, we discussed in detail an efficient way to approximate the feasible set using Fiedler vectors of feasible solutions to obtain tight upper bounds. Based on the cutting plane techniques, one can always further tighten the upper bound and eventually obtain optimal solutions. Therefore, after a brief introduction to the concepts of cutting plane techniques, we propose three cutting plane algorithms to solve the problem of maximizing algebraic connectivity to optimality.

2.3.1 Cutting plane techniques

In combinatorial optimization problems, cutting plane method generally refers to an iterative refinement of the feasible set by means of valid linear inequalities or "cuts" or "cutting planes". The procedure of adding cutting planes to obtain optimal solutions are popularly used for solving MILPs. In the early sixties, Gomory in his papers [36, 37] proposed to solve integer programs by using cutting planes, thus reducing an integer programming problem to the solution of a sequence of linear programs. Later, in the early 1990s, Ceria et. al., in their paper [38] introduced a branch-and-cut approach to solve MILPs which effectively combined the usage of Gomory cuts with the branch-and-bound procedure.

Cutting plane method for MILPs with a maximization objective works as follows: Solve the MILP by relaxing the integrality constraints to obtain an easily solvable linear program. Since this is a relaxation, the optimum value obtained will be an upper bound to the original MILP. If the optimal solution obtained for the relaxed MILP is not an integer solution, then there is guaranteed to exist a linear inequality or a "valid inequality" or a "cutting plane" or simply a cut that separates this optimal solution from the convex hull of the feasible set of MILP. Finding such a cutting plane is the "separation problem". An improved relaxation to the MILP can be constructed by adding the cut to the existing relaxation. The linear inequality is satisfied by the optimal solution of the MILP; however, it is not satisfied by the non-integral optimal solution of the relaxed linear program. The optimal value of the relaxed linear program provides a tighter upper bound to the MILP. Solving a sequence of such linear programs with monotonically decreasing upper bound until an integer solution is found is the essence of the "cutting plane method". Having a polynomial time solvable separation problem for any MILP is not trivial. However, in the literature, there are many separation heuristics for specific problems, where these heuristics are not guaranteed to generate cutting planes for every solution of the relaxed MILP.

In this dissertation, we extend the idea of the standard cutting plane method for MILPs to solve the proposed MISDP problem. Instead of relaxing the binary constraints in formulation $\mathcal{F}_1$, we relax the semi-definite constraint using a finite number of Fiedler vectors and solve the corresponding MILP as discussed in the previous section. To enforce connectivity in the feasible solutions for the MILP, we invoke the multicommodity flow formulation as discussed in formulation $\mathcal{F}_2$. Clearly, the solution obtained by solving the MILP need not be feasible for the MISDP since the semi-definite constraint can be violated. Hence, we add a valid inequality which eliminates the current integral solution to obtain an augmented MILP. Solving a sequence of such augmented MILPs terminates when the current solution is also feasible for the MISDP.

In the remainder of this section, we provide a detailed discussion on developing three algorithms based on the cutting plane techniques as discussed above. Firstly, we discuss a cutting plane algorithm, where a sequence of MILPs are solved by relaxing the semi-definite constraint using a finite number of Fiedler vectors. Secondly, we provide a bisection algorithm to reduce the MISDP to a sequence of BSDPs and discuss the gains in the computational efficiency. Thirdly, we discuss an iterative primal-dual algorithm based on the Lagrangian relaxation of the semi-definite constraint.

2.3.2 EA_1: Algorithm to compute maximum algebraic connectivity

EA_1 (EA stands for an algorithm that computes an optimal solution exactly) involves the construction of successively tighter polyhedral approximations of the positive semi-definite set corresponding the maximum algebraic connectivity problem given in formulation $\mathcal{F}_1$.

If one were to store the Fiedler vectors of some feasible solutions (spanning trees), one can relax the semi-definite constraint in $\mathcal{F}_1$ as follows:

$$\sum_{e \in E} x_e Q_i \cdot L_e - \gamma Q_i \cdot (I_n - e_0 \otimes e_0) \geq 0, \;\; i = 1, \ldots, N,$$

where N is the pre-specified number of constraints used in the termination criteria and $Q_i, \; i = 1, 2, \ldots, N$ are the dyads associated with the Fiedler vectors corresponding to the feasible solutions. If one were to directly approach the solution, one may pick a bunch of random feasible solutions and construct the associated Q_i's from their Fiedler vectors. In order to have a tighter initial relaxation of the feasible set, one can also construct the Q_i's from the feasible solutions as discussed in section 2.2.1. At the end of this section, we shall discuss the computational efficiency of EA_1 by choosing such special Fiedler vectors for the relaxation of the semi-definite constraint. One may then perform the following iteration to obtain optimal solution:

1. Solve the following MILP using as follows:

$$
\begin{aligned}
&\max \gamma \\
\text{s.t.}\quad & \sum x_e Q_i \cdot L_e - \gamma Q_i \cdot (I_n - e_0 \otimes e_0) \geq 0, \quad \text{for } i = 1, \ldots, N, \\
& \textstyle\sum_{e \in E} x_e \leq q, \\
& \textstyle\sum_{e \in \delta(S)} x_e \geq 1, \quad \forall\, S \subset V, \\
& x_e \in \{0, 1\}^{|E|}.
\end{aligned}
$$

One may observe that the exponential number of cutset constraints can be replaced with the multicommodity flow formulation as discussed in formulation $\mathcal{F}_2$.

2. Check if the optimal solution x^* to the above MILP satisfies the semi-definite constraint:

$$\sum_{e \in E} x_e^* L_e - \gamma (I_n - e_0 \otimes e_0) \succeq 0.$$

If not, one can construct a cut associated with the negative eigenvalue of $\sum_{e \in E} x_e^* L_e - \gamma(I_n - e_0 \otimes e_0)$ by first determining the corresponding eigenvector v_{N+1} and constructing a semi-definite $Q_{N+1} = v_{N+1} \otimes v_{N+1}$. We can augment the MILP with the following scalar linear constraint which cuts off this undesirable solution:

$$\sum_{e \in E} x_e Q_{N+1} \cdot L_e - \gamma Q_{N+1} \cdot (I_n - e_0 \otimes e_0) \geq 0,$$

which is clearly not satisfied when $x_e = x_e^*$, but is satisfied by the optimal solution.

3. One may then solve the augmented MILP using dual simplex algorithm.

4. This procedure is iterated until x_e^* satisfies the semi-definite constraint. Hence, x_e^* is an optimal solution.

2.3.3 EA_2: Algorithm to compute maximum algebraic connectivity

Algorithm 1 : Iterative primal-dual algorithm (EA_2)

1: Input: A primal feasible solution
2: Let $P :=$ Given primal feasible solution. Let the Fiedler vector of P be denoted as v_P and its corresponding eigenvalue represented as γ_P
3: primalCost $\leftarrow \gamma_P$
4: dualCost $\leftarrow \infty$
5: DualGap $\leftarrow$ dualCost-primalCost
6: **if** DualGap > 0 **then**
7: Use v_P to obtain another primal solution, P^*, by solving the following dual problem:

$$\begin{array}{ll} \text{dualCost}_{P^*} = & \max \ v_P \cdot (\sum_{e \in E} x_e L_e) v_P \\ \text{subject to} & \sum_{e \in \delta(S)} x_e \geq 1, \quad \forall \ S \subset V, \\ & \sum_{e \in E} x_e \leq q, \\ & x_e \in \{0,1\}^{|E|}. \end{array}$$

8: $P_t \leftarrow P^*$
9: $Cuts \leftarrow \emptyset$
10: **while** $\gamma_P > \gamma_{P_t}$ **do**
11: Augment $Cuts$ with the following constraint:

$$v_{P_t} \cdot (\sum_{e \in E} x_e L_e) v_{P_t} \geq \gamma_P$$

12: Find P^* again by solving the above dual problem with all the additional constraints in $Cuts$.
13: $P_t \leftarrow P^*$
14: **end while**
15:
16: $P \leftarrow P_t$
17: primalCost $\leftarrow \gamma_{P_t}$
18: dualCost $\leftarrow$ min (dualCost,dualCost$_{P_t}$)
19: DualGap $\leftarrow$ dualCost-primalCost
20: **Termination criterion:** *if* DualGap > 0 *return* to line 7, *else* exit with P as the optimal primal solution.
21: **end if**

EA_2 is a cutting plane algorithm based on the iterative primal-dual method as outlined in Algorithm 1. In this approach, we start with a feasible solution to the primal problem and iteratively update this feasible solution with a new solution by solving a related dual problem. The current feasible solution to the primal problem is only updated with a new solution if the algebraic connectivity of the new solution is greater than the algebraic connectivity of the current feasible solution. On the other hand, if it is certain that the new solution found using the dual problem is not optimum, a cutting plane is augmented to the dual and the dual problem is solved again (*refer to lines 11-12 of the algorithm*). The dual problem is resolved with additional cutting planes until it produces a new solution that is at least as good as the current primal feasible solution (*refer to lines 10-14 of the algorithm*). The algorithm eventually terminates when the dual cost equals the algebraic connectivity of the best known primal solution (*refer to line 20 of the algorithm*). A feature of this algorithm is that the solutions from the dual problem can be continually used to improve the primal feasible solution while continuously decreasing the optimal dual cost and hence the upper bound.

In the following discussion, we discuss the formulation of the dual problem related to the primal and some efficient ways to solve the same. We also outline how to generate cutting planes if the solution to the dual problem does not produce an optimal solution.

We form the dual problem by relaxing the semi-definite constraint,

$$\sum_{e \in E} x_e L_e \succeq \gamma(I_n - e_0 \otimes e_0),$$

and penalizing the objective with a dual variable $Q \in \mathbb{R}^{|V| \times |V|}$ if the constraint is violated. Let $\mathcal{T}$ be the set of networks on (V, E, w_e) which are connected and have at most q edges from E. Then, one may express the dual function $\Pi(Q)$, with its domain being $Q \succeq 0$ and $Q \cdot (I_n - e_0 \otimes e_0) = 1$. One may compute $\Pi(Q)$ for every Q in its domain as:

$$\Pi(Q) = \max_{x \in \mathcal{T}} [\sum_{e \in E} x_e (Q \cdot L_e)].$$

The computation of $\Pi(Q)$ may be carried out using the greedy algorithms for spanning trees (which are the simplest of the connected networks) given in [39], [40] mimicking Prim's or Kruskal algorithm. The property of connectivity is taken into account by the algorithm and hence, is simple and yet efficient. Since $\Pi(Q)$ is a dual function, it is automatically an upper bound for the maximum algebraic connectivity for every Q in its domain. In our approach, at any iteration of the algorithm, the Q we pick to solve the dual problem corresponds to the best known feasible solution, P, available to the primal problem, *i.e.*, Q is chosen to be equal to $v_P \otimes v_P$ where v_P is the Fiedler vector corresponding to P. Note that such a choice of Q always satisfies the constraints $Q \succeq 0$, $Q \cdot (I_n - e_0 \otimes e_0) = 1$ and is therefore always feasible to the dual. If a solution (say, an optimal tree denoted by P^*) that solves the dual problem has an algebraic connectivity greater than the algebraic connectivity of the primal solution P, then the primal solution is replaced with the optimal tree (*i.e.*, $P := P^*$) and a new iteration is started again. If algebraic connectivity of P^* is less than that of P, then the dual problem is augmented with the following cutting plane and solved again. This procedure is repeated until either the dual problem finds a tree with a greater algebraic connectivity or the dual cost equals the primal cost in which case the algorithm terminates. The cutting plane that is added is:

$$v_{P^*} \cdot (\sum_{e \in E} x_e L_e) v_{P^*} \geq \lambda_2(L(P)),$$

where v_{P^*} denotes the Fiedler vector corresponding to the tree P^*. Observe that the above inequality is violated if x is chosen to be P^* since $v_{P^*} \cdot L(P^*) v_{P^*} < \lambda_2(L(P))$. However, from Rayleigh's inequality, the optimal solution to the primal problem always satisfies the above inequality.

Remark 1 *The outer iteration of the algorithm (lines* $6-21$*) terminates when the dual gap becomes zero. If at the end of an outer iteration, the dual gap is not zero, several dual problems are solved until a tree with better algebraic connectivity is found. In the worst case, the number of dual problems that need to be solved in an outer iteration will be at most equal to the number of feasible structures available. Since during every outer iteration, the increment in the algebraic connectivity is positive and the number of dual problems that need to be solved is bounded, the algorithm will eventually terminate with an optimal solution in finite steps.*

2.3.4 Performance of algorithms EA_1 and EA_2

The MISDP formulation $\mathcal{F}_1$ was implemented using Matlab's toolboxes, SeDuMi and YALMIP which are state-of-the-art semi-definite solvers widely used among the researchers in the area of semi-definite programing. The proposed exact algorithms were implemented in `C++` programing language and the resulting MILP's were solved using CPLEX 12.2 with the default solver options. All computational results in this paper were implemented on a Dell Precision T5500 workstation (Intel Xeon E5630 processor @ 2.53GHz, 12GB RAM).

Construction of random instances: Random weighted adjacency matrix, A, for each instance was generated using $A = (M \circ R) + (M \circ R)^T$ where $\circ$ denotes the Hadamard product of magic square (M) and a randomly generated square matrix (R) with zero diagonal entries. The entries of R are the pseudorandom values drawn from the standard uniform distribution on the open interval (0,1) [41]. The term A_{ij} corresponds to the edge weight which may be chosen to connect nodes i and j. Every random cost matrix was chosen such that the maximum spanning tree's algebraic connectivity was greater than the algebraic connectivity of all the star graphs (spanning tree with $|V|$ nodes such that the internal node has a degree equal to $|V| - 1$). This ensured that the optimal solutions were non-trivial connected graphs. Adjacency matrices corresponding to the ten weighted complete graphs of eight nodes are shown in Appendix A.

Corresponding to the weighted adjacency matrices in Appendix A, the optimal spanning trees with maximum algebraic connectivity are shown in Figure 2.7.

In Table 2.3, for eight node networks, we compare the performance of the proposed algorithms implemented in CPLEX with the performance of directly solving the MISDP formulation $\mathcal{F}_1$ in MATLAB's SDP solver. On an average, the two proposed algorithms performed better than the SDP solver in Matlab. Moreover, the EA_2 based on iterative primal dual method performed 1.2 times faster than the EA_1 based on the polyhedral relaxation of the semi-definite constraint. Also, we observed that the MISDP solver in MATLAB ceased to reduce the gap between the upper and lower bounds it maintained during its branch-and-bound routine for networks with nine nodes and hence was practically impossible to solve. The proposed algorithms solved the nine node problem to optimality, but the computation time was in the order of many hours (8 to 9 hours). The optimal solutions for the problem with nine nodes are shown in Table 2.6.

In EA_1, twenty Fiedler vectors of random spanning trees were used to relax the semi-definite constraint. The sequence of upper bounds obtained by this algorithm for instance 3 can be seen in Figure 2.5. For this instance, the algorithm terminates with an optimal solution after choosing approximately 150 feasible spanning trees

Table 2.3: Comparison of CPU time to solve MISDP formulation using Matlab's SDP solver (T_1) with EA_1 (T_2) and EA_2 (T_3) solved using CPLEX solver for networks with 8 nodes.

Instance No.	Optimal solution	T_1 (seconds)	T_2 (seconds)	T_3 (seconds)
1	22.8042	1187.07	428.45	610.31
2	24.3207	2771.24	1323.58	1003.56
3	26.4111	1173.02	630.39	655.32
4	28.6912	559.15	631.08	495.89
5	22.5051	715.61	515.51	608.78
6	25.2167	947.16	1515.15	801.10
7	22.8752	1139.56	1371.69	860.07
8	28.4397	753.48	564.80	274.93
9	26.7965	1127.46	824.64	1287.26
10	27.4913	862.81	383.88	213.48
Avg.		1123.35	818.40	680.62

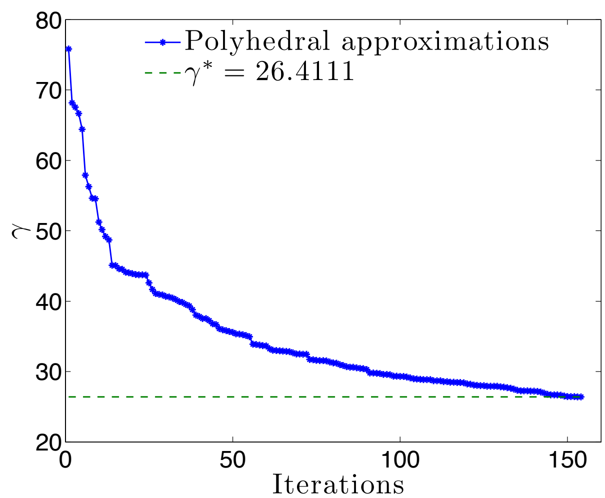


Figure 2.5: EA_1 based on polyhedral approximation of the feasible set: Plot of the upper bound on the algebraic connectivity versus iterations for instance 3 given in Table 2.3. Note that the construction of successively tighter polyhedral approximations of the feasible semi-definite set reduces the upper bound and finally terminates at the optimal solution with maximum algebraic connectivity (γ^*).

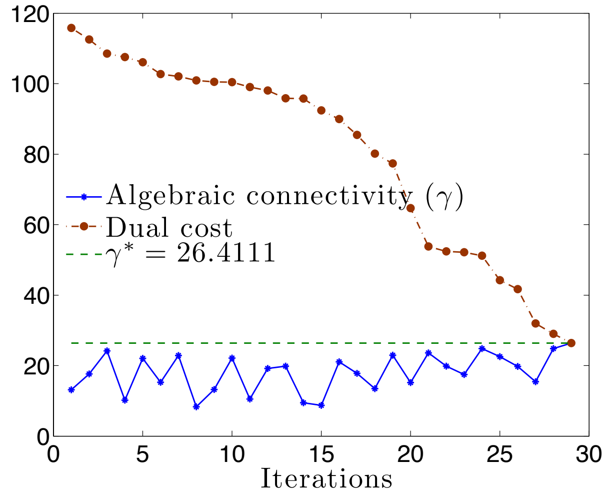


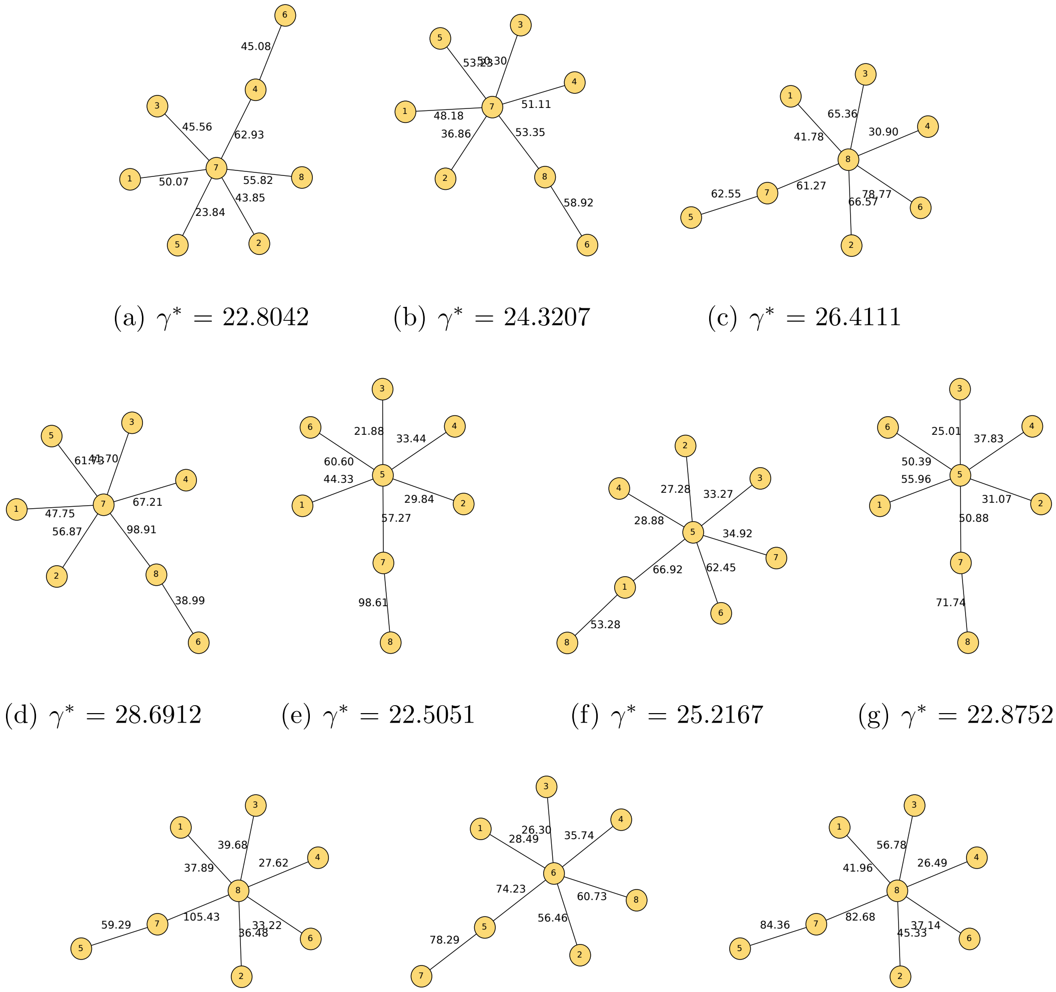
Figure 2.6: EA_2 based on iterative primal-dual method: Plot of algebraic connectivity of primal feasible solutions and dual cost versus iterations for instance 3 given in Table 2.3. Note that this algorithm terminates when the dual cost equals the maximum algebraic connectivity (γ^*).

out of the possible 262144 feasible solutions which we think is quite reasonable.

The exit criterion used for EA_2 can be clearly observed in Figure 2.6. The dual cost which is also an upper bound on the optimal solution continuously gets better with the augmentation of cutting planes and finally exits when the dual gap goes to zero. In this algorithm, the augmented dual problems were solved using the dual-simplex method.

2.3.5 EA_3: Algorithm to compute maximum algebraic connectivity

The earlier sections dealt with two algorithms which synthesized optimal networks with eight nodes in a reasonable amount of time and was a huge improvement over the existing methods to handle MISDPs. However, the computation time for solving nine node problems to optimality was large. Therefore, we propose a different approach for finding an optimal solution in this section by casting the algebraic connectivity problem as the following decision problem: Is there a connected network x with at



(a) $\gamma^* = 22.8042$ (b) $\gamma^* = 24.3207$ (c) $\gamma^* = 26.4111$

(d) $\gamma^* = 28.6912$ (e) $\gamma^* = 22.5051$ (f) $\gamma^* = 25.2167$ (g) $\gamma^* = 22.8752$

(h) $\gamma^* = 28.4397$ (i) $\gamma^* = 26.7965$ (j) $\gamma^* = 27.4913$

Figure 2.7: Optimal networks of eight nodes with maximum algebraic connectivity for the random instances shown in Table 2.3 and the corresponding adjacency matrices in Appendix A.

most q edges from E such that the algebraic connectivity of the network is at least equal to a pre-specified value ($\hat{\gamma}$)?

One of the advantages of posing this question is that the resulting problem turns out to be a Binary Semi-Definite Problem (BSDP) and correspondingly, the tools associated with construction of cutting planes are more abundant when compared to

MILPs. Also, with further relaxation of the semi-definite constraint, it can be solved using CPLEX, a high performance solver for ILPs.

The above decision problem can be mathematically posed as a BSDP by marking any vertex (say $r \in V$) in this graph as a root vertex and then choosing to find a feasible tree that minimizes the degree of this root vertex [2]. In this formulation, the only decision variables would be the binary variables denoted by x_e. Therefore, the resulting BSDP is the following:

$$\begin{aligned} \min \quad & \textstyle\sum_{e \in \delta(r)} x_e, \\ \text{s.t.} \quad & \textstyle\sum_{e \in E} x_e L_e \succeq \hat{\gamma}(I_n - e_0 \otimes e_0), \\ & \textstyle\sum_{e \in E} x_e \leq q, \\ & \textstyle\sum_{e \in \delta(S)} x_e \geq 1, \quad \forall\, S \subset V, \\ & x_e \in \{0,1\}^{|E|}. \end{aligned} \tag{2.14}$$

where, $\delta(r)$ denotes a cutset defined as $\delta(r) = \{e = (r, j) : \ j \in V \setminus r\}$.

If we can solve this BSDP efficiently, then we can use a bisection algorithm to find an optimal solution that will maximize the algebraic connectivity. In order to solve the BSDP using CPLEX, we do the following: we first consider the relaxation of the semi-definite constraint by taking a finite subset of the infinite number of linear constraints from the semi-infinite program, but however add cutset constraints to ensure that the desired network is always connected. These cutset constraints defined by the inequalities

$$\sum_{e \in \delta(S)} x_e \geq 1, \quad \forall S \subset V,$$

require that there is at least one edge chosen from the cutset of any subset S (set of edges from S to its complement $\bar{S}$ in V). If the solution to the relaxed BSDP does

[2]There are several ways to formulate the decision problem as a BSDP. We chose to minimize the degree of a node as it seems to produce reasonably good feasible solutions in every iteration.

not satisfy the semi-definite constraint, we add an eigenvalue cut that ensures that this solution will not be chosen again and then solve the augmented but a relaxed BSDP again. This cutting plane procedure is continued until a feasible solution is found. The idea of this procedure is to construct successively tighter polyhedral approximations of the feasible set corresponding to the desired level of algebraic connectivity which is very similar to the procedure discussed in EA_1. Clearly, the algebraic connectivity of the feasible solution we have is a lower bound for the original MISDP. Hence, we increment the value of $\hat{\gamma}$ to the best known lower bound plus an epsilon value and continue to solve the BSDP. This procedure of bisection is repeated until the BSDP gets infeasible, which implies that we have an optimal solution to the MISDP. The pseudo code of this procedure is outlined in Algorithm 2. Finding an eigenvalue cut that removes the infeasible solution at each iteration is clearly shown in this algorithm.

2.3.6 Performance of EA_3

All the computations in this section were performed with the same computer specifics as mentioned in section 2.3.4. In EA_3 (Algorithm 2), for a given random complete graph, we chose maximum spanning tree as an initial feasible solution since it was computationally inexpensive to evaluate using the standard greedy algorithms. For the bisection step, we assumed $\epsilon = 0.01$.

In Table 2.4, we compare the computational performance of solving a sequence of BSDPs (in formulation (3.7)) with bisection technique directly using the MATLAB's SDP solver with the proposed EA_3 based on cutting plane method implemented in CPLEX. Clearly, the computation time for the EA_3 is much faster (46.45 times) than solving BSDPs directly in MATLAB. For the same set of random instances of eight nodes, it is also worthy to note that, EA_3 performs computationally better

Algorithm 2 : EA_3 **(BSDP approach)**

Let $\mathfrak{F}$ denote a set of cuts which must be satisfied by any feasible solution

1: Input: Graph $G = (V, E, w_e)$, $e \in E$, a root vertex, r, and a finite number of Fiedler vectors, $v_i, i = 1 \dots M$

2: Choose a maximum spanning tree as an initial feasible solution, x^*

3: $\hat{\gamma} \leftarrow \lambda_2(L(x^*))$

4: **loop**

5: $\quad \mathfrak{F} \leftarrow \emptyset$

6: $\quad$ Solve:

$$\begin{array}{ll} \min & \sum_{e \in \delta(r)} x_e, \\ \text{s.t.} & \sum_{e \in E} x_e(v_i \cdot L_e v_i) \geq \hat{\gamma} \quad \forall i = 1, .., M, \\ & \sum_{e \in E} x_e \leq q, \\ & \sum_{e \in \delta(S)} x_e \geq 1, \quad \forall\ S \subset V, \\ & x_e \in \{0, 1\}^{|E|}, \\ & x_e \text{ satisfies the constraints in } \mathfrak{F}. \end{array} \tag{2.15}$$

7: $\quad$ **if** the above ILP is infeasible **then**

8: $\quad\quad$ **break loop** $\{x^*$ is the optimal solution with maximum algebraic connectivity$\}$

9: $\quad$ **else**

10: $\quad\quad$ Let x^* be an optimal solution to the above ILP. Let γ^* and v^* be the algebraic connectivity and the Fiedler vector corresponding to x^* respectively.

11: $\quad\quad$ **if** $\sum_{e \in E} x_e^* L_e \not\succeq \gamma^*(I_n - e_0 \otimes e_0)$ **then**

12: $\quad\quad\quad$ Augment $\mathfrak{F}$ with a constraint $\sum_{e \in E} x_e(v^* \cdot L_e v^*) \geq \gamma^*$.

13: $\quad\quad\quad$ Go to step 6.

14: $\quad\quad$ **end if**

15: $\quad$ **end if**

16: $\quad \hat{\gamma} \leftarrow \hat{\gamma} + \epsilon$ $\{$let ϵ be a small number$\}$

17: **end loop**

than solving MILPs using basic EA_1 and EA_2 in CPLEX as indicated in Table 2.3.

For the problems with nine nodes, EA_3 significantly reduced the average computational time from eight hours to around five to six hours.

Table 2.4: Comparison of CPU time to directly solve the BSDPs in bisection procedure using Matlab's SDP solver (T_1) with the proposed EA_3 using CPLEX solver (T_2) for networks with eight nodes.

Instance No.	λ_2^*	T_1 (seconds)	T_2 (seconds)
1	22.8042	15729.51	254.45
2	24.3207	3652.41	314.62
3	26.4111	3075.72	378.42
4	28.6912	23794.01	420.96
5	22.5051	10032.71	263.06
6	25.2167	20340.92	382.28
7	22.8752	16717.06	484.46
8	28.4397	16837.90	512.47
9	26.7965	44008.42	306.56
10	27.4913	7366.67	204.57
Avg.		15955.68	351.72

2.3.7 Performance of EA_1 with an improved relaxation of the semi-definite constraint

In section 2.3.4, the performance of EA_1 was based on the initial relaxation of the semi-definite constraint using the Fiedler vectors of random feasible solutions. With such a relaxation, the initial upper bound can be weak and hence can incur larger time to compute the optimal algebraic connectivity. However, in section 2.2.1, we discussed a method to provide tight upper bounds by relaxing the semi-definite constraint using the Fiedler vectors of feasible solutions with higher values of algebraic connectivity. Therefore, in this section, we discuss the performance of EA_1 with an improved initial relaxation of the semi-definite constraint as discussed in section 2.2.1.

By choosing thousand Fiedler vectors of good feasible solutions to initially relax the semi-definite constraint, the computation time to obtain optimal solutions are shown in Tables 2.5 and 2.6. T_1 corresponds to the time required to enumerate fifteen thousand spanning trees from the maximum spanning tree and T_2 corresponds to EA_1's time to compute optimal solutions with an improved initial relaxation.

Based on the results in Table 2.5 for the eight nodes problem, the average total computation time of EA_1 with an improved relaxation is *eight* times faster than the computation time of EA_1 without an improved relaxation. Also, the average total computation time in Table 2.5 is *two* times faster than the BSDP approach in EA_3.

Based on the results in Table 2.6 for the nine nodes problem, the average total computation time is around 2.9 hours with the best case instance being 26 minutes. This is a great improvement compared to the eight hours of computation time for EA_1 without an improved relaxation and compared to five hours of computation time for EA_3. The optimal networks with maximum algebraic connectivity for networks with nine nodes are shown in Figure 2.8.

Table 2.5: Performance of EA_1 with an improved relaxation of the positive semi-definite constraint for networks with eight nodes.

Instance No.	λ_2^*	T_1 (seconds)	T_2 (seconds)	$T_1 + T_2$ (seconds)
1	22.8042	70	12	82
2	24.3207	70	142	212
3	26.4111	70	9	79
4	28.6912	70	6	76
5	22.5051	70	7	77
6	25.2167	70	86	156
7	22.8752	70	20	90
8	28.4397	70	10	80
9	26.7965	70	39	109
10	27.4913	70	5	75
Avg.				103.6

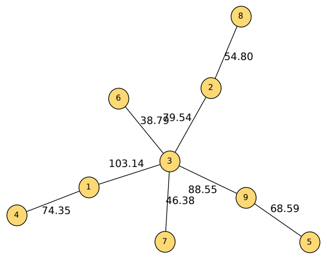
Table 2.6: Performance of EA_1 with an improved relaxation of the positive semi-definite constraint for networks with nine nodes.

Instance No.	λ_2^*	T_1 (seconds)	T_2 (seconds)	$T_1 + T_2$ (seconds)
1	28.2168	170	4295	4465
2	26.3675	170	8093	8263
3	29.8184	170	5377	5547
4	25.8427	170	32788	32958
5	24.2756	170	8880	9050
6	30.0202	170	3981	4151
7	25.6410	170	20458	20628
8	26.9705	170	13796	13966
9	33.5068	170	2908	3078
10	31.7445	170	1417	1587
Avg.				10369.3

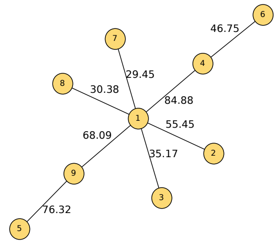
2.4 Neighborhood search heuristics

A general approach to developing heuristics for NP-hard problems primarily involves the following two phases: a) Design of algorithms, also known as construction heuristics, that can provide an initial feasible solution for the problem, and b) Design of a systematic procedure, also known as improvement heuristics, to iteratively modify this initial feasible solution to improve its quality. Since the feasible solutions discussed in this dissertation mainly concern the construction of spanning trees, development of a construction heuristic for the proposed problem is quite trivial. However, it is non-trivial to improve a feasible solution to obtain another feasible solution with better algebraic connectivity.

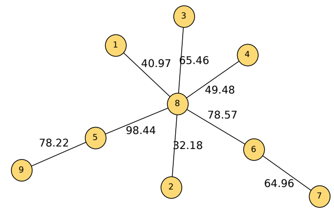
In section 2.3.5, we discussed an exact algorithm based on the BSDP approach wherein every iteration of the bisection, we are guaranteed to obtain a feasible solution with algebraic connectivity greater than or equal to a pre-specified value. How-



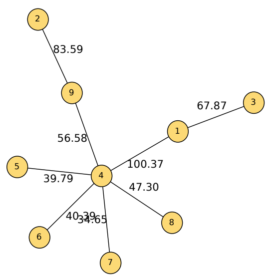
(a) $\gamma^* = 28.2168$



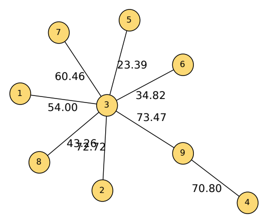
(b) $\gamma^* = 26.3675$



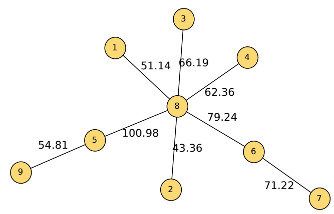
(c) $\gamma^* = 29.8184$



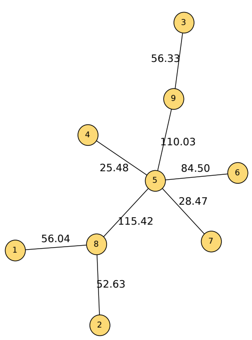
(d) $\gamma^* = 25.8427$



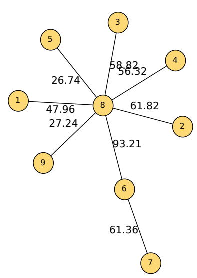
(e) $\gamma^* = 24.2756$



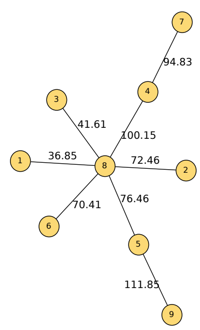
(f) $\gamma^* = 30.0202$



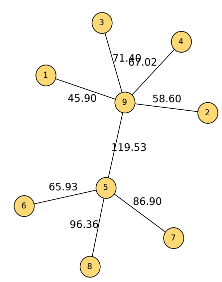
(g) $\gamma^* = 25.6410$



(h) $\gamma^* = 26.9705$



(i) $\gamma^* = 33.5068$



(j) $\gamma^* = 31.7445$

Figure 2.8: Optimal networks of nine nodes with maximum algebraic connectivity for the random instances shown in Table 2.6 and the corresponding adjacency matrices in Appendix A.

ever, the main drawback of this approach was the solving of MILPs of increasing complexity in every iteration without any guarantee on a finite computation time.

However, there are several improvement heuristics available in the literature for sequencing problem and traveling salesman type problems. Some of them include neighborhood search methods, tabu search [42] and even genetic algorithms [43]. In this section, we focus on developing quick improvement heuristics for the problem of maximizing algebraic connectivity based on neighborhood search methods. The remainder of this section is organized as follows: we initially develop k-opt heuristic and later an improved k-opt wherein the size of the search space in the neighborhood of a feasible solution is reduced significantly. We conclude the section with the computational performance of the proposed heuristics for networks up to sixty nodes.

2.4.1 k-opt heuristic

We consider a neighborhood search heuristic called 2-opt exchange heuristic, a special case of a more general k-opt heuristic which has been successfully used for solving traveling salesman problems [44]. We extend the idea of this heuristic to solve the problem of maximizing algebraic connectivity. This heuristic can also be easily extended to the problem of maximizing algebraic connectivity with resource constraints which will be discussed in the later sections.

Any new feasible solution $\mathcal{T}_2$ for this problem is said to be in the k-exchange neighborhood of a feasible solution $\mathcal{T}_1$ if $\mathcal{T}_2$ is obtained by replacing k edges in $\mathcal{T}_1$. In case of 2-opt, we start with a feasible solution, which is a spanning tree satisfying the resource constraints, and iteratively perform 2-opt exchanges for every pair of edges in the initial spanning tree until no such exchanges can be made while improving the solution. A 2-opt exchange on one such pair of edges of a random spanning tree is shown in Figure 2.10. An important issue to be addressed is to make sure that the solutions resulting after 2-opt exchanges are also feasible. Ensuring feasibility in the case of spanning trees is relatively easy as after removing 2 edges, we are guaranteed to have 3 connected components (C_1, C_2, C_3); therefore, by suitably adding any 2 edges connecting all the 3 components, one is guaranteed to obtain a spanning tree.

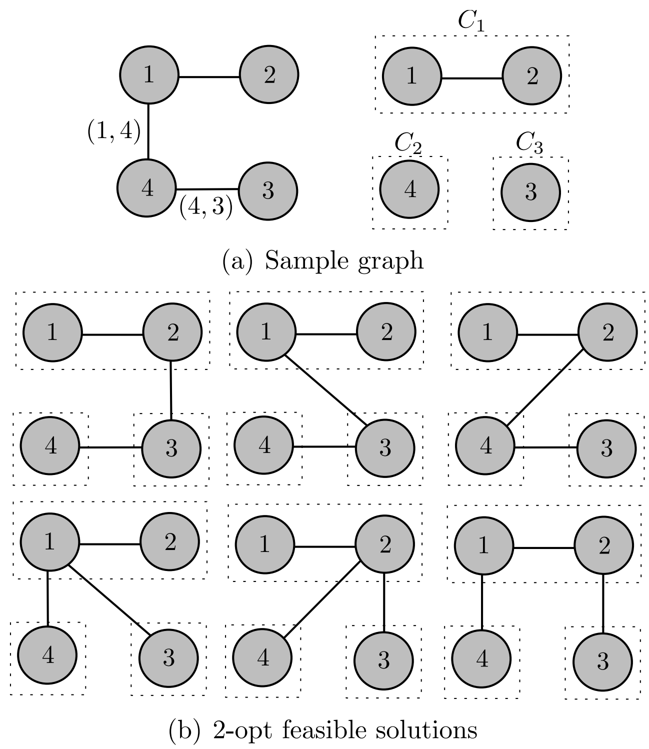


Figure 2.9: This figure illustrates the 2-opt heuristic on an initial feasible solution, $\mathcal{T}_0$. After removing a selected pair of edges $\{(1,4)(4,3)\}$ from $\mathcal{T}_0$, the three connected components are shown in (a). Part (b) shows the 2-opt exchange on the connected components to obtain new feasible solutions.

The new spanning tree (say, $\mathcal{T}_2$) is considered for replacing the initial spanning tree ($\mathcal{T}_1$) if it has a better algebraic connectivity than $\mathcal{T}_1$. A pseudo-code of 2-opt heuristic is given in Algorithm 3. Note that this heuristic performs a 2-opt exchange on a given initial spanning tree to obtain a new tree with better algebraic connectivity satisfying the resource constraint. This procedure is repeated on the current feasible solution iteratively until no improvement is possible.

Algorithm 3 : 2-opt exchange heuristic

$\mathcal{T}_0 \leftarrow$ Initial feasible solution
$\lambda_0 \leftarrow \lambda_2(L(\mathcal{T}_0)))$
for each pair of edges $\{(u_1, v_1), (u_2, v_2)\} \in \mathcal{T}_0$ **do**
  Let $\mathcal{T}_{opt}$ be the best spanning tree in the 2-exchange neighborhood of $\mathcal{T}_0$ obtained by replacing edges $\{(u_1, v_1), (u_2, v_2)\}$ in $\mathcal{T}_{opt}$ with a different pair of edges.
  if $\lambda_2(L(\mathcal{T}_{opt})) > \lambda_0$ and $\mathcal{T}_{opt}$ satisfies the resource constraint **then**
    $\mathcal{T}_0 \leftarrow \mathcal{T}_{opt}$
    $\lambda_0 \leftarrow \lambda_2(L(\mathcal{T}_{opt}))$
  end if
end for
$\mathcal{T}_0$ is the best spanning tree in the solution space with respect to the initial feasible solution

---

2.4.2 Improved k-opt heuristic

As discussed in section 2.4.1, there are two main steps in the k-opt exchange heuristic: Choosing a collection of k edges to remove from the current solution and then reconnecting the resulting, disjoint components with a new collection of k edges. Clearly, there are several combinations of k edges that can be removed from (*or* added to) a given solution, especially when the number of nodes in the graph is large. For example, while performing 3-opt heuristic on a network of 40 nodes with four connected components after deleting any three edges, there would be at least 16,000 combinations of edges which can be connected to form a spanning tree. Therefore, choosing an efficient procedure for the deletion and addition of edges is critical for developing a relatively fast algorithm. In the following subsections, we provide procedures for implementing these steps.

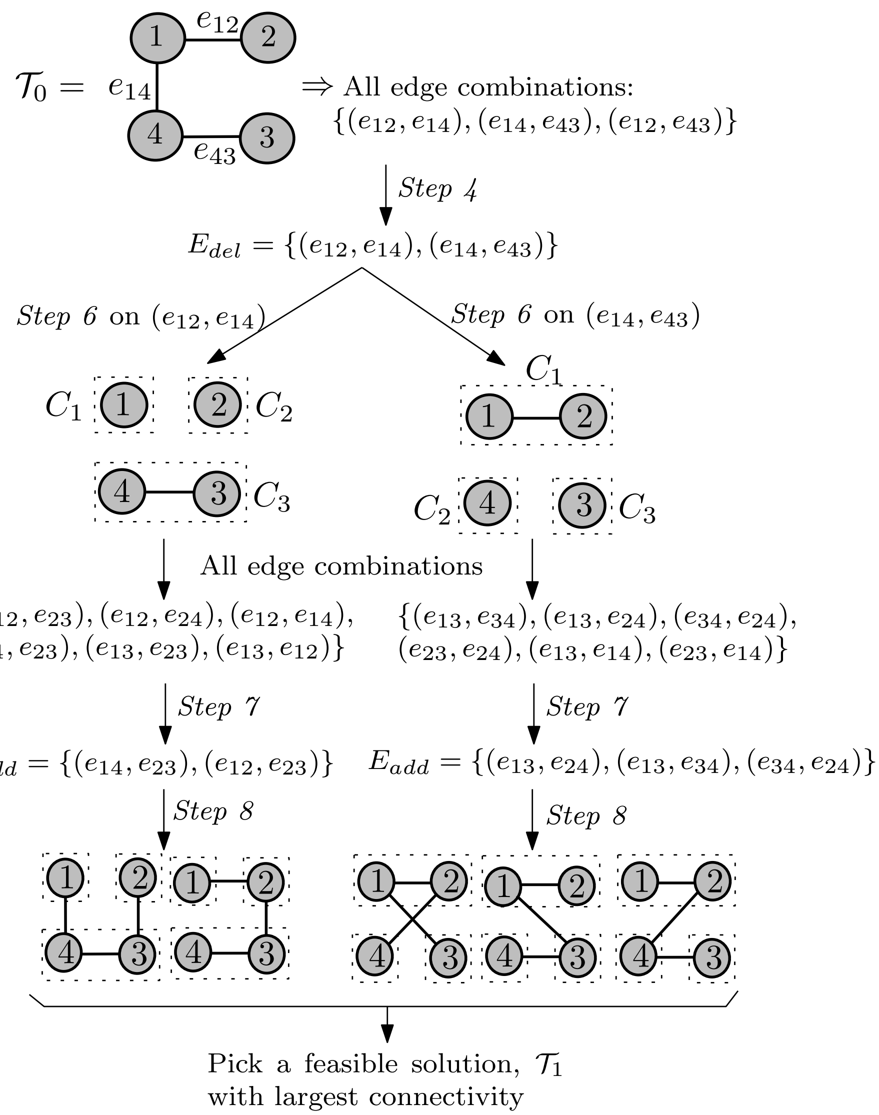


Figure 2.10: An example illustrating an improved 2-opt exchange heuristic for a network of 4 nodes.

Selecting a collection of k-edge combinations to delete: The basic idea here is to list all the possible combinations of k edges that can be deleted from the current feasible solution, assign a value for each combination, rank the combinations based on these values, and then choose a subset of these combinations for further processing. We assign a value to a combination of edges by first asking the following basic

Algorithm 4 : k-opt exchange

```
T_0 ← Initial feasible solution
λ_0 ← λ_2(L(T_0)))
E_del ← Subset of k-edge combinations considered for possible deletion as obtained
  by the edge ranking procedure
for each edge combination in E_del do
  Delete the k edges present in the edge combination to obtain connected com-
    ponents C_1, C_2, C_3, ..., C_{k+1}
  E_add ← Subset of k-edge combinations considered for possible addition as
    obtained by the edge ranking procedure
  Let T_1 be the spanning tree which is feasible and has the largest connectivity
    obtained from adding the edges in an edge combination from E_add
  if λ_2(L(T_opt)) > λ_0 and T_opt satisfies the resource constraint then
    T_0 ← T_1
    λ_0 ← λ_2(L(T_1))
  end if
end for
Output T_0 as the (new) current solution
```

question: Which are the k edges that needs to be deleted from the current solution $\mathcal{T}_0$ so that $\mathcal{T}_0$ (possibly) incurs the smallest reduction in the algebraic connectivity? To answer this question, let $\mathcal{T}_0 \setminus \{e\}$ denote the graph obtained by deleting an edge e from the graph $\mathcal{T}_0$ and by abuse of notation, let the Laplacian of the graph, $\mathcal{T}_0 \setminus \{e\}$ be denoted by $L(\mathcal{T}_0 - e)$. By variational characterization, we have the following inequality:

$$\lambda_2(L(\mathcal{T}_0 - e)) \leq v' L(\mathcal{T}_0 - e) v \quad \forall v \in \mathcal{V} \tag{2.16a}$$

$$= v' L(\mathcal{T}_0) v - v' L_e v \quad \forall v \in \mathcal{V} \tag{2.16b}$$

$$= v' L(\mathcal{T}_0) v - w_e (v_i - v_j)^2 \quad \forall v \in \mathcal{V} \tag{2.16c}$$

where, $\mathcal{V} := \{v : \sum_i v_i = 0, \|v\| = 1\}$, w_e is the weight of edge $e = (i, j)$ and

v_i represents the i^{th} component of the vector v. One may observe from the above inequality that by choosing an edge with a minimum value of $w_e(v_i - v_j)^2$, the upper bound on the algebraic connectivity of the graph, $\mathcal{T}_0 \setminus \{e\}$, is kept as high as possible. Also, we numerically observed that, $w_e(v_i - v_j)^2$ was kept to a minimum by choosing v as the eigenvector corresponding to the maximum eigenvalue of $L(\mathcal{T}_0)$. Hence, for any combination of k edges denoted by S, we assign a value given by $\sum_{e=(i,j)\in S} w_e(v_i - v_j)^2$. Then, we rank all the combinations based on the increasing values and choose a subset of these combinations that corresponds to the lowest values. In this work, the fraction of combinations that is considered for deletion is specified through a parameter called the edge deletion factor. The edge deletion factor is defined as the ratio of the number of k-edge combinations considered for deletion to the maximum number of possible k-edge combinations (*i.e.*, $\binom{n-1}{k}$). We will discuss more about this factor later in section 2.4.3.

Selecting a collection of k-edge combinations to add: In the case of spanning trees, after removing k edges, we are guaranteed to have a graph $\tilde{\mathcal{T}}_0$ with exactly $k+1$ connected components $\{C_1, C_2, C_3, \ldots, C_{k+1}\}$; therefore, by suitably adding a collection of k edges connecting all the $k+1$ components in $\tilde{\mathcal{T}}_0$, one is guaranteed to obtain a spanning tree, $\mathcal{T}_1$. Also, we add these edges while ensuring that the resulting tree satisfies the diameter constraints. The new feasible solution $\mathcal{T}_1$ is considered for replacing $\mathcal{T}_0$ if it has a larger algebraic connectivity than $\mathcal{T}_0$.

As in the edge-deletion procedure, checking for every addition of k edges may be computationally intensive for large instances. Therefore, we develop another edge ranking procedure for adding edges as follows: Let $\tilde{\mathcal{T}}_0 \cup \{e\}$ denote the graph obtained by adding an edge $e = (i, j)$ to the graph $\tilde{\mathcal{T}}_0$ and let the Laplacian of the graph, $\tilde{\mathcal{T}}_0 \cup \{e\}$, be denoted by $L(\tilde{\mathcal{T}}_0 + e)$. By variational characterization, we have the following inequality:

$$\lambda_2(L(\tilde{\mathcal{T}}_0 + e)) \leq v' L(\tilde{\mathcal{T}}_0 + e) v \quad \forall v \in \mathcal{V} \tag{2.17a}$$

$$= v' L(\tilde{\mathcal{T}}_0) v + v' L_e v \quad \forall v \in \mathcal{V} \tag{2.17b}$$

$$= v' L(\tilde{\mathcal{T}}_0) v + w_e (v_i - v_j)^2 \quad \forall v \in \mathcal{V}. \tag{2.17c}$$

One may observe from the above inequality that by choosing an edge with a maximum value of $w_e(v_i - v_j)^2$, the upper bound on the algebraic connectivity of the graph, $\tilde{\mathcal{T}}_0 \cup \{e\}$ is kept as high as possible. Just like the edge deletion step, let v be the eigenvector corresponding to the maximum eigenvalue of $L(\tilde{\mathcal{T}}_0)$. Hence, for any combination of k edges denoted by S, we assign a value given by $\sum_{e=(i,j) \in S} w_e(v_i - v_j)^2$. Then, we rank all the combinations based on decreasing values and choose a subset of these combinations that corresponds to the highest values. The number of combinations that are considered for addition is another parameter and can be specified based on the problem instances.

A pseudo-code of the k-opt exchange is outlined in Algorithm 4. An illustration of such a procedure on one such pair ($k = 2$) of edges for a spanning tree with 4 nodes is shown in Figure 2.10. This exchange is iteratively applied on the current solution until no improvements can be made.

2.4.3 Performance of k-opt and improved k-opt heuristic

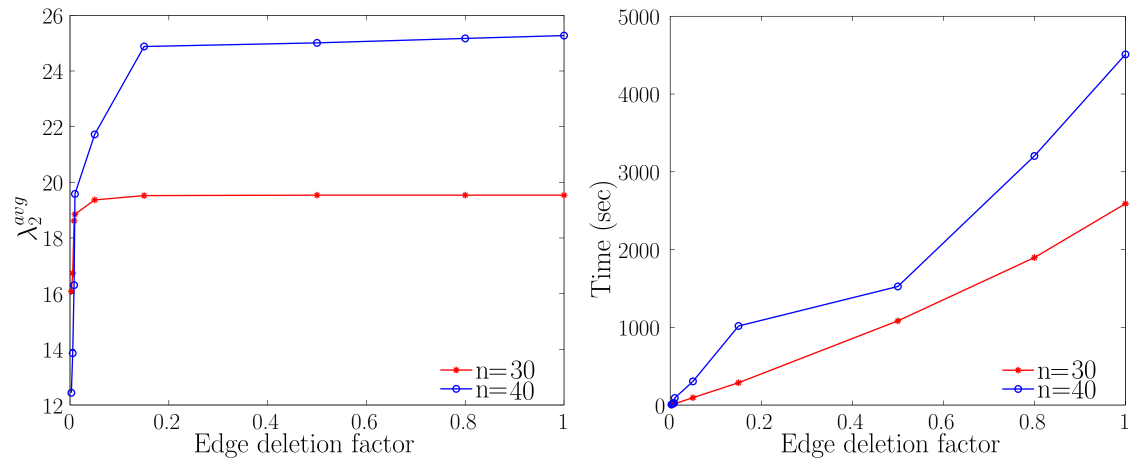
All the computations in this section were performed with the same computer specifics as mentioned in section 2.3.4.

In this section, we performed all the simulations for the case of k equal to two and three, which we shall refer as 2-opt and 3-opt heuristics. The 2-opt (Algorithm

3) and improved 2-opt heuristics (Algorithm 4) were implemented in Matlab since the heuristics terminated in a reasonable amount of time. But, we implemented the improved 3-opt heuristic (Algorithm 4) in `C++` programming language which could handle up to sixty nodes in a reasonable amount of time.

Construction of initial feasible solution: Since the primary idea of the k-opt heuristic is to search in the neighborhood space of an initial feasible solution, it would be important to construct a good initial feasible solution. As we saw in the previous section on exact algorithms (Figure 2.7), the networks with maximum algebraic connectivity tend to be clustered and are low in diameter. With this intuition, the procedure to construct an initial feasible solution is as follows: For a given complete weighted graph with n nodes, sort the n possible star graphs (two diameter graphs) in the decreasing order of the sum of the weights of the edges incident on the internal node (weighted degree) of the graph. After sorting, we observed that, performing k-opt exchange on five of these ranked star graphs provided a great improvement in the algebraic connectivity.

Selecting edge deletion and edge addition factor: For the improved k-opt heuristic, we set the edge deletion factor to be equal to 0.15 in all the simulations. We chose this value based on the simulation results shown in Figure 2.11. This figure shows the average algebraic connectivity of the final solution (and the computation time) obtained using the improved $3-$opt heuristic as a function of the edge deletion factor. We observed that there was not much improvement in the quality of the feasible solutions beyond a value of 0.15 (for the edge deletion factor) even for large instances ($n = 30, 40$). Hence, we chose 0.15 as the edge deletion factor. Also, we set the number of combinations of edges to be added to be at most equal to 5^k. For improved 3-opt, this parameter was set to 125. We chose this value based on the simulation results shown in Figure 2.12. For improved 2-opt, this parameter was set to 25 after performing similar simulations.



(a) (b)

Figure 2.11: Average values of the algebraic connectivity (a) and computation times (b) obtained as a function of the edge deletion factor using the improved 3-opt heuristic over ten instances. In these computations, the maximum number of edge combinations considered for addition between any two components was set to 125.

Performance of k-opt and improved k-opt with respect to optimal solutions: For the problem with 8 nodes, we define the solution quality of the proposed heuristic as

$$\text{Solution quality} = \frac{\lambda_2^* - \lambda_2^{kopt}}{\lambda_2^*} \times 100$$

where λ_2^{kopt} denotes the algebraic connectivity of the solution found by the k-opt heuristic and λ_2^* represents the optimum. The results shown in Table 2.7 are for 10 random instances generated for networks with 8 nodes. Based on the results in Table 2.7, it can be seen that the quality of solutions found by the k-opt ($k = 2, 3$) and

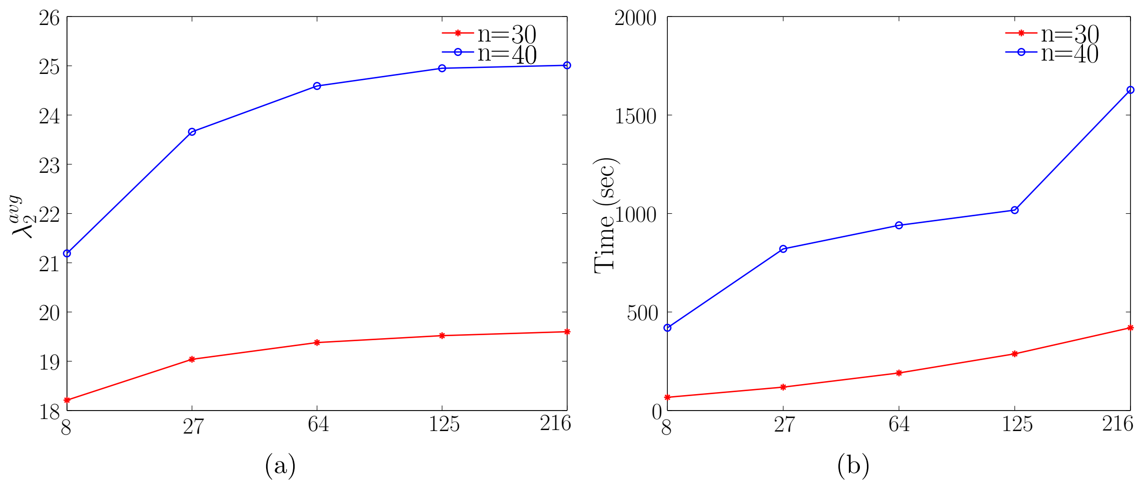
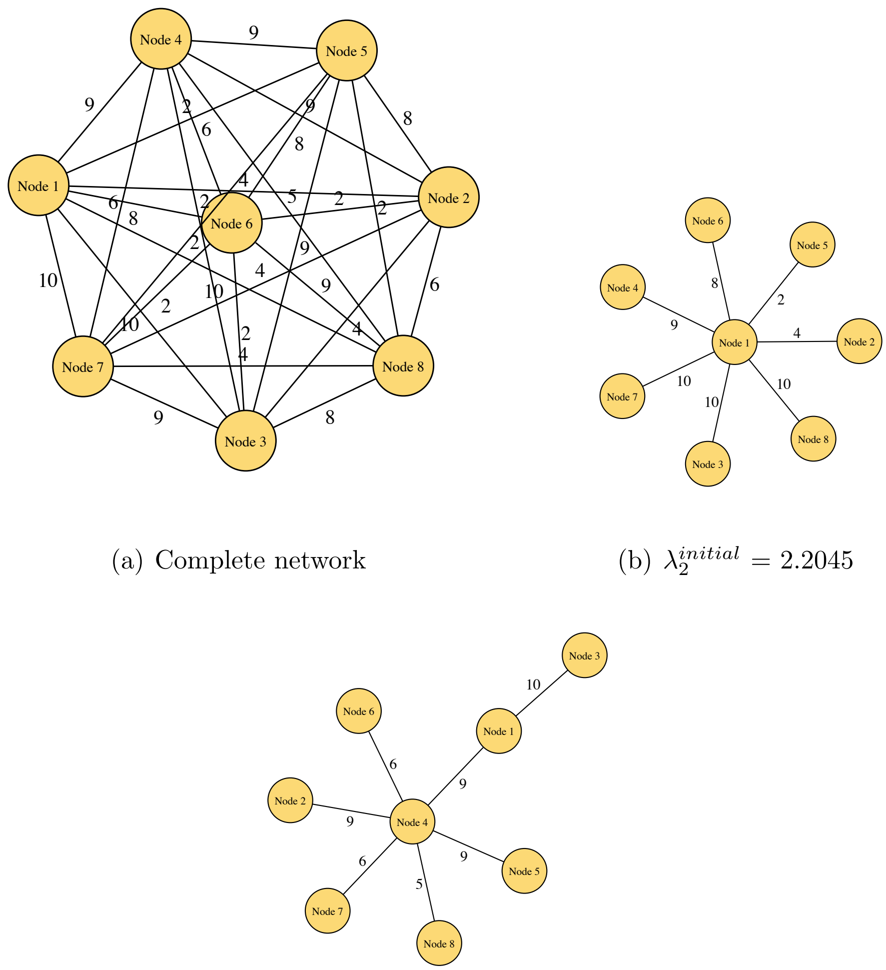


Figure 2.12: The average algebraic connectivity (a) and computation times (b) obtained as a function of the maximum number of edge combinations considered for addition between any two components in the improved 3-opt heuristic over ten instances. In these computations, the edge deletion factor was set to 0.15.

the improved k-opt ($k = 2, 3$) heuristic were very good and gave optimal solutions for all the 10 random instances. Also, on an average, the computation time for the heuristics were less than 1.5 seconds to obtain the best feasible solution for the problem with 8 nodes. An improvement in the computational time for improved k-opt heuristic can be observed for larger instances as discussed in the later parts of this section. Instance 1 of Table 2.7 is pictorially shown in Figure 3.2.

Performance of k-opt and improved k-opt for large instances: For problems with larger instances ($n \geq 10$), in Table 2.8, we analyze the improvement in the computation time of improved 2-opt heuristic with respect to the standard 2-opt heuristic and also compare their solution qualities.

For a given initial feasible solution, the neighborhood search space for the improved 2-opt is a subset of the neighborhood search space for the standard 2-opt. Hence, we define the reduction in the value of the algebraic connectivity of improved



(a) Complete network

(b) $\lambda_2^{initial} = 2.2045$

(c) λ_2^{2opt},$\lambda_2^{3opt} = 3.9712$

Figure 2.13: A network with all possible edges connecting 8 nodes including edge weights are shown in (a). (b) represents the initial feasible solution which is a star graph. (c) represents an optimal network which also happens to be the solution found by the 2-opt and 3-opt heuristics.

2-opt from the standard 2-opt as

$$\text{percent reduction} = \frac{\lambda_2^{2opt} - \lambda_2^{2opt_{imp}}}{\lambda_2^{2opt}} \times 100$$

Table 2.7: Comparison of the quality of the solutions found by the k-opt heuristic (Algorithm 3) for networks with 8 nodes. λ_2^* is the optimal algebraic connectivity.

	Optimal solution		k-opt, Improved k-opt (k=2,3)		
No.	λ_2^*	Time (sec)	λ_2^{kopt}	Solution quality	Time (sec)
1	3.9712	180.57	3.9712	0.00	1.1
2	4.3101	408.10	4.3101	0.00	2.1
3	3.9297	621.85	3.9297	0.00	1.3
4	3.5275	216.79	3.5275	0.00	2.3
5	3.8753	470.63	3.8753	0.00	0.8
6	3.7972	342.14	3.7972	0.00	1.2
7	3.7125	377.47	3.7125	0.00	1.7
8	3.9205	313.12	3.9205	0.00	1.6
9	3.7940	341.84	3.7940	0.00	2.3
10	3.8923	316.86	3.8923	0.00	2.1
Avg.		358.43		0.00	1.5

where λ_2^{2opt} is the algebraic connectivity of a solution obtained from the 2-opt heuristic and $\lambda_2^{2opt_{imp}}$ is the algebraic connectivity of a solution obtained from the improved 2-opt heuristic. From the results in Table 2.8, it can be seen that the improved 2-opt heuristic performed almost as good as the 2-opt heuristic without much reduction in the quality of the solution but with a very remarkable improvement in the computational time to obtain the feasible solution. Therefore, it can be observed that the greedy procedure of deletion and addition of the edges based on the variational characterization of the eigenvalues has reduced the neighborhood search space very effectively.

In Table 2.9, we further study the improvement in the solution quality of the improved 3-opt heuristic with respect to the improved 2-opt heuristic. For this

Table 2.8: Comparison of 2-opt with improved 2-opt heuristic solutions for various problem sizes. Here, the solution quality was averaged over ten random instances for each n.

n	2-opt	Improved 2-opt	
	Time (sec)	Percent reduction	Time (sec)
10	0.88	0.00	0.10
15	8.45	0.00	0.52
20	60.47	0.00	1.65
25	240.57	0.60	3.59
30	1533.98	0.81	12.13
35	3468.75	0.79	37.85
40	5899.62	1.20	57.28
45	8897.69	1.16	116.09
50	10089.31	1.27	139.99
55	12980.78	1.80	350.83
60	16001.02	2.01	505.36

purpose, we define the percent improvement as follows:

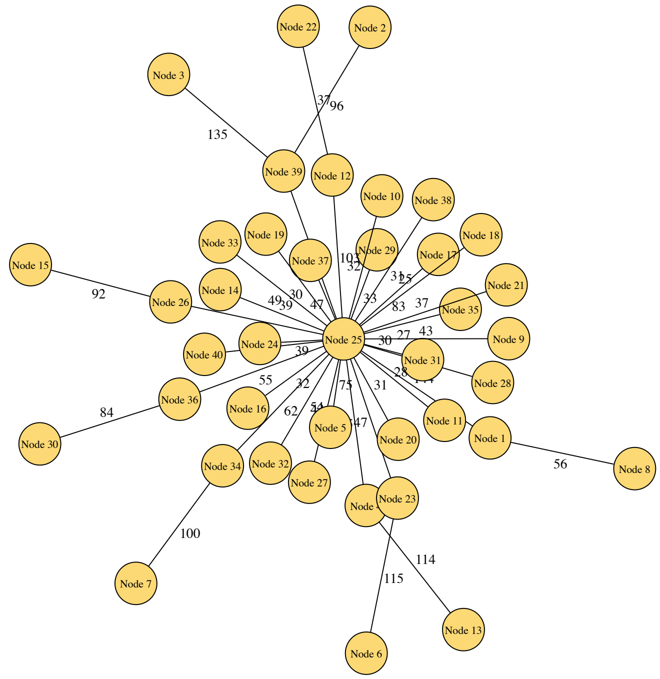
$$\text{percent improvement} = \frac{\lambda_2^{3opt_{imp}} - \lambda_2^{2opt_{imp}}}{\lambda_2^{3opt_{imp}}} \times 100$$

where $\lambda_2^{3opt_{imp}}$ is the algebraic connectivity of a solution obtained from the improved 3-opt heuristic and $\lambda_2^{2opt_{imp}}$ is the algebraic connectivity of a solution obtained from the improved 2-opt heuristic. From the results in Table 2.9, it can be seen that the improved 3-opt heuristic performed consistently better than the improved 2-opt, though the quality of solution was quite comparable in an average sense. It can also be observed that there were instances where the improvement in the solution quality of the improved 3-opt from the improved 2-opt heuristic was up to around 18 percent for large instances.

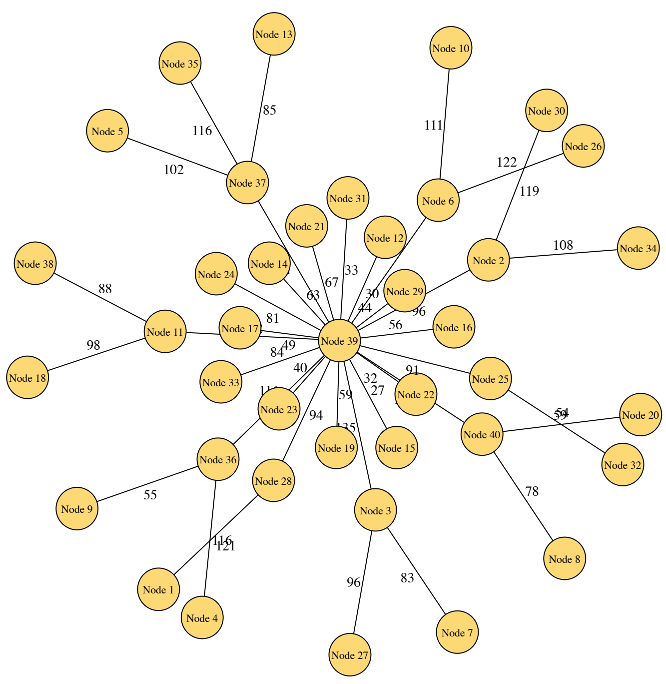
In summary, computational results suggested that the improved 3-opt heuristic performed the best while the improved 2-opt heuristic provided a good trade-off between finding good feasible solutions and the required computation time. Figure 2.14 illustrates the solutions obtained from the improved 2-opt and 3-opt search heuristic for a network with 40 nodes.

Table 2.9: Comparison of improved 3-opt and improved 2-opt heuristic solutions for various problem sizes. The percent improvement values were averaged over ten random instances for each n.

n	Improved 2-opt	Improved 3-opt		
	Time (sec)	Time (sec)	Average percent improvement	Maximum percent improvement
10	0.10	0.29	0.00	0.00
15	0.52	3.06	0.01	0.08
20	1.65	16.32	0.27	2.73
25	3.59	60.38	0.60	4.92
30	12.13	274.93	2.07	7.56
35	37.85	480.15	2.02	12.59
40	57.28	1016.99	5.62	17.89
45	116.09	2309.60	7.10	15.41
50	139.99	4219.17	1.38	5.10
55	350.83	6798.34	6.98	17.56
60	505.36	8974.46	7.97	16.92



(a) $\lambda_2^{2opt_{imp}} = 21.4088$



(b) $\lambda_2^{3opt_{imp}} = 26.0731$

Figure 2.14: Improved 2-opt and 3-opt exchange heuristic solutions for a network with 40 nodes.

3. *ALGORITHMS FOR THE MAXIMIZATION OF ALGEBRAIC CONNECTIVITY UNDER RESOURCE CONSTRAINTS

We discussed earlier the variant of **BP** involving the construction of an adhoc infrastructure network with UAVs that can be central to civilian and military applications. We also alluded briefly to the desirable attributes of the UAV adhoc network such as: a) Lower diameter to minimize latency in communicating data/information across the network, b) A limit/budget on the power consumed by the UAVs due to their limited battery capacities and c) High isoperimetric number so that the bottlenecking in a network can only occur at higher data rates while at the same time be robust to node and link failures.

In addition to the bound on the number of communication links, treating the requirements on diameter and power consumption as the constraints on the resources, a variant of the **BP** that arises in the UAV application may be posed as follows: Given a collection of UAVs which can serve as backbone nodes, how should they be arranged and connected so that

(i) the convex hull of the projections of their locations on the ground spans a minimum area of coverage,

(ii) the resources such as the diameter of the network, total UAV power consumption for maintaining connectivity and the total number of communication links

*Reprinted with permission from 1) Algorithms for Finding Diameter-constrained Graphs with Maximum Algebraic Connectivity by H.Nagarajan, S.Rathinam, S.Darbha and K.R.Rajagopal, 2012. *Dynamics of Information Systems: Mathematical Foundations*, 121–135, Copyright 2012 by Springer.
2. Synthesizing robust communication networks for UAVs with resource constraints by H.Nagarajan, S.Rathinam and S.Darbha, 2012. *Dynamic Systems and Control Conference, 2012, 2012*. 2524–2533, Copyright 2012 by ASME.

employed are within their respective prescribed bounds, and

(iii) algebraic connectivity of the network is maximum among all possible networks satisfying the constraints (i) and (ii).

Since each of these resource constraints makes the problem much harder, we separately formulate the diameter and power consumption constraint in the forthcoming sections. Hence, in the remainder of this section, we shall discuss the respective mathematical formulations and extend the algorithms based on cutting plane techniques as discussed in the in section 2.3 to synthesize optimal networks.

3.1 Maximization of algebraic connectivity with diameter constraint

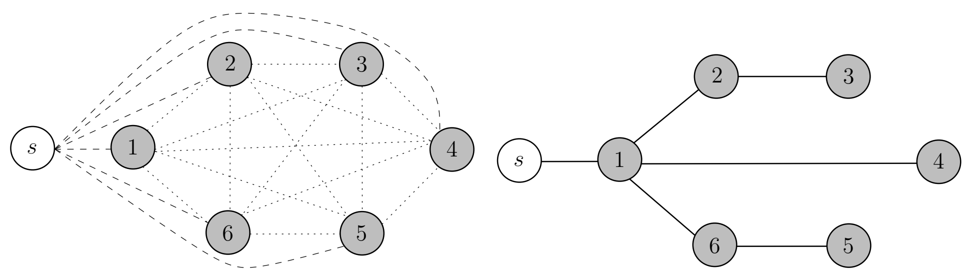
3.1.1 Problem formulation

Based on the notation defined in section 2.1, the problem of choosing at most q edges from E so that the algebraic connectivity of the augmented network is maximized and the diameter of the network is within a given constant (D) can be posed as follows:

$$\begin{aligned} \gamma^* = \quad & \max \lambda_2(L(x)), \\ \text{s.t.} \quad & \textstyle\sum_{e \in E} x_e \leq q, \\ & \delta_{uv}(x) \leq D \quad \forall u, v \in V, \\ & x_e \in \{0,1\}^{|E|}. \end{aligned} \tag{3.1}$$

where $\delta_{uv}(x)$ represents the number of edges on the shortest path joining the two nodes u and v in the network with an incident vector x. In the above formulation, there are two challenges that need to be overcome before one can pose the above problem as a MISDP. First, the objective is a non-linear function of the edges in the network; secondly, the diameter constraint as stated in formulation (3.1) requires one to implicitly compute the number of edges in the shortest path joining any two vertices. We have already discussed in section 2.1 of section 2, how to address the above non-linear problem as a MISDP which is as follows:

$$
\begin{aligned}
\gamma^* = \quad & \max \gamma, \\
\text{s.t.} \quad & \textstyle\sum_{e \in E} x_e L_e \succeq \gamma(I_n - e_0 \otimes e_0), \\
& \textstyle\sum_{e \in E} x_e \leq q, \\
& \delta_{uv}(x) \leq D \quad \forall u, v \in V, \\
& x_e \in \{0,1\}^{|E|}.
\end{aligned} \tag{3.2}
$$



(a) Original graph augmented with a source node

(b) Feasible solution

Figure 3.1: Illustration of an addition of the source node (s) to the original (complete) graph represented by shaded nodes. If one were given that the diameter of the original graph must be at most $D = 4$, then restricting the length of each of the paths from the source node to $(D/2) + 1 = 3$, and allowing only one incident edge on s will suffice as shown in (b)

The next difficulty one needs to address stems from the diameter constraints formulated in (3.1). To simplify the presentation, let us limit our search of an optimal network to the set of all the spanning trees. Also, let the parameter D which limits the diameter of the network be an even number. Then, it is well known [45] that a spanning tree has a diameter no more than an even integer (D) if and only if there exists a central node p such that the path from p to any other node in the graph consists of at most $D/2$ edges. If the central node p is given, then one can use the multicommodity flow formulation [32] to keep track of the number of edges present in any path originating from node p. However, since p is not known a priori, a common way to address this issue is to augment the network with a source node (s) and connect this source node to each of the remaining vertices in the network with an edge (refer to figure 3.1). If one were to find a spanning tree in this augmented network such that there is only one edge incident on the source node and the path from the source node to any other node in the graph consists of at most $\frac{D}{2}+1$ edges, the diameter constraint for the original network will be naturally satisfied.

In order to impose the diameter constraints formulated in (3.1), we add a source node (s) to the graph (V, E) and add an edge joining s to each vertex in V, *i.e.*, $\tilde{V} = V \cup \{s\}$ and $\tilde{E} = E \cup (s, j) \quad \forall j \in V$. We then construct a tree spanning all the nodes in $\tilde{V}$ while restricting the length of the path from s to any other node in $\tilde{V}$. The additional edges emanating from the source node are used only to formulate the diameter constraints, and they do not play any role in determining the algebraic connectivity of the original graph.

Constraints representing a spanning tree are commonly formulated in the literature using the multicommodity flow formulation. In this formulation, a spanning tree is viewed as a network which facilitates the flow of a unit of commodity from the source node to each of the remaining vertices in $\tilde{V}$. A commodity can flow directly between two nodes if there is an edge connecting the two nodes in the network. Similarly, a commodity can flow from the source node to a vertex v if there is a path joining the source node to vertex v in the network. One can guarantee that all the vertices in V are connected to the source node by constructing a network that allows for a distinct unit of commodity to be shipped from the source node to each vertex in V. To formalize this further, let a distinct unit of commodity (also referred to as the k^{th} commodity) corresponding to the k^{th} vertex be shipped from the source node. Let f_{ij}^k be the k^{th} commodity flowing from node i to node j. Then, the constraints which express the flow of the commodities from the source node to the vertices can be formulated as follows:

$$\sum_{j \in \tilde{V} \setminus \{s\}} (f_{ij}^k - f_{ji}^k) = 1 \quad \forall k \in V \text{ and } i = s, \tag{3.3a}$$

$$\sum_{j \in \tilde{V}} (f_{ij}^k - f_{ji}^k) = 0 \quad \forall i, k \in V \text{ and } i \neq k, \tag{3.3b}$$

$$\sum_{j \in \tilde{V}} (f_{ij}^k - f_{ji}^k) = -1 \quad \forall i, k \in V \text{ and } i = k, \tag{3.3c}$$

$$f_{ij}^k + f_{ji}^k \leq x_e \quad \forall \ e := (i,j) \in \tilde{E}, \forall k \in V, \tag{3.3d}$$

$$\sum_{e \in \tilde{E}} x_e = |\tilde{V}| - 1, \tag{3.3e}$$

$$0 \leq \ f_{ij}^k \ \leq 1 \quad \forall i, j \in \tilde{V}, \forall k \in V, \tag{3.3f}$$

$$x_e \in \{0, 1\} \quad \forall e \in \tilde{E}. \tag{3.3g}$$

Constraints (3.3a) through (3.3c) state that each commodity must originate at the source node and terminate at its corresponding vertex. Equation (3.3d) states that the flow of commodities between two vertices is possible only if there is an edge joining the two vertices. Constraint (3.3e) ensures that the number of edges in the chosen network corresponds to that of a spanning tree. An advantage of using this formulation is that one now has access directly to the number of edges on the path joining the source node to any vertex in the graph. That is, $\sum_{(i,j) \in \tilde{E}} f_{ij}^k$ denotes the length of the path from s to k. Therefore, the diameter constraints now can be expressed as:

$$\sum_{(i,j)\in\tilde{E}} f_{ij}^k \le (D/2+1) \quad \forall k \in V, \tag{3.4a}$$

$$\sum_{j\in V} x_{sj} = 1. \tag{3.4b}$$

To summarize, the MISDP for the network synthesis problem with diameter constraints is:

$$\begin{aligned} \gamma^* = \quad & \max \gamma, \\ \text{s.t.} \quad & \textstyle\sum_{e\in E} x_e L_e \succeq \gamma(I_n - e_0 \otimes e_0), \\ & \textstyle\sum_{e\in E} x_e \le q, \\ & \text{Constraints in (3.3) and (3.4)}, \\ & x_e \in \{0,1\}^{|E|}. \end{aligned} \tag{3.5}$$

Note that the formulation in (3.5) is for the case when the desired network is a spanning tree and the bound on the diameter of the spanning tree is an even number. Using the results in [45], similar formulations can also be stated for more general networks with no restrictions on the parity of the bound. In this section, we will concentrate on the formulation presented in (3.5).

3.1.2 Algorithm for determining maximum algebraic connectivity with diameter constraint

In order to pose the problem as a BSDP, let the specified level of connectivity be $\hat{\gamma}$. The decision problem can be mathematically formulated as follows: Is there an incident vector x such that

$$
\begin{aligned}
&\textstyle\sum_{e\in E} x_e L_e \succeq \hat{\gamma}(I_n - e_0 \otimes e_0),\\
&\textstyle\sum_{e\in E} x_e \leq q,\\
&\text{Constraints in (3.3) and (3.4)},\\
&x_e \in \{0,1\}^{|E|}\ ?
\end{aligned} \tag{3.6}
$$

The above problem can be posed as a BSDP by marking any vertex in V as a root vertex r and then choosing to find a feasible tree that minimizes the degree of this root vertex [1]. In this formulation, the only decision variables would be the binary variables denoted by x_e and the flow variables denoted by f_{ij}^k. Therefore, the BSDP we have is the following:

$$
\begin{aligned}
&\min \textstyle\sum_{e\in\delta(r)} x_e,\\
\text{s.t.}\quad &\textstyle\sum_{e\in E} x_e L_e \succeq \hat{\gamma}(I_n - e_0 \otimes e_0),\\
&\textstyle\sum_{e\in E} x_e \leq q,\\
&\text{Constraints in (3.3) and (3.4)},\\
&x_e \in \{0,1\}^{|E|}.
\end{aligned} \tag{3.7}
$$

As expected, the cutting plane algorithm for the above BSDP in conjunction with bisection techniques to solve the original MISDP (3.1) to optimality is in very similar lines as discussed in Algorithm 2. Hence, we present just the pseudo code of the procedure in Algorithm 5 without discussing the details.

3.1.3 Performance of proposed algorithm

All the computations in this section were performed with the same computer specifics as mentioned in section 2.3.4.

As discussed in earlier sections, the semi-definite programming toolboxes in Mat-

[1] There are several ways to formulate the decision problem as a BSDP. For example, one can also aim to minimize the total weight of the augmented graph defined as $\sum_e w_e x_e$. We chose to minimize the degree of a node as it gave reasonably good computational results.

Table 3.1: Comparison of computational time (CPU time) of the proposed algorithm for different limits on the diameter of the graph and γ^* is the optimal algebraic connectivity. The algorithm was implemented in CPLEX for instances involving 6 nodes.

Instance	**diameter $\leq$ 4**		**no diameter constraint**	
	γ^*	T_1 (sec)	γ^*	T_2 (sec)
1	39.352	7	559.539	8
2	39.920	4	546.915	8
3	67.270	6	765.744	6
4	50.262	10	713.925	5
5	31.218	8	569.959	4
6	52.344	8	662.326	7
7	35.513	7	637.331	6
8	38.677	7	704.89	6
9	46.427	11	574.132	5
10	40.945	7	597.241	5
11	36.770	10	586.950	9
12	42.885	6	587.027	5
13	30.880	8	569.482	10
14	47.583	3	543.145	6
15	37.277	4	517.401	9
16	37.439	11	704.228	8
17	51.434	10	639.456	3
18	42.476	3	620.974	10
19	29.934	3	576.275	4
20	46.980	6	536.366	6
21	25.955	6	630.748	9
22	49.220	6	601.309	4
23	53.282	6	607.615	6
24	45.909	5	524.214	6
25	48.120	3	549.210	3

Table 3.2: Comparison of computational time (CPU time) of the proposed algorithm for different limits on the diameter of the graph and γ^* is the optimal algebraic connectivity. The algorithm was implemented in CPLEX for instances involving 8 nodes.

Instance	**diameter $\leq$ 4**		**diameter $\leq$ 6**		**no diameter constraint**	
	γ^*	T_1 (sec)	γ^*	T_2 (sec)	γ^*	T_3 (sec)
1	66.1636	298.10	93.0846	184.26	631.739	495.23
2	39.2994	477.34	54.3061	416.43	631.883	980.98
3	44.8588	803.45	45.9793	634.54	604.213	4253.01
4	66.5337	394.02	78.7357	221.21	757.490	815.01
5	33.8383	519.28	53.8226	480.23	755.205	706.25
6	46.6083	1033.09	75.6113	349.12	513.994	586.34
7	51.1379	781.07	63.3915	385.17	550.717	949.30
8	42.8026	931.50	77.4458	319.51	807.108	333.93
9	58.1182	489.43	84.7166	348.82	769.641	482.55
10	50.5110	492.11	54.3155	323.33	646.711	1789.64
11	43.6888	791.01	107.1820	212.34	729.171	472.71
12	47.5213	693.13	82.2919	219.20	655.867	1061.16
13	42.4918	468.44	53.2514	698.21	698.129	1421.38
14	41.1752	445.26	48.9485	261.18	523.118	977.67
15	44.8202	518.13	63.8735	509.77	639.540	504.42
16	40.1853	540.19	72.1540	396.25	690.719	661.91
17	66.6196	480.70	108.0970	254.47	735.361	476.87
18	62.9801	499.78	69.1063	233.33	622.840	1372.58
19	40.7602	542.69	54.9466	343.04	650.096	236.65
20	60.1121	607.19	81.2138	209.15	607.008	590.38
21	66.3578	588.31	80.3600	408.78	609.370	730.82
22	42.8765	776.38	75.5561	458.80	666.251	734.43
23	42.7949	400.03	62.8144	638.11	444.903	942.26
24	63.1568	590.91	73.7841	333.03	680.411	804.27
25	31.3830	232.18	44.6972	231.16	630.107	818.93

Algorithm 5 :Algorithm for determining maximum algebraic connectivity with diameter constraint

Let $\mathfrak{F}$ denote a set of cuts which must be satisfied by any feasible solution

1: Input: Graph $G = (V, E, w_e)$, $e \in E$, a root vertex r, diameter D and a finite number of Fiedler vectors, $v_i, i = 1 \dots M$
2: Choose a maximum spanning tree as an initial feasible solution, x^*
3: $\hat{\gamma} \leftarrow \lambda_2(L(x^*))$
4: **loop**
5: $\mathfrak{F} \leftarrow \emptyset$
6: Solve:

$$\begin{aligned} \min \quad & \textstyle\sum_{e \in \delta(r)} x_e, \\ \text{s.t.} \quad & \textstyle\sum_{e \in E} x_e (v_i \cdot L_e v_i) \geq \hat{\gamma} \quad \forall i = 1, .., M, \\ & \textstyle\sum_{e \in E} x_e \leq q, \\ & x_e \in \{0, 1\}^{|E|}, \\ & \text{Constraints in (3.3) and (3.4),} \\ & x_e \text{ satisfies the constraints in } \mathfrak{F}. \end{aligned} \tag{3.8}$$

7: **if** the above ILP is infeasible **then**
8: **break loop** $\{x^*$ is the optimal solution with maximum algebraic connectivity$\}$
9: **else**
10: Let x^* be an optimal solution to the above ILP. Let γ^* and v^* be the algebraic connectivity and the Fiedler vector corresponding to x^* respectively.
11: **if** $\sum_{e \in E} x_e^* L_e \not\succeq \gamma^* (I_n - e_0 \otimes e_0)$ **then**
12: Augment $\mathfrak{F}$ with a constraint $\sum_{e \in E} x_e (v^* \cdot L_e v^*) \geq \gamma^*$.
13: Go to step 6.
14: **end if**
15: **end if**
16: $\hat{\gamma} \leftarrow \hat{\gamma} + \epsilon$ $\{$let ϵ be a small number$\}$
17: **end loop**

lab could not be used to solve the proposed formulation with the semi-definite and diameter constraints even for instances with 6 nodes primarily due to the inefficient memory management. However, due to the combinatorial explosion resulting from the increased size of the problem, the proposed algorithm with CPLEX solver could provide optimal solutions in a reasonable amount of run time for instances upto 8 nodes.

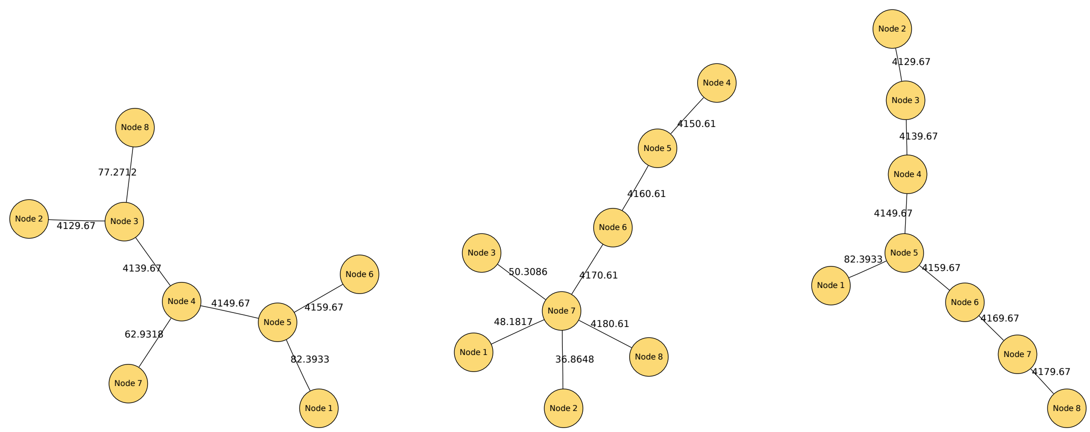
We shall now compare the computational times of the proposed algorithm to obtain optimal solutions for different values of the bound on the diameter. The results shown in Tables (3.1) and (3.2) are for 25 random instances generated for networks with 6 and 8 nodes, respectively. Based on the results in Table (3.1), we observed that the average run time for obtaining optimal solution for the 6 nodes problem with diameter constraint was (average T_1) $6.6s$ and without diameter constraint was (average T_2) $6.3s$. Based on the results in Table (3.2), we observed that the average run time for the problem without diameter constraints (average $T_3 = 927.95s$) was 1.61 times greater than the average run time for the problem with diameter ≤ 4 (average $T_1 = 575.75s$) and 2.56 times greater than the average run time for the problem with diameter ≤ 6 (average $T_2 = 362.77s$). Optimal networks with various diameters corresponding to instances 1 and 2 of Table (3.2) with 8 nodes may be found in Figure 3.2.

3.2 Maximization of algebraic connectivity with power consumption constraint

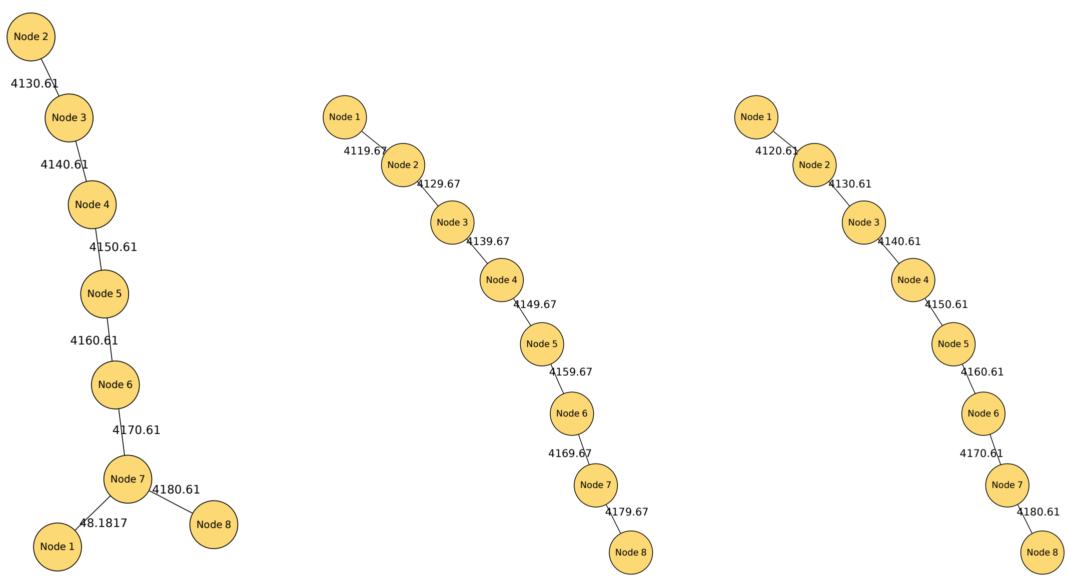
In this section, we mathematically formulate the total power consumed by the UAV network as a function of the eigenvalues of the Laplacian and later pose the problem of synthesizing a robust backbone UAV network subject to power consumption constraint as a MISDP problem. Essentially, the problem is to determine the backbone UAV network with maximum algebraic connectivity subject to the constraints on the total power consumed by the network and the number of communication links.

3.2.1 Related literature

The idea of using UAVs to communicate data has been proposed in the literature; for example, the use of UAVs as relays in disaster areas has been proposed in [46], [47],



(a) Instance 1, diameter ≤ 4 (b) Instance 2, diameter ≤ 4 (c) Instance 1, diameter ≤ 6



(d) Instance 2, diameter ≤ 6 (e) Instance 1, no diameter constraint (f) Instance 2, no diameter constraint

Figure 3.2: (a),(c),(e) correspond to optimal networks with maximum algebraic connectivity subject to various diameter constraints for instance 1 (from Table (3.2)). Similar plots for instance 2 are also shown in (b),(d),(e).

[48], [49], [50] to facilitate a mobile communication network connecting the emergency responders, control towers and different agencies, thereby enabling a timely exchange of information between the relevant entities. A similar architecture was envisioned in [51] for GPS denied navigation of UAVs. Employing some UAVs primarily as data transmitters has several advantages. For instance, each vehicle may not have the high-power transmitter and antennas to communicate to the ground station, and even if it does, such direct links are not suitable for environments with obstructions or non-line-of-sight communications[47]. In addition, the UAVs may be operating in dynamic environments where regular cellular towers are either damaged or non-existent. For these reasons, researchers have proposed meshing architectures [47],[52]. In this architecture, UAVs with a higher communication capability act as mobile base stations (also referred to as backbone nodes) and its primary job is to connect the individual vehicles or the regular nodes with limited communication capability to the control stations. A typical example of such a network is shown in Figure 3.3. Various objectives have been considered in the recent work on optimization of networks with backbone nodes; for example, in [53], the objective is to minimize the number of mobile backbone nodes so that all the regular nodes are connected; in [54], the objective is to optimally chose the location of mobile backbone nodes so as to maximize the number of regular nodes achieving a minimum throughput. However, in [53, 54], either the backbone nodes are not allowed to communicate with each other or the connectivity among the backbone nodes is enforced by requiring a minimum number of communication links that connect them.

In this work, we consider another objective to optimize the network of mobile nodes; this objective better reflects the robustness of connectivity among the backbone nodes due to random or unexpected failure of communication networks.

The problem dealing with maximization of algebraic connectivity subject to

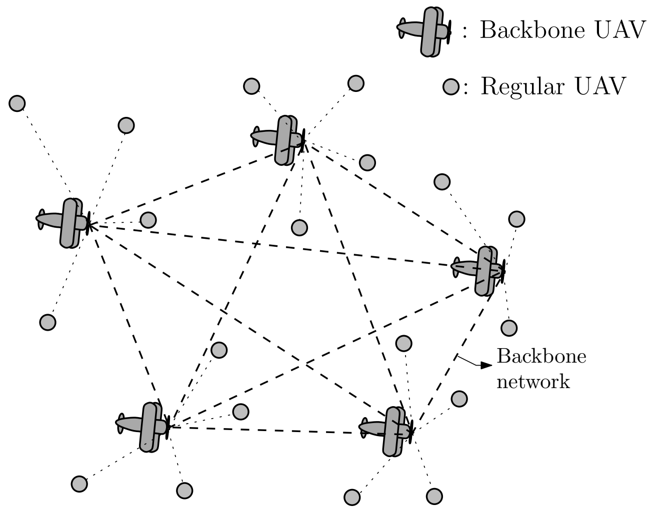


Figure 3.3: A typical representation of the UAV backbone network where backbone UAVs/nodes provide communication support to the regular nodes and each regular node is assigned to one backbone node as shown.

wiring cost constraint considered by Varshney [14] is closely related to the proposed problem. Varshney proposes the use of reverse convex program to find a relaxed solution and proposes the use of rounding to get a feasible solution from the relaxed solution. Their work lack the development of a systematic procedure to obtain optimal network. Also, the numerical results presented in [14] are limited to instances of problems with at most 7 nodes.

Therefore, in this section, we propose an algorithm based on cutting plane technique to determine an optimal network for the problem of maximizing algebraic connectivity subject to power consumption constraint. Lastly, we apply a 2-opt heuristic (developed in section 2.4.1) to find feasible solutions, and use a simple bound on the optimal algebraic connectivity stemming from the resource constraint to estimate the quality of the feasible solutions.

3.2.2 Mathematical formulation of the power consumption constraint

In this subsection, we first formulate the power consumption constraint, and in the subsequent subsections, we pose the algebraic connectivity problem with all the resource constraints as a Mixed Integer Semi-Definite Program.

One may sometimes have the choice of positioning UAVs that serve as backbone nodes. In this case, there is a natural problem of determining the positions of the UAVs such that the convex hull of the projection of the locations of these UAVs on to two dimensions is at least equal to the minimum area of coverage, A_0. Since the total power consumed by these UAVs is an important consideration, as defined in the introduction, if there is a communication link between the i^{th} and j^{th} UAV, the power consumed is given by $P_{ij} := \alpha_{ij} d_{ij}^2$. The total power consumed by the collection of backbone UAVs is $\sum_{(i,j)\in E} P_{ij} x_{ij}$. To formulate the power consumption constraint, we now pose the following subproblem: Given the network topology of the backbone UAVs, what would be an optimal placement of the nodes such that

a. the total power consumed is minimum, and

b. the projected area of the convex hull of the backbone UAVs in the ground plane in is at least A_0?

Clearly, if one were to solve this subproblem, one can solve the original problem of synthesizing the network of backbone UAVs by considering only those topologies that result in the total power consumption within the specified budget, P_{max}, and then picking one network topology among them with the maximum algebraic connectivity. Hence, we shall discuss the formulation of this subproblem in the remainder of this section.

Suppose the location of the i^{th} UAV is given by (a_i, b_i, c_i), so that for a given topology of communication (provided by the set E_g of edges), the total power consumed may be written as:

$$\sum_{(i,j)\in E_g} P_{ij} = \sum_{(i,j)\in E_g} \alpha_{ij}((a_i - a_j)^2 + (b_i - b_j)^2 + (c_i - c_j)^2), \tag{3.9a}$$

$$= \mathbf{a} \cdot \sum_{e=(i,j)\in E_g} \alpha_{ij} L_e \mathbf{a} + \mathbf{b} \cdot \sum_{e=(i,j)\in E_g} \alpha_{ij} L_e \mathbf{b} + \mathbf{c} \cdot \sum_{e=(i,j)\in E_g} \alpha_{ij} L_e \mathbf{c}, \tag{3.9b}$$

where L_e is the local Laplacian matrix corresponding to the edge e and $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$ are the vectors whose i^{th} components provide respectively the a, b and c coordinates of the i^{th} UAV.

In order to improve spatial spread, we will require that the area of the convex hull of the projections of UAVs' locations on the horizontal plane be at least a specified amount, say A_0. Without loss of generality, assuming that the origin is at the centroid of the convex hull as shown in figure 3.4 (for a simple case of five UAVs), we have the following constraints, $\mathbf{1} \cdot \mathbf{a} = 0$ and $\mathbf{1} \cdot \mathbf{b} = 0$ where $\mathbf{1} \in \mathbb{R}^n$ is a vector of ones. Since we are dealing with the convex hull of the projected locations of the UAVs, one may number the projected locations and order them appropriately, so that the area may be triangulated with the centroid being one of the vertices of every triangle in the triangulation and that the area of the convex hull may be expressed as a bilinear function: $A(\mathbf{a}, \mathbf{b})$ has the the property that $A(\mathbf{a}, \mathbf{b}) = -A(\mathbf{b}, \mathbf{a})$. Hence, for some skew-symmetric matrix, Ω, one may express the area as:

$$A(\mathbf{a}, \mathbf{b}) = \mathbf{a} \cdot \Omega \mathbf{b}.$$

For the case of five nodes shown in Figure 3.4, the skew symmetric matrix is given by:

$$\Omega = \frac{1}{2}\begin{pmatrix} 0 & 1 & 0 & 0 & -1 \\ -1 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 1 \\ 1 & 0 & 0 & -1 & 0 \end{pmatrix}$$

Since Ω is skew-symmetric, $\mathbf{b} \cdot \Omega\mathbf{b} = 0$, the component of the vector $\mathbf{a}$ along $\mathbf{b}$ will not contribute to the projected area. Therefore, we may require that $\mathbf{a} \cdot \mathbf{b} = 0$ so that $\mathbf{a}$ contributes fully to the projected area.

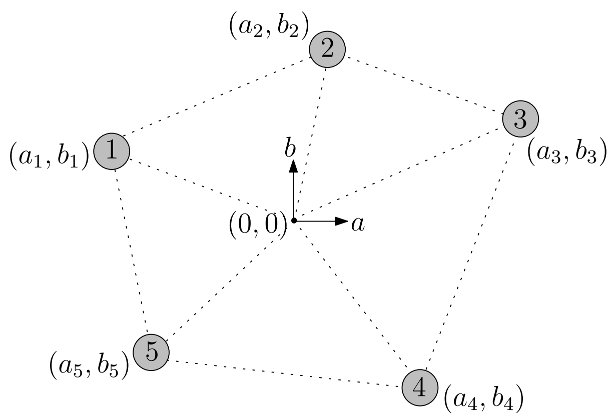


Figure 3.4: Convex hull of the projections of five UAVs' locations on the horizontal plane with the centroid of the area at the origin.

Imposing the non-linear constraint $\mathbf{a} \cdot \Omega\mathbf{b} \geq A_0$ is hard; for this reason, we will alternatively specify the constraint on the spread of UAVs indirectly through requiring the variance in their coordinates to be at least R^2 units. Hence, we recast the problem for locating the UAVs so that the total power consumption is a minimum as:

$$\begin{aligned} \text{Minimize} \quad & P = \mathbf{a} \cdot L\mathbf{a} + \mathbf{b} \cdot L\mathbf{b} + \mathbf{c} \cdot L\mathbf{c} \\ \text{subject to} \quad & \mathbf{1} \cdot \mathbf{a} = 0, \ \mathbf{1} \cdot \mathbf{b} = 0 \\ & \mathbf{a} \cdot \mathbf{b} = 0, \\ & \mathbf{a} \cdot \mathbf{a} \geq R^2, \ \mathbf{b} \cdot \mathbf{b} \geq R^2. \end{aligned}$$

Therefore, by the variational characterization of eigenvalues, the minimum total power consumed for a given network topology of the UAVs is given by:

$$R^2(\lambda_2(L) + \lambda_3(L)),$$

where the optimal $\mathbf{c}$ is along the vector $\mathbf{1}$ (corresponding to the zero eigenvalue of L), $\mathbf{a}$ and $\mathbf{b}$ along the eigenvectors corresponding to the second and third smallest eigenvalues of L. In other words, the optimal location of UAVs is such that they must lie in the same plane (i.e., with their c coordinates being the same) and their a and b coordinates must lie along the eigenvectors corresponding to the second and third smallest eigenvalues of the Laplacian for the specified network topology so that the total power consumption of the communication network is a minimum.

3.2.3 Problem of maximizing algebraic connectivity with power consumption constraint

As we discussed in section 3.2.2, we showed that, given a network topology for the UAVs, an optimal placement of the nodes would be along the second and third eigenvector directions and that the minimum power consumed would be $R^2(\lambda_2(L)+$

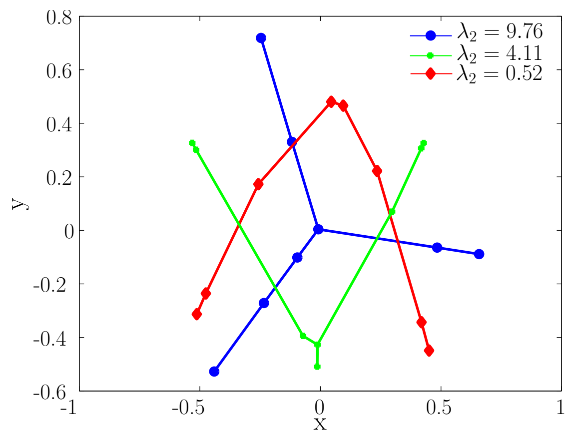
$\lambda_3(L)$) where L is the Laplacian of that particular topology. Naturally, one would also be interested in synthesizing a network topology (x) which connects all the UAVs and such that a) the topology is robust/well-connected against random failure of links and b) the total power consumed is bounded by a prescribed upper bound ($\tilde{P}_{max}$), i.e,

$$\lambda_2(L(x)) + \lambda_3(L(x)) \leq P_{max}$$

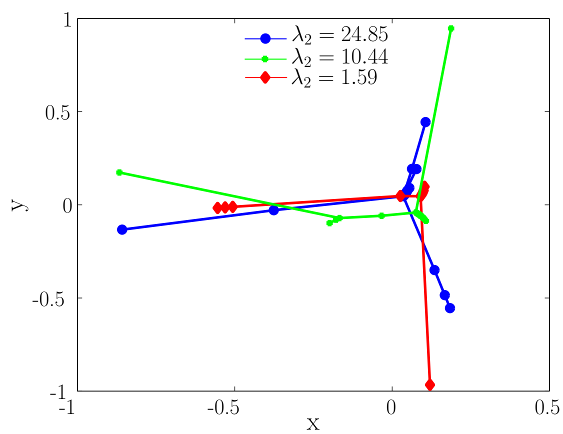
where $P_{max} = \tilde{P}_{max}/R^2$.

In order to synthesize a well-connected topology, we chose to maximize the λ_2 or the algebraic connectivity of the weighted Laplacian of the network where weights of the edges correspond to the proportionality constant, α_{ij}. As discussed in the introduction, algebraic connectivity has been extensively used in the literature as a measure for robustness of networks. This objective of maximizing algebraic connectivity is reasonable since we observed that the topology of networks with higher λ_2 tend to spread the UAV locations better. Sample feasible networks for 8 and 20 nodes with varying values of λ_2 are shown in figure 3.5. In the case of 8 nodes, networks with lower λ_2 are weakly connected since a random removal of any node/edge can disconnect the entire network. Networks with lower λ_2 seem to have a higher diameter which can incur more delays in the communication of data among UAVs. The situation is similar in the case of 20 nodes in figure 3.5.

Based on the model of the power constraint and the notation defined earlier, the problem of choosing at most q (positive integer) edges from E so that the algebraic connectivity of the augmented network is maximized can be posed as follows:



(a) 8 nodes

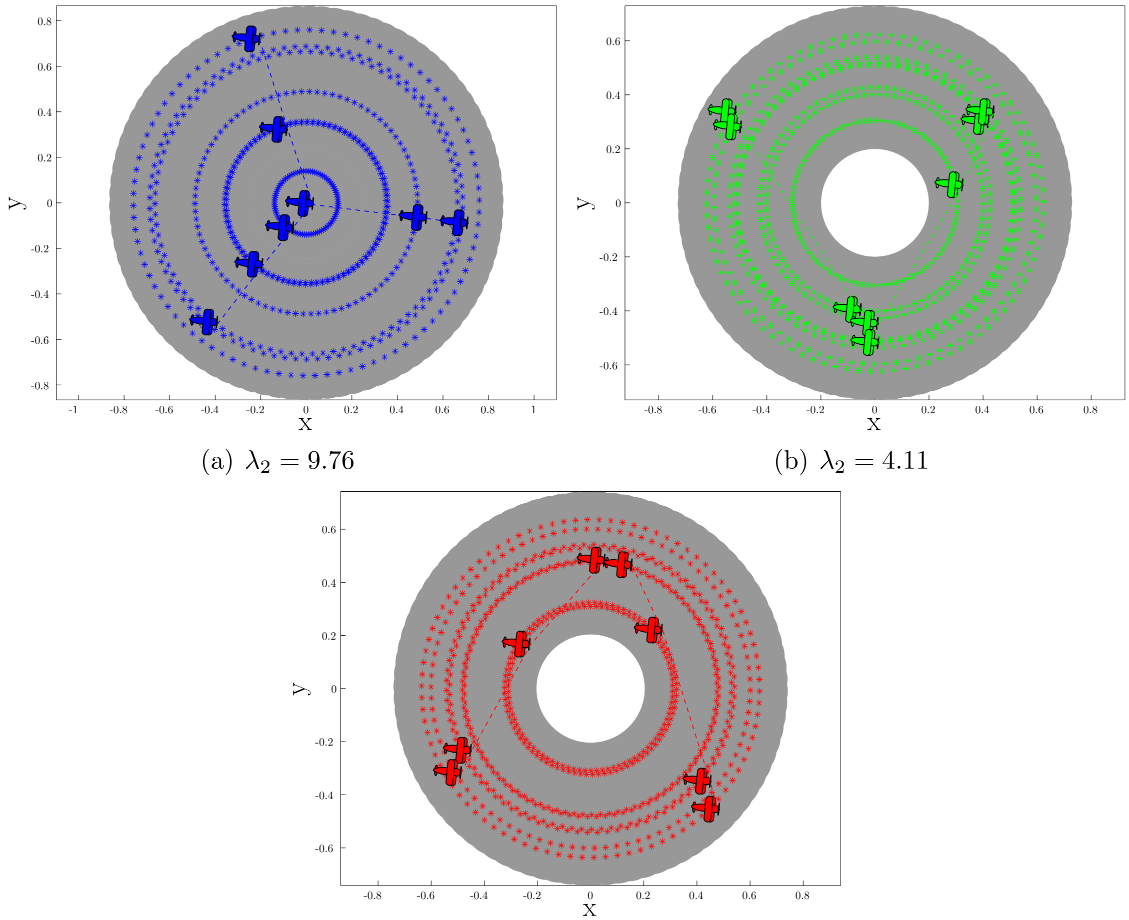


(b) 20 nodes

Figure 3.5: This figure represents the positioning of UAVs for various objective values subject to power consumption constraint. Maximizing $\lambda_2(L)$ indicates that the UAV locations are more uniformly distributed with well connected topologies.

$$\begin{aligned} & \max \lambda_2(L(x)), \\ \text{s.t.} \quad & \lambda_2(L(x)) + \lambda_3(L(x)) \leq P_{max}, \\ & \textstyle\sum_{e \in E} x_e \leq q, \\ & x \in \{0,1\}^{|E|}. \end{aligned} \tag{3.10}$$

Since the objective of this problem is non-linear, it can be converted to a more tractable but an equivalent MISDP with the non-linear power consumption constraint



(a) $\lambda_2 = 9.76$ (b) $\lambda_2 = 4.11$ (c) $\lambda_2 = 0.52$

Figure 3.6: This figure represents the trajectories of the UAVs when the backbone UAV network (8 nodes) is subject to a rigid body rotation by 360 degrees about their respective centroids. Radius of communication of 0.1 was chosen for all the UAVs. Note that, the network corresponding to largest λ_2 value has the maximum coverage unlike the networks with with lower λ_2.

as follows:

$$
\begin{aligned}
&\max \gamma, \\
\text{s.t.} \quad & L(x) \succeq \gamma(I_n - e_0 \otimes e_0), \\
& \lambda_2(L(x)) + \lambda_3(L(x)) \leq P_{max}, \\
& \textstyle\sum_{e \in E} x_e \leq q, \\
& x \in \{0,1\}^{|E|}.
\end{aligned}
\tag{3.11}
$$

The correctness of this formulation exactly follows the proof given in section 2.1.1.

3.2.4 Algorithm for determining maximum algebraic connectivity with power consumption constraint

Currently, efficient tools for solving MISDPs are not available. In this section, we extend the idea of the algorithm proposed for solving the **BP**, which was based on the cutting plane method. The basic idea of this method is to find an outer approximation (relaxation) of the feasible set of the MISDP problem and solve the optimization problem over the outer approximation (which we refer to as a relaxed problem). If the optimal solution for the relaxed problem is feasible for the MISDP problem, it is also clearly optimal for the MISDP problem; otherwise, one must refine the outer approximation, e.g., via the introduction of additional linear inequalities (referred to as cuts). One may then iteratively refine the outer approximation until the optimal solution of the outer approximation is also feasible for the MISDP.

We initially tried solving the MISDP problem by relaxing the non-convex power consumption constraint and adding linear inequalities to cut off optimal solutions of the relaxed semi-definite programs. We used Matlab and state-of-the-art semi-definite solvers such as the Sedumi for this implementation and found that it could not handle problems of size greater than 5 nodes. Hence, we opted to pose this problem as a MILP problem so that the available high performance solvers such as

CPLEX can be used.

It is important to understand the sources of difficulty when implementing a cutting plane algorithm for the MISDP problem under consideration. For every vector $v \in \Re^n$ such that $v \cdot \mathbf{1} = 0$ and $\|v\|_2 = 1$, the semi-definiteness requirement is equivalent to

$$\sum_{e \in E} x_e v \cdot L_e v \geq \gamma.$$

In essence, for every such vector v, there is a linear constraint in x_e and γ and since the number of vectors v satisfying $v \cdot \mathbf{1} = 0$ and $\|v\|_2 = 1$ is uncountably infinite, the semi-definite requirement is equivalent to an uncountable number of linear constraints in the discrete variables $x_e,\ e \in E$ and γ. This constraint makes the use of standard ILP tools difficult. One can pick any finite subset of these linear constraints to construct a polyhedral outer approximation.

The non-convex nature of the power consumption constraint in (3.11) makes it difficult to be taken care of directly by the standard ILP tools. For this reason, we relax this constraint and provide a method for the construction of "cuts" that cut off any feasible solution of the relaxed problem that is not feasible for the MISDP.

The schema for solving the MISDP is as follows:

- **Step 0: Initialization:** Pick a finite set of unit vectors, say $v_i,\ i = 1, 2, \ldots, M$ that are perpendicular to $\mathbf{1}$, and the polyhedral outer approximation, $\mathcal{P}_0$ is the feasible set of the inequalities:

  $$\sum_{e \in E} x_e v_i \cdot L_e v_i \geq \gamma, \quad i = 1, 2, \ldots, M,$$

  and $x_e \in \{0, 1\},\ e \in E$.

- **Step 1: Refinement of Polyhedral Outer Approximation:** This step involves the developing of "cuts". Since the initial polyhedral approximation relaxes the semi-definite constraint and the power consumption constraint (3.11), we outline a method to find the linear inequalities that cut off solutions that are not feasible for either of these constraints.

a. **Cut for the semi-definite constraint violation:** A violation of semi-definite constraint can result in a graph being disconnected. However, we augment the constraints from a multicommodity flow formulation [32] in order to ensure that the optimal solution of the relaxed problem is not disconnected.

   If the semi-definiteness requirement (3.11) is violated by the optimal solution given by $(x_e^*, \gamma^*), e \in E$, (which we assume is connected now) one may readily use the eigenvalue cut, i.e., if

$$\sum_{e \in E} x_e^* L_e - \gamma^*(I_n - e_0 \otimes e_0) \not\succeq 0,$$

   then there is at least one eigenvalue of the matrix on the left hand side of the above inequality that is negative. Hence, if one were to consider the corresponding normalized eigenvector, say v, then

$$\sum_{e \in E} x_e^* v \cdot L_e v < \gamma^*,$$

   and hence, one may refine the polyhedral outer approximation by augmenting an additional constraint that must be satisfied by any feasible solution to MISDP:

$$\sum_{e \in E} x_e v \cdot L_e v \geq \gamma. \tag{3.12}$$

This additional constraint ensures that the solution $x_e^*,\ e \in E$ that was optimal for the relaxed problem will not be feasible now for the augmented set of inequalities and the feasible set of the augmented set of inequalities is a refined outer approximation. Also, it can be easily proved that the inequality (3.12) is a valid inequality for the original problem based on the variational characterization of the eigenvalues.

b. **Cut for the power consumption constraint violation:** If the constraint on power consumption in (3.11) is violated by x_e^* where $e \in E^* \subset E$, one may introduce a constraint requiring that not all the edges of the optimal solution may be used, and can introduce a branch according to

$$\sum_{e \in E^*} x_e \leq \sum_{e \in E^*} x_e^* - 1,$$

or

$$\sum_{e \in E^*} x_e \geq \sum_{e \in E^*} x_e^* + 1.$$

Since we seek spanning trees in the numerical examples, we only require the former constraint, namely

$$\sum_{e \in E^*} x_e \leq \sum_{e \in E^*} x_e^* - 1, \tag{3.13}$$

to be enforced in the algorithm. It can again be easily proved that inequality (3.13) is a valid inequality as follows: Since (3.13) is an inequality on the number of edges in the spanning tree, let τ_i be the i^{th} spanning tree among n^{n-2} possible spanning trees and E_{τ_i} be the edges in τ_i^{th} spanning tree. Then we know that

$$0 \leq |E_{\tau_i} \cap E_{\tau_j}| \leq n-2 \quad \forall\, i,j = 1,\ldots,n^{n-2}, i \neq j.$$

From this, it is clear that

$$\sum_{e \in E_{\tau_i}} x_e \leq n-2$$

uniquely eliminates τ_i retaining all other spanning trees valid.

- **Step 2:** Solve the relaxed problem, i.e., solve the optimization problem over the feasible set of the refined approximation using ILP solvers to get an updated solution $x_e^*,\ e \in E$ and go to Step 1.

The pseudo code of this procedure is outlined in Algorithm 6. The Algorithm 6 is guaranteed to terminate in finite number of iterations since the number of feasible solutions for this problem is finite (n^{n-2} for a problem with n nodes). The cut for eliminating solutions that do not satisfy the semi-definite constraint is shown in steps 7 through 11 of Algorithm 6. Step 12 of Algorithm 6 corresponds to the cut for eliminating solutions that violate the power consumption constraint.

3.2.5 Performance of proposed algorithm

In this section, we discuss the computational performance of the proposed algorithm (Algorithm 6) to solve the problem of maximizing algebraic connectivity subject to the consumption constraint. All the computations in this section were performed with the same computer specifics as mentioned in section 2.3.4.

In order to solve the MISDP in step 6 of the algorithm (6), we used the Sedumi solver in Matlab and found that they could not handle problems with more than

Algorithm 6 :Algorithm for determining maximum algebraic connectivity with power constraint

Notation: Let $\hat{I} = (I_n - e_0 \otimes e_0)$.
Let $\mathfrak{F}$ denote a set of cuts which must be satisfied by any feasible solution

1: Input: A graph $G = (V, E)$, a weight (w_e) for each edge $e \in E$, P_{max}, and a finite number of Fiedler vectors, $v_i, i = 1 \ldots M$
2: Choose any spanning tree, x_0 such that $\lambda_2(L(x_0)) + \lambda_3(L(x_0)) > P_{max}$
3: $x^* \leftarrow x_0$
4: $\mathfrak{F} \leftarrow \emptyset$
5: **while** $\lambda_2(L(x^*)) + \lambda_3(L(x^*)) > P_{max}$ **do**
6: Solve:

$$\begin{array}{ll} \max & \gamma, \\ \text{s.t.} & \sum_{e \in E} x_e(v_i \cdot L_e v_i) \geq \gamma \quad \forall i = 1, .., M, \\ & \sum_{e \in E} x_e \leq q, \\ & \sum_{e \in \delta(S)} x_e \geq 1, \quad \forall\, S \subset V, \\ & x_e \in \{0, 1\}^{|E|}, \\ & x_e \text{ satisfies the constraints in } \mathfrak{F}. \end{array}$$

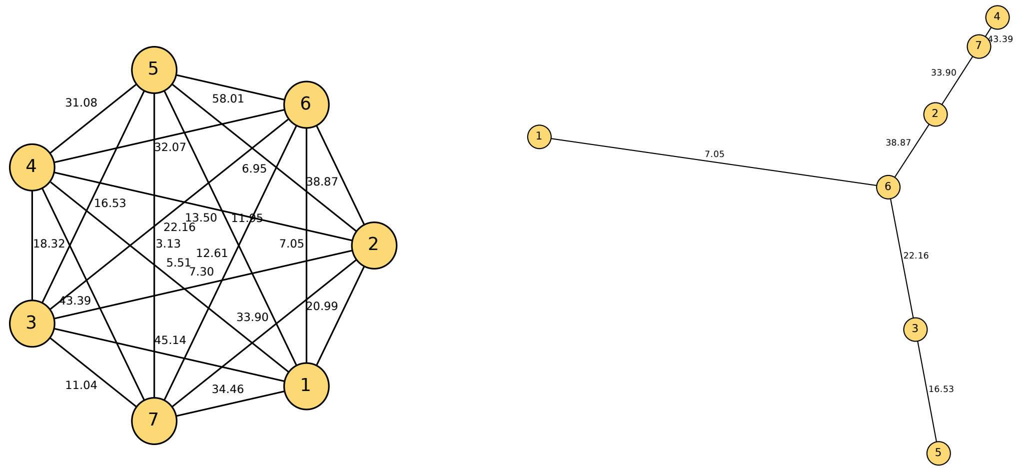
Let (x^*, γ^*) be an optimal solution to the above problem.
7: **if** $\sum_{e \in E} x_e^* L_e \not\succeq \gamma^* \hat{I}$ **then**
8: Find the Fiedler vector v^* corresponding to x^*.
9: Augment $\mathfrak{F}$ with a constraint $v^* \cdot L(x^*) v^* \geq \gamma^*$.
10: Go to step 6.
11: **end if**
12: If $\lambda_2(L(x^*)) + \lambda_3(L(x^*)) \not\leq P_{max}$, augment $\mathfrak{F}$ with a cut $\mathbf{1} \cdot x \leq \mathbf{1} \cdot x^* - 1$.
13: Go to step 6.
14: **end while**

five nodes. However, solving the same MISDP using the proposed cutting plane algorithm performed comparatively better using the CPLEX solver and could handle up to eight nodes though the computation time was in the order of hours.

Table 3.3 shows the optimal solutions for ten random instances with seven nodes. For the case of 7 nodes, we assumed P_{max} equal to fifteen to ensure that the problem had a feasible solution. From this table, it can be observed that the average computational time to obtain an optimal solution based on the proposed algorithm was around seven hours. Even though the algorithm (6) provides successive tighter polyhedral approximations with the augmentation of valid inequalities, the convergence to an optimal solution is very slow. The reason for the slow convergence can be attributed to the strength of the cuts added due to the violation of the power consumption constraint. These cuts are merely solution elimination constraints which eliminates only the current infeasible integral solution. Generating more valid and stronger cuts at every iteration of the algorithm can possibly reduce the computation time to obtain optimal solutions. A sample network with maximum algebraic connectivity satisfying the power consumption constraint for instance #1 of Table 3.3 is shown in Figure 3.7.

Table 3.3: Computational performance of algorithm (6) to solve the problem of maximizing algebraic connectivity subject to the power consumption constraint. T_1 corresponds to the CPU time taken by CPLEX solver to solve instances with 7 nodes. Note that $\lambda_2^*+\lambda_3^*$ represents the power incurred by each network with optimal connectivity as indicated under λ_2^*. P_{max} is chosen to be equal to fifteen for all the instances.

Instance	λ_2^*	$\lambda_2^* + \lambda_3^*$	T_1 (sec)
1	7.1278	14.7192	12291.28
2	7.1457	14.9988	10350.02
3	6.7300	14.8166	58481.36
4	6.9879	14.9829	58845.25
5	7.2684	14.8568	28891.07
6	6.4437	14.9999	26543.17
7	7.0472	14.9261	17218.10
8	7.0047	14.9225	27589.19
9	7.0526	14.6940	16413.76
10	7.1569	14.5328	12353.42
Avg.			26894.41



(a) Complete graph for $n = 7$ (b) Optimal network for $n = 7$

Figure 3.7: In this figure, part (a) represents a complete graph of 7 nodes with random edge weights. Part (b) represents an optimal network with maximum algebraic connectivity ($\lambda_2^* = 7.1278$) synthesized from the complete graph which satisfies the power consumption constraint ($\lambda_2^* + \lambda_3^* \leq 15$). Note that the locations of the nodes in (b) are aligned in the second and third eigenvector directions.

3.2.6 Lower bounds based on the BSDP approach

In section 3.2.5, we observed that the convergence of the algorithm (6) to optimality was very slow and was in the order of many hours. Hence, in this section, we modify the proposed algorithm based on the techniques developed in the earlier sections on BSDP approach to obtain quick lower bounds and corresponding feasible solutions.

The MISDP in Step 6 of the algorithm (6) can also be solved to optimality based on the BSDP approach as discussed in section 2.3.5. The basic idea of this approach is to solve the feasibility problem where we are interested to obtain a network with a specified level of connectivity which satisfies the power consumption constraint. For completeness, we summarize the algorithm to obtain lower bounds for the problem of maximizing algebraic connectivity with power consumption constraint in (7). Clearly, at every iteration of this algorithm, $\hat{\gamma}$ serves as the lower bound whose value monotonically increases until the optimality is reached. Correspondingly, x^* at every iteration serves as the feasible network satisfying the power consumption constraint.

Since for this particular problem, we know that $\lambda_2(L(x)) + \lambda_3(L(x)) \leq P_{max}$ and $\lambda_2(L(x)) \leq \lambda_3(L(x))$, a trivial upper bound on the algebraic connectivity would be

$$\lambda_2(L(x)) \leq \frac{P_{max}}{2}.$$

Hence, we use this upper bound to corroborate the quality of the lower bounds obtained for larger instances.

Construction of an initial feasible solution: As we discussed in the lower bounding procedure in (7), construction of an initial feasible solution is the first step. For the problem we have considered in this section, an initial feasible solution is any spanning tree whose power incurred will be less than a given value of upper bound. Though the eigenvalues of the Laplacian are non-linear functions of the edge weights of the graph, we observed that the values of $\lambda_2 + \lambda_3$ are reasonably low for spanning trees with relatively low edge weights as seen in figure 3.8. Since we found that enumerating a fixed number of spanning trees (10000 trees) starting from a minimum spanning tree using the algorithm discussed in [34] was computationally easier, the best initial feasible solution which satisfies the power consumption constraint was chosen from these enumerated trees.

Quality of lower bounds: Computationally, we observed that the proposed lower

Algorithm 7 : Lower bounding algorithm (BSDP approach)

Let $\mathfrak{F}$ denote a set of cuts which must be satisfied by any feasible solution

1: Input: Graph $G = (V, E, w_e)$, $e \in E$, a root vertex, r, P_{max} and a finite number of Fiedler vectors, $v_i, i = 1 \dots M$

2: Choose any spanning tree, x^* such that $\lambda_2(L(x^*)) + \lambda_3(L(x^*)) \leq P_{max}$

3: $\hat{\gamma} \leftarrow \lambda_2(L(x^*))$

4: **loop**

5: $\quad\mathfrak{F} \leftarrow \emptyset$

6: $\quad$Solve:

$$\begin{array}{ll} \min & \sum_{e\in\delta(r)} x_e, \\ \text{s.t.} & \sum_{e\in E} x_e(v_i \cdot L_e v_i) \geq \hat{\gamma} \quad \forall i = 1, .., M, \\ & \sum_{e\in E} x_e \leq q, \\ & \sum_{e\in\delta(S)} x_e \geq 1, \quad \forall\ S \subset V, \\ & x_e \in \{0,1\}^{|E|}, \\ & x_e \text{ satisfies the constraints in } \mathfrak{F}. \end{array} \tag{3.14}$$

7: $\quad$**if** the above ILP is infeasible **then**

8: $\quad\quad$**break loop** {x^* is an optimal solution with maximum algebraic connectivity}

9: $\quad$**else**

10: $\quad\quad$Let x^* be an optimal solution to the above ILP. Let γ^* and v^* be the algebraic connectivity and the Fiedler vector corresponding to x^* respectively.

11: $\quad\quad$**if** $\sum_{e\in E} x_e^* L_e \not\succeq \gamma^*(I_n - e_0 \otimes e_0)$ **then**

12: $\quad\quad\quad$Augment $\mathfrak{F}$ with a constraint $\sum_{e\in E} x_e(v^* \cdot L_e v^*) \geq \gamma^*$.

13: $\quad\quad\quad$Go to step 6.

14: $\quad\quad$**end if**

15: $\quad$**end if**

16: $\quad$**if** $\lambda_2(L(x^*)) + \lambda_3(L(x^*)) \nleq P_{max}$ **then**

17: $\quad\quad$augment $\mathfrak{F}$ with a cut $\mathbf{1} \cdot x \leq \mathbf{1} \cdot x^* - 1$.

18: $\quad$**else**

19: $\quad\quad\hat{\gamma} \leftarrow \hat{\gamma} + \epsilon$ {let ϵ be a small number}

20: $\quad\quad$Go to step 6.

21: $\quad$**end if**

22: **end loop**

bounding procedure based on the BSDP approach provided very good quality lower bounds. For the seven nodes problem, we limited the computation time of the lower bounding algorithm to three minutes. As shown in Table 3.4, on an average,

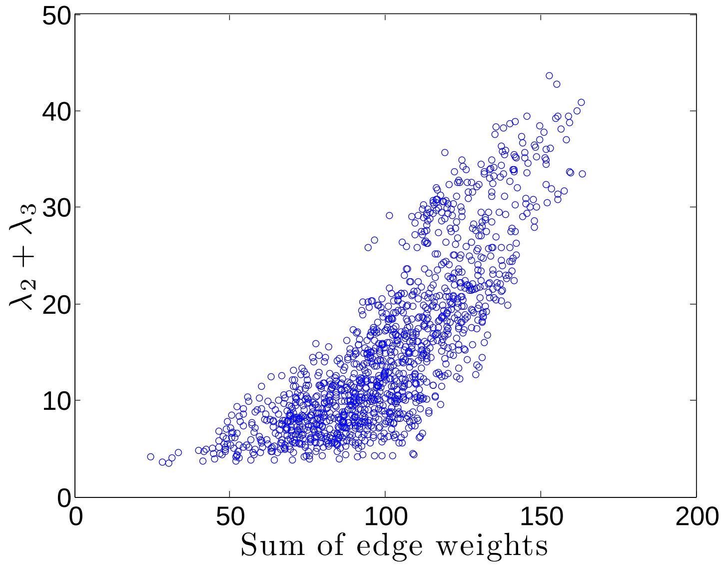


Figure 3.8: Enumeration of all spanning trees for a random instance with six nodes. It can be observed that spanning trees with lesser sum of edge weights incur lesser power consumption.

the lower bound obtained was within 3.5 % from the optimal solution. This was indeed a tremendous improvement in terms of the computation time compared to the algorithm discussed in the previous section. Certainly, the computation time of the lower bounding procedure depends on the value chosen for P_{max}. As expected, higher the value of P_{max}, larger would be the computation time for the BSDP approach since the number of bisection steps would increase.

We also tested the computational performance of the lower bounding procedure for the case of ten nodes with P_{max} equal to thirty. The computation time was limited to three minutes. As shown in Table 3.5, on an average, the lower bound obtained was within 15.2 % from the upper bound as described earlier. Since, this percent gap was with respect to the upper bound, we expect this gap to further reduce when evaluated with respect to the optimal solutions.

Table 3.4: Quality of lower bounds obtained based on the BSDP approach for the problem with *seven* nodes. λ_2^{LB} represents the lower bound obtained by terminating the algorithm (7) in three minutes. The value of P_{max} is equal to fifteen.

Instance	**Optimal solution**		**Lower bound**	
	λ_2^*	$\lambda_2^* + \lambda_3^*$	${\lambda_2}^{LB}$	$\frac{\lambda_2^* - {\lambda_2}^{LB}}{\lambda_2^*}$ (% gap)
1	7.1278	14.7192	7.1067	0.3
2	7.1457	14.9988	6.8177	4.6
3	6.7300	14.8166	6.4897	3.6
4	6.9879	14.9829	6.6520	4.8
5	7.2684	14.8568	7.2608	0.1
6	6.4437	14.9999	6.3397	1.6
7	7.0472	14.9261	6.8744	2.5
8	7.0047	14.9225	6.7295	3.9
9	7.0526	14.6940	6.2335	11.6
10	7.1569	14.5328	6.9836	2.4
Avg.				3.5

3.2.7 Performance of 2-opt heuristic

In this section, we discuss the performance of 2-opt heuristic to obtain feasible solutions for the problem of maximizing algebraic connectivity under the power consumption constraint. Section 2.4.1 has dealt in detail with the k-opt heuristic for synthesizing feasible solutions for the **BP**. Since the extension of this heuristic for incorporating an additional constraint on the power consumption is quite straight forward, we do not delve into the details. Instead, for completeness, we summarize the 2-opt heuristic including the additional resource constraint in (8).

Solution quality of 2-opt heuristic: In Table 3.6, we present the solution quality of the 2-opt heuristic solutions with respect to the optimal solutions for the problem

Table 3.5: Quality of lower bounds obtained based on the BSDP approach for the problem with *ten* nodes. λ_2^{LB} represents the lower bound obtained by terminating the algorithm (7) in three minutes. Note that $\frac{P_{max}}{2}$ is an upper bound on the optimal solution and value of P_{max} is equal to thirty.

Instances	${\lambda_2}^{LB}$	$\frac{\frac{P_{max}}{2} - {\lambda_2}^{LB}}{\frac{P_{max}}{2}}$ (% gap)	${\lambda_2}^{LB} + {\lambda_3}^{LB}$
1	12.9106	14.0	29.176
2	12.4978	16.7	29.204
3	12.3475	17.7	29.623
4	12.7198	15.2	29.329
5	12.7786	14.8	29.393
6	12.2478	18.3	29.381
7	11.3301	24.5	28.804
8	14.0376	6.4	28.632
9	12.7420	15.1	28.062
10	13.5049	10.0	29.187
Avg.		15.2	

instances of seven nodes. We define the solution quality or percent gap as

$$\frac{\lambda_2^* - \lambda_2^{2opt}}{\lambda_2^*} * 100$$

where λ_2^* is the algebraic connectivity of the optimal solution and λ_2^{2opt} is the algebraic connectivity of the 2-opt heuristic solution. It is clear from Table 3.6 that the average percent gap of the 2-opt solution from the optimal solution was around one percent and many of the 2-opt solutions were indeed optimal. Also, we empirically observed that the 2-opt heuristic solutions were optimal for 60 % of the random (100) instances. Therefore, from this short numerical study, we observed that the performance of 2-opt heuristic was phenomenal since it could generate feasible solutions within one percent gap from the optimal solutions within a few seconds of the CPU time.

In Table 3.7, we present the scalability of 2-opt heuristic solutions for instances

Algorithm 8 : 2-opt exchange heuristic

$\mathcal{T}_0 \leftarrow$ Initial feasible solution satisfying resource constraints
$\lambda_0 \leftarrow \lambda_2(L(\mathcal{T}_0)))$
Input: P_{max}
for each pair of edges $\{(u_1, v_1), (u_2, v_2)\} \in \mathcal{T}_0$ **do**
    Let $\mathcal{T}_{opt}$ be the best spanning tree in the 2-exchange neighborhood of $\mathcal{T}_0$ obtained by replacing edges $\{(u_1, v_1), (u_2, v_2)\}$ in $\mathcal{T}_{opt}$ with a different pair of edges.
    if $\lambda_2(L(\mathcal{T}_{opt})) > \lambda_0$ and $\lambda_2(L(\mathcal{T}_{opt})) + \lambda_3(L(\mathcal{T}_{opt})) \leq P_{max}$ **then**
        $\mathcal{T}_0 \leftarrow \mathcal{T}_{opt}$
        $\lambda_0 \leftarrow \lambda_2(L(\mathcal{T}_{opt}))$
    end if
end for
$\mathcal{T}_0$ is the best spanning tree in the solution space with respect to the initial feasible solution

up to 25 nodes. Though we could not obtain optimal solutions for larger instances, as discussed earlier in section 3.2.6, we used the trivial upper bound on the optimal λ_2 which stems from the power consumption constraint. Hence, the percent gap in Table 3.7 is given by

$$\frac{\frac{P_{max}}{2} - \lambda_2^{2opt}}{\frac{P_{max}}{2}} * 100.$$

The average percent gap was an average value evaluated over ten random instances for each size of the problem. P_{max} values in Table 3.7 were chosen randomly such that there is existed a feasible solution. Again we observed that the 2-opt heuristic performed very well for larger instances and also the average percent gap reduced with the increase in the size of the problem. Certainly, the percent gap depends on the values of P_{max} chosen, that is, the larger the value of P_{max}, larger would be the percent gap. A sample network satisfying the power consumption constraint found by the 2-opt heuristic can be seen in figure 3.9.

Table 3.6: 2-opt heuristic solutions for the problem of maximizing algebraic connectivity with power consumption constraint. The results in this table are for instances with seven nodes. Note that $\lambda_2^* + \lambda_3^*$ represents the total power incurred by each network with optimal connectivity as indicated under λ_2^*

Instances	**Optimal solution**		**2-opt solution**		
	λ_2^*	$\lambda_2^* + \lambda_3^*$	λ_2^{2opt}	$\lambda_2^{2opt} + \lambda_3^{2opt}$	% gap
1	7.1278	14.7192	7.1278	14.7192	0.00
2	7.1457	14.9988	7.0880	14.9557	0.81
3	6.7300	14.8166	6.7300	14.8166	0.00
4	6.9879	14.9829	6.7837	14.9440	2.92
5	7.2684	14.8568	7.2684	14.8568	0.00
6	6.4437	14.9999	6.3817	14.9933	0.96
7	7.0472	14.9261	7.0472	14.9261	0.00
8	7.0047	14.9225	7.0047	14.9225	0.00
9	7.0526	14.6940	6.6520	14.0760	5.68
10	7.1569	14.5328	7.1569	14.5328	0.00
Avg.					1.04

Table 3.7: 2-opt heuristic solutions for the problem of maximizing algebraic connectivity with power consumption constraint. Corresponding to every n, the value of λ_2^{2opt} and the percent gap is averaged over ten random instances.

n	P_{max}	Average λ_2^{2opt}	Average % gap
8	20	9.2964	7.04
9	20	9.6658	3.34
10	20	9.8182	1.82
12	25	12.3150	1.48
15	30	14.9268	0.49
20	50	24.8734	0.51
25	100	49.9549	0.10

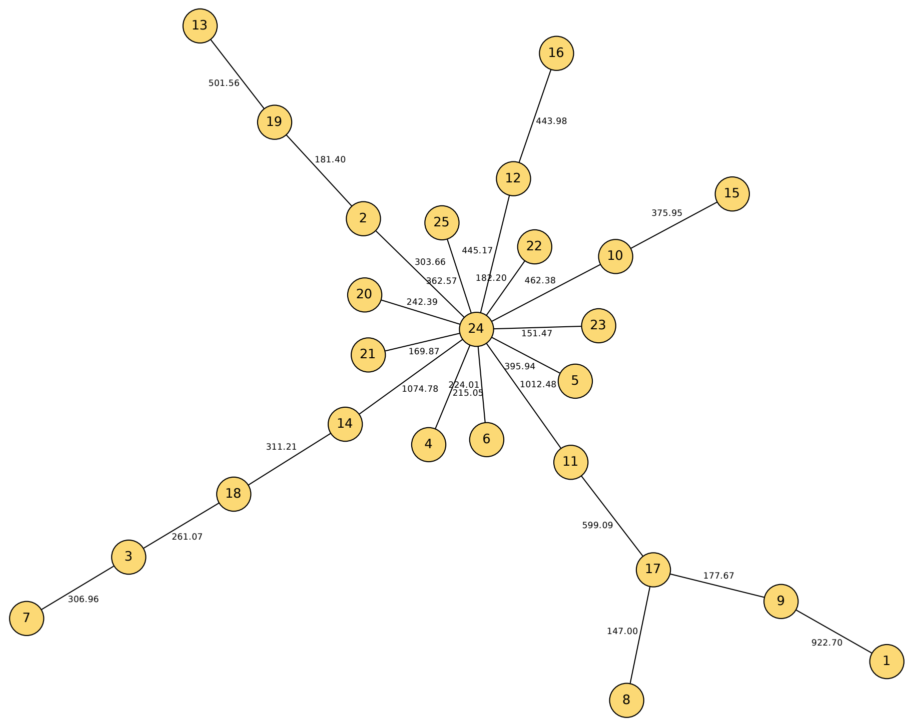


Figure 3.9: 2-opt heuristic solution for a problem with 25 nodes and random edge weights. For the shown network, $\lambda_2^{2opt} = 49.9379$ (percent gap = 0.12) and $P_{max} = 100$. This figure is just a representation of the connectivity of the network and does not necessarily represent the location of nodes.

4. CONCLUSIONS

In this dissertation, we aimed at understanding the relevance of a simplified version of an open problem in system realization theory which has several important applications in disparate fields of engineering. The basic problem in the context of mechanical systems we considered was as follows: Given a collection of masses and a set of linear springs with a specified cost and stiffness, a resource constraint in terms of a budget on the total cost, the problem was to determine an optimal connection of masses and springs so that the resulting structure was as stiff as possible. Under certain assumptions, we showed that the the structure is stiff when the second non-zero natural frequency of the interconnection is maximized.

We also aimed at understanding the relevance of the variants of this problem in deploying UAVs for civilian and military applications. In particular, we were interested in synthesizing a communication network among the UAVs subject to resource and performance constraints. Some of the important resource constraints considered were: limit on the maximum number of communication links, power consumed and maximum latency in routing the information between any pair of UAVs in the network. As a performance objective, we considered algebraic connectivity (second non-zero eigenvalue of the network's Laplacian) as the measure since it determines the convergence rate of consensus protocols and error attenuation in UAV formations.

The mechanical/UAV network synthesis problem, formulated as a Mixed Integer Semi-Definite Problem (MISDP), had scarce literature on the development of systematic procedures to solve this problem. To address this void in the literature, we developed *novel* algorithms to obtain optimal solutions and upper bounds for moderate sized problems and fast heuristic algorithms to obtain good sub-optimal solutions for larger problems.

We posed the problem of maximizing algebraic connectivity as three equivalent formulations: MISDP formulation, MISDP formulation with connectivity constraints and Fiedler vector formulation as MILP. We observed that the binary relaxation of the MISDP formulation did not necessarily satisfy the cutset constraints and hence invoked the multicommodity flow formulation to ensure that the connectivity requirements were satisfied. We also posed this problem equivalently as a MILP using the Fiedler vectors of the feasible solutions since there are not many efficient MISDP solvers available. We observed that the binary relaxations of these formulations were very weak (up to 128 percent deviation from the optimal solution for eight nodes problem). However, owing to the various useful features of the three equivalent formulations, we developed effective methods for obtaining upper bounds and optimal solutions.

Relaxing the feasible set by outer approximating the semi-definite constraint in the MISDP formulation with a finite number of Fiedler vectors would naturally lead to an upper bound on the maximum algebraic connectivity. Based on this idea, we proposed a procedure to effectively enumerate the Fiedler vectors of the feasible solutions to obtain tight upper bounds. We observed that the upper bound was tighter when the semi-definite constraint was relaxed with Fiedler vectors of solutions with higher values of algebraic connectivity. With thousand such Fiedler vectors used for relaxation, the average percent deviation of the upper bound from the optimal solution was within 4.13% (best deviation = 0%) for the eight nodes problem and was within 42.9% (best deviation = 25.6%) for nine nodes problem. For the problem with ten and twelve nodes, the average percent deviation of the upper bound from the best known feasible solution was within 65.7% (best deviation = 37.7%) and 116.1% (best deviation = 92.1%) respectively. However, the main drawback of this procedure is the enumeration of good feasible solutions. If the construction of feasible solutions is non-trivial, then this procedure would not be effective to obtain tight upper bounds.

We also proposed three cutting plane algorithms to solve the proposed MISDP to optimality. Firstly, algorithm EA_1 was based on the construction of successively tighter polyhedral approximations of the positive semi-definite set. Secondly, iterative primal-dual algorithm EA_2 considered the Lagrangian relaxation of the semi-definite constraint where the primal feasible solution was updated iteratively with a better solution obtained by solving the related dual problem. Thirdly, algorithm EA_3 was based on the Binary Semi-Definite Program (BSDP) approach in conjunction with cutting plane and bisection techniques. Computationally, we observed that the proposed algorithms implemented in CPLEX performed much better than the available MISDP solvers in Matlab. In particular, though the performance of EA_1 and EA_2 were comparable, an improved relaxation of the semi-definite constraint in EA_1 with good Fiedler vectors tremendously reduced the computation time to obtain optimal solutions. Computationally, EA_1 with an improved relaxation of the feasible set performed at least eight times better than the standard EA_1 and EA_2 and at least two times better than EA_3. However, without an a priori knowledge of good Fiedler vectors to relax the feasible set, EA_3 performed computationally better than EA_1 and EA_2 for problems up to nine nodes. Another useful feature of EA_3 was the continually improving lower bound with a corresponding feasible solution at every bisection step. This was very useful to obtain quick feasible solutions with a good lower bound for the problem of maximizing algebraic connectivity under power consumption constraint.

We also developed quick improvement heuristics for the problem of maximizing algebraic connectivity based on neighborhood search methods. In particular, we extended the idea of the well known k-opt search which has been successfully implemented for traveling salesman problems. k-opt search aims to iteratively search for better solutions by performing an exchange of edges in each iteration. The standard 2-opt (two edges exchanged in every iteration) search performed very well and provided optimal solutions for problems with up to nine nodes. However, for larger problems ($n \geq 15$), owing to the exponential rise in the number of edge deletion and addition combinations, standard 2-opt was very slow. Hence, we proposed an improved k-opt search where the search space was significantly but effectively reduced based on the variational characterization of eigenvalues. Computational results suggested that the improved 3-opt search performed the best while the improved 2-opt search provided a good trade-off between finding good solutions and the required computation time.

Finally, we proposed algorithms to address the variants of **BP** subject to resource constraints such as, the diameter constraint and the power consumption constraint. We posed the problem of maximizing algebraic connectivity of a network as a MISDP and the diameter of the graph was formulated using a multicommodity flow formulation. We provided computational results for problems involving seven and eight nodes under varying limits on the diameter of the graph. Even though the proposed algorithm was an improvement over state-of-the-art MISDP solvers, there is definitely a need for faster algorithms that can handle more number of vertices.

We posed the problem of maximizing algebraic connectivity of a network as a MISDP and mathematically formulated the power consumption constraint by relating it to the second and third eigenvalues of the networks's Laplacian. We proposed an algorithm to obtain optimal solutions based on cutting plane method. Though this algorithm was an improvement over the existing MISDP solvers, it could handle only smaller instances (up to seven nodes) without much guarantee on the run time.

We employed the BSDP approach in conjunction with the bisection technique to obtain quick lower bounds from the associated feasible solutions. Terminating the algorithm in three minutes, the average percent deviation of the lower bound from the optimal solution for seven nodes problem was 3.5%. For the problem with ten nodes, the lower bound obtained was within 15.2% from the upper bound (a simple bound on the optimal algebraic connectivity stemming from the power consumption constraint). Lastly, we applied 2-opt heuristic to find good feasible solutions. For the seven nodes problem, the average percent deviation of the 2-opt solution from the optimal solution was within 1.04%. For larger problem sizes (up to 25 nodes), 2-opt heuristic performed very well with respect to the values of P_{max} chosen.

REFERENCES

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

[2] C. Bob. Rehabilitation of existing structures in seismic zones. *Progress in Structural Engineering and Materials*, 3(4):353–359, 2001.

[3] H. Lehpamer. *RFID Design Principles*. Artech House Publishers, 2012.

[4] W. C. Jakes and D. C. Cox. *Microwave Mobile Communications*. Wiley-IEEE Press, 1994.

[5] S. K. Yadlapalli, S. Darbha, and K. R. Rajagopal. Information flow and its relation to stability of the motion of vehicles in a rigid formation. *IEEE Transactions on Automatic Control*, 51(8):1315–1319, 2006.

[6] H. Wang. *Robustness of networks*. PhD thesis, Delft University of Technology, 2009.

[7] Y. Kim and M. Mesbahi. On maximizing the second smallest eigenvalue of a state-dependent graph Laplacian. *IEEE Transactions on Automatic Control*, 51(1):116–120, 2006.

[8] S. Martinez, J. Cortes, and F. Bullo. Motion coordination with distributed information. *IEEE Control Systems Magazine*, 27(4):75–88, 2007.

[9] M. M. Zavlanos and G. J. Pappas. Potential fields for maintaining connectivity of mobile networks. *IEEE Transactions on Robotics*, 23(4):812–816, 2007.

[10] W. A. Malik, S. Rajagopal, S. Darbha, and K. R. Rajagopal. Maximizing the algebraic connectivity of a graph subject to a constraint on the maximum number of edges. *Advances in Dynamics and Control: Theory, Methods and Applications, Cambridge Scientific Publishers Ltd*, pages 135–148, 2009.

[11] J. Cong, L. He, C. Koh, and P. H. Madden. Performance optimization of VLSI interconnect layout. *Integration, the VLSI journal*, 21:1–94, November 1996.

[12] H. Zhou F. Y. Young H. H. Yang H. M. Chen, D. F. Wong and N. Sherwani. Integrated Floorplanning and Interconnect Planning. *Layout Optimization in VLSI Designs*, 2001.

[13] M. S. Nakhla and R. Achar. *Interconnect Modeling and Simulation, The VLSI Handbook, Edited by W. Chen.* CRC Press, 2000.

[14] L. R. Varshney. Distributed inference networks with costly wires. In *American Control Conference, 2010*, pages 1053–1058. IEEE, 2010.

[15] K. Heath, N. Gelfand, M. Ovsjanikov, M. Aanjaneya, and L. J. Guibas. Image webs: Computing and exploiting connectivity in image collections. 2010.

[16] P. Wei and S. Dengfeng. Weighted Algebraic Connectivity: An Application to Airport Transportation Network. In *International Federation of Automatic Control World Congress*, 2011.

[17] H Nagarajan, P Wei, S Rathinam, and D Sun. Air transportation network robustness optimization under limited legs itinerary constraint. In *International Conference on Research in Air Transportation.* National Aeronautics and Space Administration, 2012.

[18] I. K. Son. *Design and Optimization of Free Space Optical Networks*. PhD thesis, Auburn University, 2010.

[19] W. Liu, H. Sirisena, K. Pawlikowski, and A. McInnes. Utility of algebraic connectivity metric in topology design of survivable networks. In *International Workshop on Design of Reliable Communication Networks*, pages 131–138. IEEE, 2009.

[20] R. Kalman. The evolution of system theory: My memories and hopes. Plenary Speech. In *International Federation of Automatic Control World Congress, Prague*. IFAC, 2005.

[21] S. Seshu and M. B. Reed. *Linear Graphs and Electrical Networks*. Addison-Wesley Pub. Co, 1961.

[22] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(2):298–305, 1973.

[23] R. Merris. A survey of graph Laplacians. *Linear and Multilinear Algebra*, 39(1):19–31, 1995.

[24] B. Mohar. The Laplacian spectrum of graphs. *Graph theory, combinatorics, and applications*, 2:871–898, 1991.

[25] C. Maas. Transportation in graphs and the admittance spectrum. *Discrete Applied Mathematics*, 16(1):31–49, 1987.

[26] A. Ghosh and S. Boyd. Upper bounds on algebraic connectivity via convex optimization. *Linear algebra and its applications*, 418(2-3):693–707, 2006.

[27] Y. Kim. Bisection algorithm of increasing algebraic connectivity by adding an edge. *IEEE Transactions on Automatic Control*, 55(1):170 –174, 2010.

[28] A. Atamtürk and V. Narayanan. Lifting for conic mixed-integer programming. *Mathematical programming*, 126(2):351–363, 2011.

[29] A. Atamtürk and V. Narayanan. Conic mixed-integer rounding cuts. *Mathematical Programming*, 122(1):1–20, 2010.

[30] K. Krishnan. *Linear programming approaches to semidefinite programming problems*. PhD thesis, Rensselaer Polytechnic Institute, 2002.

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

[32] T. L. Magnanti and L. A. Wolsey. Optimal trees. *Handbooks in operations research and management science*, 7:503–615, 1995.

[33] J. F. Sturm. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. *Optimization methods and software*, 11(1):625–653, 1999.

[34] H. N. Gabow, Z. Galil., T. Spencer, and R. E. Tarjan. Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. *Combinatorica*, 6(2):109–122, 1986.

[35] J. Lofberg. Yalmip: A toolbox for modeling and optimization in matlab. In *IEEE International Symposium on Computer Aided Control Systems Design*, pages 284–289. IEEE, 2004.

[36] R. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 64(5):275–278, 1958.

[37] R. Gomory. An algorithm for the mixed integer problem. Technical report, DTIC Document, The Rand Corporation, 1960.

[38] S. Ceria, G. Cornuéjols, and M. Dawande. Combining and strengthening gomory cuts. In *Integer Programming and Combinatorial Optimization*, pages 438–451. Springer, 1995.

[39] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. The MIT press, 2001.

[40] E. L. Lawler. *Combinatorial optimization: networks and matroids*. Dover Publications, 2001.

[41] MATLAB. *version 7.11.0 (R2010b)*. The MathWorks Inc., Natick, Massachusetts, 2010.

[42] M. Gendreau, A. Hertz, and G. Laporte. A tabu search heuristic for the vehicle routing problem. *Management science*, 40(10):1276–1290, 1994.

[43] J. Potvin. Genetic algorithms for the traveling salesman problem. *Annals of Operations Research*, 63(3):337–370, 1996.

[44] G. A. Croes. A method for solving traveling-salesman problems. *Operations Research*, 6(6):791–812, 1958.

[45] L. Gouveia and T. L. Magnanti. Network flow models for designing diameter-constrained minimum-spanning and steiner trees. *Networks*, 41(3):159–173, 2003.

[46] A. S. Ibrahim, K. G. Seddik, and K. J. Ray Liu. Improving connectivity via relays deployment in wireless sensor networks. In *GLOBECOM*, pages 1159–1163, 2007.

[47] E. W. Frew and T. X. Brown. Networking issues for small unmanned aircraft systems. *Journal of Intelligent and Robotic Systems*, 54:21–37, March 2009.

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

[49] O. Burdakov, P. Doherty, K. Holmberg, J. Kvarnstrom, and P. R. Olsson. Positioning unmanned aerial vehicles as communication relays for surveillance tasks. In *Proceedings of Robotics: Science and Systems*, Seattle, USA, June 2009.

[50] P. Zhan, K. Yu, and A. L. Swindlehurst. Wireless relay communications with unmannned aerial vehicles: performance and optimization. *IEEE Transactions on Aerospace and Electronic Systems*, 2010.

[51] M. S. Asher, S. J. Stafford, R. J. Bamberger, A. Q. Rogers, D. Scheidt, and R. Chalmers. Radionavigation alternatives for US army ground forces in GPS denied environments. In *International Technical Meeting of The Institute of Navigation*, pages 508–532, 2001.

[52] D. L. Gu, G. Pei, H. Ly, M. Gerla, B. Zhang, and X. Hong. UAV aided intelligent routing for ad-hoc wireless network in single-area theater. In *IEEE Wireless Communications and Networking Conference*, 2000.

[53] A. Srinivas, G. Zussman, and E. Modiano. Construction and maintenance of wireless mobile backbone networks. *IEEE/ACM Transactions on Networking*, 17(1):239–252, 2009.

[54] E. M. Craparo, J. P. How, and E. Modiano. Throughput optimization in mobile backbone networks. *IEEE Transactions on Mobile Computing*, 10(4):560–572, 2011.

APPENDIX A

APPENDIX

All the computational results in section 2.3 on algorithms for computing optimal solutions are based on the weighted adjacency matrices shown below.

Random weighted adjacency matrices for *eight* nodes problem

$$A_1 = \begin{pmatrix} 0 & 4.561 & 19.020 & 37.537 & 82.393 & 18.295 & 50.073 & 5.511 \\ 4.561 & 0 & 50.358 & 2.819 & 5.916 & 34.933 & 43.855 & 44.377 \\ 19.020 & 50.358 & 0 & 16.268 & 11.806 & 2.159 & 45.568 & 77.271 \\ 37.537 & 2.819 & 16.268 & 0 & 28.642 & 45.083 & 62.932 & 24.352 \\ 82.393 & 5.916 & 11.806 & 28.642 & 0 & 2.590 & 23.840 & 13.704 \\ 18.295 & 34.933 & 2.159 & 45.083 & 2.590 & 0 & 4.041 & 35.791 \\ 50.073 & 43.855 & 45.568 & 62.932 & 23.840 & 4.041 & 0 & 55.830 \\ 5.511 & 44.377 & 77.271 & 24.352 & 13.704 & 35.791 & 55.830 & 0 \end{pmatrix}$$

$$A_2 = \begin{pmatrix} 0 & 7.991 & 19.023 & 40.147 & 46.093 & 9.834 & 48.182 & 39.823 \\ 7.991 & 0 & 82.412 & 17.293 & 26.714 & 31.590 & 36.865 & 22.808 \\ 19.023 & 82.412 & 0 & 34.046 & 22.715 & 18.902 & 50.309 & 14.671 \\ 40.147 & 17.293 & 34.046 & 0 & 25.462 & 10.701 & 51.117 & 34.138 \\ 46.093 & 26.714 & 22.715 & 25.462 & 0 & 38.596 & 53.231 & 16.664 \\ 9.834 & 31.590 & 18.902 & 10.701 & 38.596 & 0 & 13.779 & 58.921 \\ 48.182 & 36.865 & 50.309 & 51.117 & 53.231 & 13.779 & 0 & 53.351 \\ 39.823 & 22.808 & 14.671 & 34.138 & 16.664 & 58.921 & 53.351 & 0 \end{pmatrix}$$

$$A_3 = \begin{pmatrix} 0 & 5.449 & 13.087 & 39.460 & 14.189 & 26.056 & 30.279 & 41.788 \\ 5.449 & 0 & 23.490 & 18.772 & 24.992 & 43.876 & 14.074 & 66.580 \\ 13.087 & 23.490 & 0 & 13.379 & 44.093 & 11.845 & 45.530 & 65.366 \\ 39.460 & 18.772 & 13.379 & 0 & 28.403 & 54.327 & 68.801 & 30.908 \\ 14.189 & 24.992 & 44.093 & 28.403 & 0 & 31.147 & 62.558 & 8.237 \\ 26.056 & 43.876 & 11.845 & 54.327 & 31.147 & 0 & 21.427 & 78.777 \\ 30.279 & 14.074 & 45.530 & 68.801 & 62.558 & 21.427 & 0 & 61.276 \\ 41.788 & 66.580 & 65.366 & 30.908 & 8.237 & 78.777 & 61.276 & 0 \end{pmatrix}$$

$$A_4 = \begin{pmatrix} 0 & 3.166 & 10.819 & 69.610 & 7.771 & 35.867 & 47.759 & 11.385 \\ 3.166 & 0 & 23.452 & 26.608 & 13.743 & 63.817 & 56.875 & 12.734 \\ 10.819 & 23.452 & 0 & 16.165 & 30.174 & 46.717 & 41.704 & 66.899 \\ 69.610 & 26.608 & 16.165 & 0 & 5.841 & 57.495 & 67.210 & 14.102 \\ 7.771 & 13.743 & 30.174 & 5.841 & 0 & 63.502 & 61.732 & 23.618 \\ 35.867 & 63.817 & 46.717 & 57.495 & 63.502 & 0 & 11.427 & 38.997 \\ 47.759 & 56.875 & 41.704 & 67.210 & 61.732 & 11.427 & 0 & 98.913 \\ 11.385 & 12.734 & 66.899 & 14.102 & 23.618 & 38.997 & 98.913 & 0 \end{pmatrix}$$

$$A_5 = \begin{pmatrix} 0 & 2.544 & 18.566 & 23.983 & 44.333 & 11.513 & 47.634 & 8.196 \\ 2.544 & 0 & 17.548 & 20.902 & 29.848 & 56.828 & 16.094 & 45.784 \\ 18.566 & 17.548 & 0 & 20.030 & 21.883 & 21.306 & 19.583 & 13.961 \\ 23.983 & 20.902 & 20.030 & 0 & 33.448 & 50.940 & 7.763 & 22.462 \\ 44.333 & 29.848 & 21.883 & 33.448 & 0 & 60.604 & 57.279 & 7.599 \\ 11.513 & 56.828 & 21.306 & 50.940 & 60.604 & 0 & 19.492 & 7.163 \\ 47.634 & 16.094 & 19.583 & 7.763 & 57.279 & 19.492 & 0 & 98.613 \\ 8.196 & 45.784 & 13.961 & 22.462 & 7.599 & 7.163 & 98.613 & 0 \end{pmatrix}$$

$$A_6 = \begin{pmatrix} 0 & 3.368 & 5.354 & 64.684 & 66.925 & 28.203 & 41.094 & 53.284 \\ 3.368 & 0 & 34.119 & 8.390 & 27.285 & 35.904 & 11.076 & 51.050 \\ 5.354 & 34.119 & 0 & 33.155 & 33.273 & 28.636 & 34.563 & 59.182 \\ 64.684 & 8.390 & 33.155 & 0 & 28.884 & 20.305 & 43.513 & 15.110 \\ 66.925 & 27.285 & 33.273 & 28.884 & 0 & 62.458 & 34.925 & 3.265 \\ 28.203 & 35.904 & 28.636 & 20.305 & 62.458 & 0 & 4.674 & 27.095 \\ 41.094 & 11.076 & 34.563 & 43.513 & 34.925 & 4.674 & 0 & 45.437 \\ 53.284 & 51.050 & 59.182 & 15.110 & 3.265 & 27.095 & 45.437 & 0 \end{pmatrix}$$

$$A_7 = \begin{pmatrix} 0 & 5.721 & 8.828 & 22.020 & 55.966 & 5.384 & 34.178 & 43.546 \\ 5.721 & 0 & 17.823 & 18.462 & 31.074 & 26.090 & 18.068 & 28.879 \\ 8.828 & 17.823 & 0 & 23.527 & 25.014 & 48.801 & 40.533 & 53.078 \\ 22.020 & 18.462 & 23.527 & 0 & 37.835 & 38.275 & 4.024 & 19.766 \\ 55.966 & 31.074 & 25.014 & 37.835 & 0 & 50.395 & 50.884 & 11.786 \\ 5.384 & 26.090 & 48.801 & 38.275 & 50.395 & 0 & 12.491 & 35.477 \\ 34.178 & 18.068 & 40.533 & 4.024 & 50.884 & 12.491 & 0 & 71.750 \\ 43.546 & 28.879 & 53.078 & 19.766 & 11.786 & 35.477 & 71.750 & 0 \end{pmatrix}$$

$$A_8 = \begin{pmatrix} 0 & 1.537 & 12.505 & 45.077 & 68.271 & 6.608 & 20.672 & 37.893 \\ 1.537 & 0 & 76.166 & 11.996 & 10.903 & 25.450 & 57.973 & 36.482 \\ 12.505 & 76.166 & 0 & 37.794 & 22.848 & 20.843 & 15.406 & 39.688 \\ 45.077 & 11.996 & 37.794 & 0 & 37.311 & 29.056 & 36.097 & 27.623 \\ 68.271 & 10.903 & 22.848 & 37.311 & 0 & 63.989 & 59.293 & 4.220 \\ 6.608 & 25.450 & 20.843 & 29.056 & 63.989 & 0 & 12.757 & 33.223 \\ 20.672 & 57.973 & 15.406 & 36.097 & 59.293 & 12.757 & 0 & 105.431 \\ 37.893 & 36.482 & 39.688 & 27.623 & 4.220 & 33.223 & 105.431 & 0 \end{pmatrix}$$

$$A_9 = \begin{pmatrix} 0 & 7.473 & 13.871 & 74.945 & 59.785 & 28.499 & 36.559 & 41.392 \\ 7.473 & 0 & 63.104 & 1.118 & 18.255 & 56.460 & 30.670 & 28.415 \\ 13.871 & 63.104 & 0 & 21.090 & 12.332 & 26.304 & 31.328 & 38.784 \\ 74.945 & 1.118 & 21.090 & 0 & 34.870 & 35.743 & 13.807 & 6.835 \\ 59.785 & 18.255 & 12.332 & 34.870 & 0 & 74.240 & 78.291 & 8.182 \\ 28.499 & 56.460 & 26.304 & 35.743 & 74.240 & 0 & 13.607 & 60.731 \\ 36.559 & 30.670 & 31.328 & 13.807 & 78.291 & 13.607 & 0 & 100.509 \\ 41.392 & 28.415 & 38.784 & 6.835 & 8.182 & 60.731 & 100.509 & 0 \end{pmatrix}$$

$$A_{10} = \begin{pmatrix} 0 & 4.673 & 11.233 & 47.921 & 20.123 & 5.275 & 11.570 & 41.965 \\ 4.673 & 0 & 59.460 & 26.490 & 24.895 & 48.453 & 49.937 & 45.337 \\ 11.233 & 59.460 & 0 & 20.843 & 21.083 & 33.312 & 3.120 & 56.785 \\ 47.921 & 26.490 & 20.843 & 0 & 23.790 & 14.368 & 57.961 & 26.491 \\ 20.123 & 24.895 & 21.083 & 23.790 & 0 & 63.058 & 84.360 & 10.774 \\ 5.275 & 48.453 & 33.312 & 14.368 & 63.058 & 0 & 6.137 & 37.142 \\ 11.570 & 49.937 & 3.120 & 57.961 & 84.360 & 6.137 & 0 & 82.681 \\ 41.965 & 45.337 & 56.785 & 26.491 & 10.774 & 37.142 & 82.681 & 0 \end{pmatrix}$$

Random weighted adjacency matrices for *nine* nodes problem

$$A_1 = \begin{pmatrix} 0 & 51.109 & 103.141 & 74.350 & 3.664 & 13.229 & 15.797 & 18.230 & 30.797 \\ 51.109 & 0 & 79.543 & 7.805 & 19.555 & 18.661 & 25.386 & 54.808 & 67.820 \\ 103.141 & 79.543 & 0 & 25.047 & 4.786 & 38.796 & 46.383 & 6.685 & 88.554 \\ 74.350 & 7.805 & 25.047 & 0 & 28.353 & 23.511 & 55.800 & 46.123 & 91.246 \\ 3.664 & 19.555 & 4.786 & 28.353 & 0 & 39.345 & 74.242 & 116.722 & 68.593 \\ 13.229 & 18.661 & 38.796 & 23.511 & 39.345 & 0 & 61.739 & 65.714 & 3.377 \\ 15.797 & 25.386 & 46.383 & 55.800 & 74.242 & 61.739 & 0 & 6.930 & 25.114 \\ 18.230 & 54.808 & 6.685 & 46.123 & 116.722 & 65.714 & 6.930 & 0 & 30.790 \\ 30.797 & 67.820 & 88.554 & 91.246 & 68.593 & 3.377 & 25.114 & 30.790 & 0 \end{pmatrix}$$

$$A_2 = \begin{pmatrix} 0 & 55.451 & 35.171 & 84.885 & 5.505 & 20.855 & 29.453 & 30.388 & 68.093 \\ 55.451 & 0 & 66.881 & 7.059 & 15.901 & 21.996 & 19.397 & 65.193 & 64.995 \\ 35.171 & 66.881 & 0 & 11.186 & 20.066 & 14.621 & 61.816 & 69.104 & 45.769 \\ 84.885 & 7.059 & 11.186 & 0 & 52.853 & 46.755 & 65.175 & 47.878 & 72.586 \\ 5.505 & 15.901 & 20.066 & 52.853 & 0 & 14.783 & 39.858 & 15.650 & 76.328 \\ 20.855 & 21.996 & 14.621 & 46.755 & 14.783 & 0 & 63.148 & 55.653 & 6.730 \\ 29.453 & 19.397 & 61.816 & 65.175 & 39.858 & 63.148 & 0 & 3.325 & 11.846 \\ 30.388 & 65.193 & 69.104 & 47.878 & 15.650 & 55.653 & 3.325 & 0 & 30.335 \\ 68.093 & 64.995 & 45.769 & 72.586 & 76.328 & 6.730 & 11.846 & 30.335 & 0 \end{pmatrix}$$

$$A_3 = \begin{pmatrix} 0 & 34.962 & 106.416 & 83.430 & 3.962 & 19.667 & 16.809 & 40.972 & 23.189 \\ 34.962 & 0 & 62.136 & 6.460 & 21.299 & 20.235 & 55.212 & 32.185 & 65.989 \\ 106.416 & 62.136 & 0 & 3.817 & 19.672 & 35.388 & 38.132 & 65.466 & 31.264 \\ 83.430 & 6.460 & 3.817 & 0 & 23.553 & 67.925 & 56.553 & 49.485 & 72.971 \\ 3.962 & 21.299 & 19.672 & 23.553 & 0 & 64.328 & 41.192 & 98.448 & 78.225 \\ 19.667 & 20.235 & 35.388 & 67.925 & 64.328 & 0 & 64.970 & 78.577 & 6.603 \\ 16.809 & 55.212 & 38.132 & 56.553 & 41.192 & 64.970 & 0 & 5.339 & 19.096 \\ 40.972 & 32.185 & 65.466 & 49.485 & 98.448 & 78.577 & 5.339 & 0 & 16.332 \\ 23.189 & 65.989 & 31.264 & 72.971 & 78.225 & 6.603 & 19.096 & 16.332 & 0 \end{pmatrix}$$

$$A_4 = \begin{pmatrix} 0 & 37.880 & 67.875 & 100.379 & 6.067 & 8.170 & 32.228 & 30.653 & 34.148 \\ 37.880 & 0 & 80.509 & 5.478 & 21.916 & 35.606 & 22.303 & 51.762 & 83.597 \\ 67.875 & 80.509 & 0 & 9.380 & 22.829 & 41.847 & 43.423 & 38.497 & 80.200 \\ 100.379 & 5.478 & 9.380 & 0 & 39.792 & 40.394 & 34.656 & 47.301 & 56.581 \\ 6.067 & 21.916 & 22.829 & 39.792 & 0 & 82.544 & 76.565 & 82.045 & 15.671 \\ 8.170 & 35.606 & 41.847 & 40.394 & 82.544 & 0 & 60.790 & 137.604 & 5.053 \\ 32.228 & 22.303 & 43.423 & 34.656 & 76.565 & 60.790 & 0 & 4.473 & 16.443 \\ 30.653 & 51.762 & 38.497 & 47.301 & 82.045 & 137.604 & 4.473 & 0 & 23.297 \\ 34.148 & 83.597 & 80.200 & 56.581 & 15.671 & 5.053 & 16.443 & 23.297 & 0 \end{pmatrix}$$

$$A_5 = \begin{pmatrix} 0 & 74.434 & 54.004 & 49.180 & 3.425 & 23.574 & 14.026 & 46.679 & 27.705 \\ 74.434 & 0 & 72.725 & 12.127 & 5.963 & 32.650 & 47.091 & 6.417 & 5.720 \\ 54.004 & 72.725 & 0 & 6.283 & 23.399 & 34.824 & 60.464 & 43.262 & 73.479 \\ 49.180 & 12.127 & 6.283 & 0 & 34.042 & 36.230 & 30.105 & 61.880 & 70.808 \\ 3.425 & 5.963 & 23.399 & 34.042 & 0 & 41.705 & 37.664 & 71.445 & 28.397 \\ 23.574 & 32.650 & 34.824 & 36.230 & 41.705 & 0 & 49.857 & 64.825 & 4.676 \\ 14.026 & 47.091 & 60.464 & 30.105 & 37.664 & 49.857 & 0 & 6.500 & 13.624 \\ 46.679 & 6.417 & 43.262 & 61.880 & 71.445 & 64.825 & 6.500 & 0 & 17.935 \\ 27.705 & 5.720 & 73.479 & 70.808 & 28.397 & 4.676 & 13.624 & 17.935 & 0 \end{pmatrix}$$

$$A_6 = \begin{pmatrix} 0 & 64.068 & 10.484 & 82.702 & 5.059 & 17.211 & 41.722 & 51.143 & 34.027 \\ 64.068 & 0 & 38.358 & 13.136 & 19.432 & 8.179 & 36.737 & 43.368 & 44.477 \\ 10.484 & 38.358 & 0 & 4.736 & 19.992 & 35.610 & 68.747 & 66.199 & 100.487 \\ 82.702 & 13.136 & 4.736 & 0 & 26.705 & 69.996 & 24.366 & 62.367 & 68.319 \\ 5.059 & 19.432 & 19.992 & 26.705 & 0 & 6.220 & 42.855 & 100.982 & 54.818 \\ 17.211 & 8.179 & 35.610 & 69.996 & 6.220 & 0 & 71.220 & 79.242 & 6.379 \\ 41.722 & 36.737 & 68.747 & 24.366 & 42.855 & 71.220 & 0 & 4.100 & 13.469 \\ 51.143 & 43.368 & 66.199 & 62.367 & 100.982 & 79.242 & 4.100 & 0 & 20.063 \\ 34.027 & 44.477 & 100.487 & 68.319 & 54.818 & 6.379 & 13.469 & 20.063 & 0 \end{pmatrix}$$

$$A_7 = \begin{pmatrix} 0 & 84.178 & 40 & 94.496 & 3.252 & 19.661 & 19.108 & 56.048 & 40.033 \\ 84.178 & 0 & 110.743 & 7.442 & 15.846 & 35.148 & 24.472 & 52.636 & 21.187 \\ 40 & 110.743 & 0 & 19.213 & 19.871 & 9.568 & 67.812 & 51.830 & 56.333 \\ 94.496 & 7.442 & 19.213 & 0 & 25.484 & 26.072 & 51.210 & 33.379 & 75.134 \\ 3.252 & 15.846 & 19.871 & 25.484 & 0 & 84.510 & 28.471 & 115.426 & 110.035 \\ 19.661 & 35.148 & 9.568 & 26.072 & 84.510 & 0 & 34.251 & 47 & 6.161 \\ 19.108 & 24.472 & 67.812 & 51.210 & 28.471 & 34.251 & 0 & 3.757 & 9.776 \\ 56.048 & 52.636 & 51.830 & 33.379 & 115.426 & 47 & 3.757 & 0 & 31.904 \\ 40.033 & 21.187 & 56.333 & 75.134 & 110.035 & 6.161 & 9.776 & 31.904 & 0 \end{pmatrix}$$

$$A_8 = \begin{pmatrix} 0 & 46.103 & 105.745 & 61.239 & 5.627 & 18.101 & 24.459 & 47.970 & 58.582 \\ 46.103 & 0 & 26.179 & 6.670 & 23.523 & 30.729 & 48.579 & 61.829 & 49.850 \\ 105.745 & 26.179 & 0 & 7.958 & 26.819 & 26.925 & 39.978 & 58.829 & 59.187 \\ 61.239 & 6.670 & 7.958 & 0 & 16.321 & 65.826 & 27.566 & 56.328 & 93.999 \\ 5.627 & 23.523 & 26.819 & 16.321 & 0 & 19.572 & 19.077 & 26.750 & 87.654 \\ 18.101 & 30.729 & 26.925 & 65.826 & 19.572 & 0 & 61.369 & 93.219 & 4.440 \\ 24.459 & 48.579 & 39.978 & 27.566 & 19.077 & 61.369 & 0 & 5.917 & 14.002 \\ 47.970 & 61.829 & 58.829 & 56.328 & 26.750 & 93.219 & 5.917 & 0 & 27.248 \\ 58.582 & 49.850 & 59.187 & 93.999 & 87.654 & 4.440 & 14.002 & 27.248 & 0 \end{pmatrix}$$

$$A_9 = \begin{pmatrix} 0 & 49.524 & 47.001 & 97.199 & 5.978 & 14.685 & 32.794 & 36.858 & 34.073 \\ 49.524 & 0 & 92.926 & 2.890 & 25.966 & 23.370 & 19.856 & 72.462 & 61.856 \\ 47.001 & 92.926 & 0 & 20.361 & 10.725 & 26.954 & 27.427 & 41.613 & 39.825 \\ 97.199 & 2.890 & 20.361 & 0 & 21.730 & 43.284 & 94.839 & 100.156 & 61.767 \\ 5.978 & 25.966 & 10.725 & 21.730 & 0 & 52.430 & 78.240 & 76.468 & 111.859 \\ 14.685 & 23.370 & 26.954 & 43.284 & 52.430 & 0 & 129.134 & 70.418 & 1.984 \\ 32.794 & 19.856 & 27.427 & 94.839 & 78.240 & 129.134 & 0 & 2.300 & 16.639 \\ 36.858 & 72.462 & 41.613 & 100.156 & 76.468 & 70.418 & 2.300 & 0 & 31.418 \\ 34.073 & 61.856 & 39.825 & 61.767 & 111.859 & 1.984 & 16.639 & 31.418 & 0 \end{pmatrix}$$

$$A_{10} = \begin{pmatrix} 0 & 73.479 & 78.550 & 57.077 & 2.770 & 16.830 & 27.284 & 13.703 & 45.902 \\ 73.479 & 0 & 96.045 & 7.971 & 14.967 & 18.793 & 24.575 & 22.947 & 58.603 \\ 78.550 & 96.045 & 0 & 9.560 & 14.197 & 14.033 & 69.942 & 64.482 & 71.409 \\ 57.077 & 7.971 & 9.560 & 0 & 44.664 & 40.189 & 51.466 & 33.023 & 67.021 \\ 2.770 & 14.967 & 14.197 & 44.664 & 0 & 65.935 & 86.901 & 96.362 & 119.533 \\ 16.830 & 18.793 & 14.033 & 40.189 & 65.935 & 0 & 90.886 & 50.091 & 3.862 \\ 27.284 & 24.575 & 69.942 & 51.466 & 86.901 & 90.886 & 0 & 3.593 & 20.033 \\ 13.703 & 22.947 & 64.482 & 33.023 & 96.362 & 50.091 & 3.593 & 0 & 14.719 \\ 45.902 & 58.603 & 71.409 & 67.021 & 119.533 & 3.862 & 20.033 & 14.719 & 0 \end{pmatrix}$$