Heuristics for Synthesizing Robust Networks with a Diameter Constraint

Harsha Nagarajan, Peng Wei, Sivakumar Rathinam and Dengfeng Sun

Abstract

Robustness of a network in the presence of node or link failures plays an important role in the design of the network. A key factor that quantifies this robustness is the algebraic connectivity of the network. In this article, the authors address the problem of finding a network that maximizes the algebraic connectivity of the network while ensuring that the length of the shortest path joining any two nodes in the network is within a given bound. This article presents $k-$opt and tabu search heuristics for finding feasible solutions for this network synthesis problem. Computational results are also presented to corroborate the performance of the proposed algorithms.

I. Introduction

This article addresses the problem of finding a network that maximizes the algebraic connectivity of the network while ensuring that the length of the shortest path joining any two nodes in the network is within a given bound. This problem is a simpler version of a system realization problem that arises in several applications including biomedicine, mechanical systems, as well as the design of Very Large Scale Integrated (VLSI) circuits [1]. We are mainly interested in addressing this problem as it arises in the coordination of Unmanned Aerial Vehicles (UAVs) and in the design of transportation systems.

Several UAV applications require a Mobile Adhoc NETwork (MANET) to be deployed to facilitate the exchange of information between the vehicles in the network. Each vehicle represents a node in the MANET and the communication links between the vehicles represent the edges in the MANET. The basic problem here is to synthesize a collection of communication links in the MANET so that all the vehicles can communicate with each other, the number of links on a communication path between any two vehicles is at most equal to a given number and the MANET is *well-connected*. Algebraic connectivity serves as a measure of wellconnectedness of a network and can be used to identify good network topologies.

The algebraic connectivity of a network is defined as the second, smallest eigenvalue of the Laplacian matrix associated with the network. In applications where a node or a link (an edge) can randomly fail, algebraic connectivity is specifically considered to be a superior measure of connectivity compared to some of the other well known measures such as the node or edge connectivity. Node (edge) connectivity specifies the minimum number of nodes (edges) that must removed for the network to be disconnected. The node and the edge connectivity of any spanning tree connecting all the nodes is equal to 1. Therefore, measures such as the node or edge connectivity cannot distinguish between a spanning tree that is a star versus a spanning tree that is a Hamiltonian path. However, star networks are considered to be more robust towards any random failures of nodes or edges than an Hamiltonian path. Algebraic connectivity serves as a better measure in these cases because the algebraic connectivity of a star network is much greater than the algebraic connectivity of a Hamiltonian path.

In addition to UAV applications, algebraic connectivity has also been used as a measure of wellconnectedness in the design of transportation networks. For example, an air transportation network consists of airports and flight routes between airport pairs [2]. Usually, an undirected graph is used to describe an air transportation network [3]–[6] where the set of nodes represents all the airports and the set of edges represents all the flight routes. The topology or the structure of this network plays a vital role in determining the average time spent by the flights in the air and at airports, the average number of flights

H. Nagarajan and S. Rathinam are with Department of Mechanical Engineering, Texas A & M University. P. Wei and D. Sun are with School of Aeronautics and Astronautics, Purdue University.

waiting in queues over all the airports and the average number of hops or legs in a passenger's itinerary. The network also needs to be robust towards unpredictable node or link failures that may happen due to airline budget cuts, weather hazards and economic policies. There are several studies [4], [7]–[9] that have aimed to relate the performance of an air transportation network in terms of delays and failures (as mentioned above) with the topology or the structure of the transportation network. The overall approach here is that if a specific topology of structure is better, then one can develop the required mathematical tools to design a transportation network with the desired topology. Specifically, the authors in [4], [7]–[9] have shown that one of the factors that characterizes the performance of a network is its algebraic connectivity. The authors in [4] have demonstrated that an air transportation network for the United States with a higher algebraic connectivity exhibited superior performance in terms of average flight times, delays, queue length and the distance traveled by the passengers. In order to restrict the number of legs or stops incurred by the passengers as they travel, it is important to synthesize a network with additional constraints that limit the number of edges present in the path joining any two nodes in the network.

In this article, we restrict our attention to networks that are spanning trees. We consider the basic mathematical problem of synthesizing a spanning tree that maximizes the algebraic connectivity of the spanning tree subject to a limit on the diameter of the tree. The diameter of a tree is defined as the number of edges present in the longest path connecting any two nodes in the tree. Henceforth, we will refer to this problem as the algebraic connectivity problem. It is well known that this problem is NP-Hard [10]. Finding an optimal solution to this problem seems to be very difficult even for a network with 8 or 9 nodes [1] [11]. For n nodes, the number of distinct spanning trees available to connect all the nodes is n^{n-2}; therefore, for 8 nodes, there are 262144 combinatorial possibilities. As finding an optimal solution seems to be very challenging, we focus on developing good heuristics in this article.

The algebraic connectivity problem has received scant attention in the literature. In [12], the authors consider an un-weighted version of the algebraic connectivity problem with no connectivity constraints. In [1], the authors explain the importance of this problem in the context of several applications and present an optimal algorithm to solve the problem with no connectivity constraints. In [13], Wei and Sun study a simpler variant of this optimization problem which aims to add k additional edges to strengthen the network. In [11], the authors solve the algebraic connectivity problem in the context of UAV applications. The following are the contributions of this article:

- We develop heuristics based on $k-$opt and tabu search methods to find good feasible solutions to the problem.
- We present computational results to corroborate the performance of the proposed algorithms. These results indicate that the 3-opt heuristic performed the best among the proposed heuristics while the 2-opt heuristic provided a good trade-off between finding good solutions and the required computation time.

The rest of the article is organized as follows. In section II, we formulate the problem. Local search heuristics like k-opt and tabu search are presented in sections III and IV respectively. In section V, we evaluate the performance of our algorithms via simulations. Section VI concludes this article.

II. Problem formulation

Consider an undirected graph $G := (V; E)$ with V denoting all the n nodes and E representing all the edges in the graph. Each edge in the graph is associated with a weight which acts as a proxy to the likeliness of the edge to fail [14]. An edge is assigned a larger weight if the edge is a stronger link and is not likely to fail; on the other hand, an edge is assigned a relatively smaller weight if the edge is more likely to fail. Let w_e represent the edge weight associated with the edge $e \in E$. For any edge $e \in E$, let x_e represent a binary variable that specifies if edge e is present in the chosen network or not; $x_e = 1$ if e is chosen and $x_e = 0$ otherwise. Let the incidence vector, $x \in \{0, 1\}^{|E|}$, denote the entire network with the component x_e corresponding to $e \in E$. Let c_i denote the i^{th} column of the identity matrix I_n of size $|V| = n$. If the nodes i and j are connected using edge e, let $L_e = w_e(c_i - c_j) \otimes (c_i - c_j)$ where

$\otimes$ represents the tensor product. Using this notation, the Laplacian [15], [12], [4] of the chosen network may be defined as follows:

$$L(x) = \sum_{e \in E} x_e L_e.$$

Let $\lambda_2(L(x))$ denote the algebraic connectivity of the chosen network. To simplify the problem, we restrict our search for optimal networks from the set of all the spanning trees. A spanning tree connects all the nodes in the network and contains exactly $|V| - 1$ edges. As stated by Fiedler in [15], a graph is connected if and only if the algebraic connectivity of the graph is greater than zero. Therefore, the algebraic connectivity of any spanning tree will be greater than zero and the algebraic connectivity of any disconnected graph will be equal to zero. Hence, the problem of choosing a spanning tree while maximizing its algebraic connectivity can be posed as follows:

$$\underset{x}{\text{maximize}} \;\; \lambda_2(L(x)),$$

subject to

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

$$x \in \{0,1\}^{|E|}, \tag{1b}$$

$$\delta_{ij}(x) \leq D \quad \forall i, j \in V, \tag{1c}$$

where $\delta_{ij}(x)$ denotes the number of edges in the shortest path joining any two nodes i and j in the network represented by x, and D is the limit on the diameter of the network. Using the results in [4], [12], [1], the above problem can be formulated as a mixed integer, semi-definite program with the diameter constraint as follows:

$$\gamma^* = \underset{x,\gamma}{\text{maximize}} \;\; \gamma,$$

subject to

$$L(x) \succeq \gamma \; (I_n - e_0 \otimes e_0), \tag{2a}$$

$$\sum_{e \in E} x_e = |V| - 1, \tag{2b}$$

$$x \in \{0,1\}^{|E|}, \tag{2c}$$

$$\delta_{ij}(x) \leq D \quad \forall i, j \in V, \tag{2d}$$

where e_0 is the normalized eigenvector corresponding to the first eigenvalue of x, and for any two square matrices A, B with the same dimensions, $A \succeq B$ if and only if $A - B$ is a positive semi-definite matrix.

III. k-OPT HEURISTIC

$k-$opt heuristics were initially proposed for solving routing problems in [16]. In this heuristic, a feasible spanning tree is first generated by choosing any star graph with one of the vertices having a degree equal to $|V| - 1$. Then, a new spanning tree in a local neighborhood of the current, feasible solution is chosen such that the new solution is feasible and has a larger algebraic connectivity. This procedure is iterated until a better spanning tree cannot be found.

Suppose $\mathcal{T}_1$ and $\mathcal{T}_2$ are two feasible solutions for the algebraic connectivity problem. Then, $\mathcal{T}_2$ is said to be in the $k-$exchange neighborhood of $\mathcal{T}_1$ if $\mathcal{T}_2$ can be obtained by replacing k edges in $\mathcal{T}_1$. In a k-opt exchange, we find a (new) feasible solution in the $k-$exchange neighborhood of the current solution and replace the current solution with this new solution if the algebraic connectivity of the new solution is

Algorithm 1 : k-opt exchange

Input: D (positive integer)
$\mathcal{T}_0 \leftarrow$ Initial feasible solution satisfying diameter constraints
$\lambda_0 \leftarrow \lambda_2(L(\mathcal{T}_0)))$
$E_{del} \leftarrow$ Subset of k-edge combinations considered for possible deletion as obtained by the edge ranking procedure in subsection III-A
for each edge combination in E_{del} **do**
  Delete the k edges present in the edge combination to obtain connected components $C_1, C_2, C_3, \ldots, C_{k+1}$
  $E_{add} \leftarrow$ Subset of k-edge combinations considered for possible addition as obtained by the edge ranking procedure in subsection III-B
  Let $\mathcal{T}_1$ 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 $\lambda_2(L(\mathcal{T}_1)) > \lambda_0$ **then**
    $\mathcal{T}_0 \leftarrow \mathcal{T}_1$
    $\lambda_0 \leftarrow \lambda_2(L(\mathcal{T}_1))$
  end if
end for
Output $\mathcal{T}_0$ as the (new) current solution

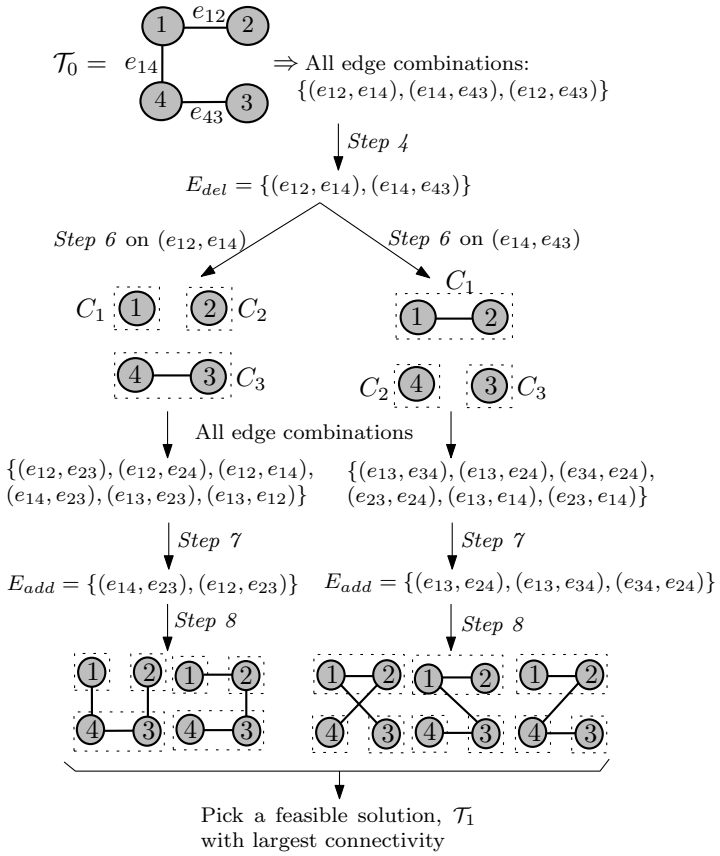


Fig. 1: An example illustrating a 2-opt exchange.

larger than the algebraic connectivity of the current solution. In summary, a $k-$opt heuristic starts with an initial feasible solution and iteratively applies a $k-$opt exchange until no improvements can be made.

There are two main steps in this $k-$opt exchange: 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. 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.

A. 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 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:

$$\lambda_2(L(\mathcal{T}_0 - e)) \leq v' L(\mathcal{T}_0 - e) v \quad \forall v \in \mathcal{V} \tag{3a}$$
$$= v' L(\mathcal{T}_0) v - v' L_e v \quad \forall v \in \mathcal{V} \tag{3b}$$
$$= v' L(\mathcal{T}_0) v - w_e (v_i - v_j)^2 \quad \forall v \in \mathcal{V} \tag{3c}$$

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 the simulations section.

B. 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:

$$\lambda_2(L(\tilde{\mathcal{T}}_0 + e)) \leq v' L(\tilde{\mathcal{T}}_0 + e) v \quad \forall v \in \mathcal{V} \tag{4a}$$
$$= v' L(\tilde{\mathcal{T}}_0) v + v' L_e v \quad \forall v \in \mathcal{V} \tag{4b}$$
$$= v' L(\tilde{\mathcal{T}}_0) v + w_e (v_i - v_j)^2 \quad \forall v \in \mathcal{V}. \tag{4c}$$

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 1. 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 1. This exchange is iteratively applied on the current solution until no improvements can be made.

IV. Tabu Search Heuristic

Algorithm 2 : Tabu search heuristic

Let s^* be the best optimal solution found by the algorithm and let λ_2^* be the algebraic connectivity of s^*. Let m denote the number of edges in the initial feasible solution (*i.e.*, $m := |V| - 1$). Also, let $MaxIterations$ represent the maximum number of iterations allowed in the tabu heuristic.

$\mathcal{T}_0 \leftarrow$ Initial feasible solution satisfying diameter constraints
$s := \mathcal{T}_0$, $\lambda_2^* := \lambda_2(\mathcal{T}_0)$, $s^* := \mathcal{T}_0$, T is an empty tabu list which can store at most m solutions.
for $i = 1$ to $MaxIterations$ **do**
  for edge e_k in s **do**
    Construct the set of edges $N(s, e_k)$
  end for
  for any spanning tree s' such that s' contains exactly one edge from each of the sets in $\{N(s, e_1), \cdots, N(s, e_k)\}$ **do**
    if s' is feasible **then**
      if $\lambda_2(s') > \lambda_2^*$ **then**
        $s = s'$
        $T \leftarrow T \bigcup \{s'\}$; *$T$ only stores at most m solutions from iterations $i - m + 1, \cdots, , i - 1, i$*
        $\lambda_2^* = \lambda_2(s)$, $s^* = s$
        break
      else
        if s' is not in T **then**
          $s = s'$
          $T \leftarrow T \bigcup \{s'\}$; *$T$ only stores at most m solutions from iterations $i - m + 1, \cdots, , i - 1, i$*
          break
        end if
      end if
    end if
  end for
  if $s \neq s'$ **then**
    break *//The heuristic terminates here if there is no feasible tree s' in the neighborhood of s or if every feasible s' with a lower algebraic connectivity ($< \lambda_2^*$) is already in the tabu list.*
  end if
end for
output λ_2^* and s^*

In this section, we present a tabu search heuristic (algorithm 2) for finding good feasible solutions for the algebraic connectivity problem. Similar to the $k-$opt heuristic, the tabu search heuristic first starts with a feasible solution s and attempts to construct a candidate feasible solution s', in the neighborhood of s with a larger algebraic connectivity (we will later explain how we choose the neighborhoods for the tabu search). If the algebraic connectivity of s' ($\lambda_2(s')$) is larger than the best known value (λ_2^*), s'

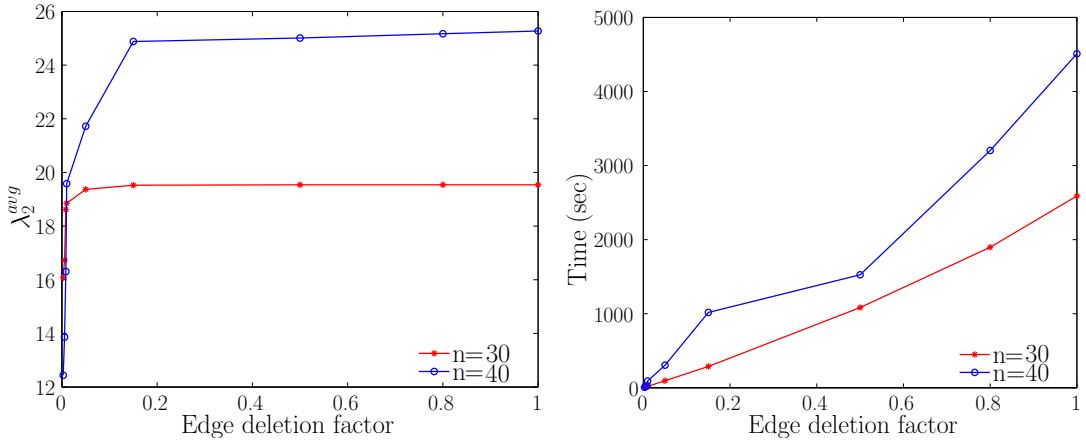
is set as the current feasible solution ($i.e.$, $s := s'$) and λ_2^* is updated (steps 9-13 in the algorithm). If the algebraic connectivity of s' is at most equal to λ_2^*, s' is set as the current feasible solution but no updates are made to λ_2^* (steps 15-19 in the algorithm). The above procedure is then repeated again until the number of iterations reaches a maximum limit.

There are two features that makes the tabu search heuristic different from the $k-$opt heuristic. Firstly, unlike the $k-$opt heuristic which aims to make an exchange only if a better solution is found, the tabu search allows for the heuristic to move to solutions that may have a smaller algebraic connectivity than the current best solution. Even though there may not be any improvement in the short run, the tabu search heuristic can end up finding better solutions as the number of iterations increases. Secondly, in any iteration, s' is set as the current feasible solution only if s' has not been encountered in any of the previous m iterations. This is accomplished by creating a tabu list and storing all the recent m feasible solutions that has been previously considered by the algorithm in this list. Therefore, when the algebraic connectivity of s' is found to be at most equal to λ_2^*, s' is set as the current feasible solution only if s' is not present in the tabu list.

The overall tabu search heuristic is presented in algorithm 2. In the remainder of this subsection, we explain how the local neighborhoods are chosen for the current feasible solution s. Suppose e_i is the i^{th} edge in s. Let the edge e_i join vertices v_1 and v_2 in the solution s. Also, define $N(s, e_i)$ as the set of all the edges that are incident on v_1 or v_2 but not present in s. Then, the neighborhood of the current solution s consists of any solution s' such that there is exactly one edge present in s' from $N(s, e_i)$ for $i = 1, \cdots, |V| - 1$.

V. Computational results

The k-opt and tabu search heuristics were implemented in Matlab. The bound on the diameter of a feasible tree was set to four ($D = 4$). The diameter of a tree is found by implementing the well known Floyd Warshall algorithm [17]. To check for feasibility, we find the diameter of the given tree using the Floyd Warshall algorithm and verify if this diameter is at most equal to the prescribed bound. For the tabu search heuristic, the size of the tabu search list was set to 20 and the parameter $MaxIterations$ was set to 100.



(a) (b)

Fig. 2: Average values of the algebraic connectivity (a) and computation times (b) obtained as a function of the edge deletion factor using the 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.

For the $k-$opt heuristic, we set the edge deletion factor (Section III-A) to be equal to 0.15 in all the simulations. We chose this value based on the simulation results shown in Figure 2. This figure shows the average algebraic connectivity of the final solution (and the computation time) obtained using the $3-$opt

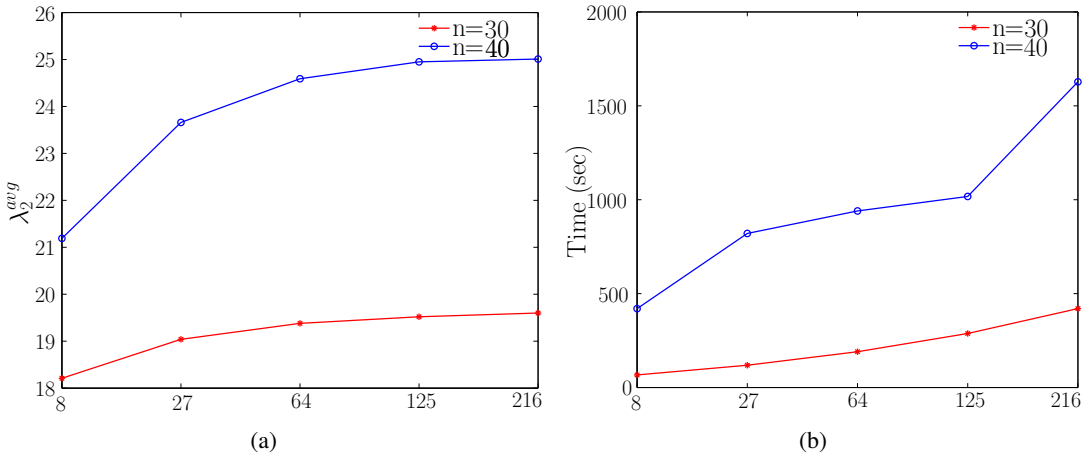


Fig. 3: The average algebraic connectivity (a) and computation times (b) obtained as a function of the maximum number of edge combinations considered for addition between any two components in the 3-opt heuristic over ten instances. In these computations, the edge deletion factor was set to 0.15.

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 (Section III-B) to be at most equal to 5^k. For 3-opt, this parameter was set to 125. We chose this value for 3-opt based on the simulation results shown in Figure 3. For 2-opt, this parameter was set to 25 after performing similar simulations.

The semi-definite programming toolboxes (Yalmip and Sedumi) in Matlab could not solve the proposed formulation (6) with the semi-definite and diameter constraints even for instances with 5 nodes primarily due to inefficient memory management. Therefore, the algorithm (3) presented in the appendix was implemented to find an optimal solution for instances up to 8 nodes. For more than 8 nodes, the algorithm in the appendix could not find an optimal solution in a reasonable amount of run time. This algorithm was implemented in C++ programming language and the resulting Integer Linear Programs (ILP's) were solved using CPLEX 12.2 with all the solver options set to default. All the developed algorithms were run on a Dell Precision T5500 workstation (Intel Xeon E5630 processor @ 2.53GHz, 12GB RAM).

Let us now compare the deviation of the solutions found by the algorithms with respect to the optimal solution. We define deviation (in %) as $\frac{\lambda_2^* - \lambda_2^{heuristic}}{\lambda_2^*} \times 100$ where $\lambda_2^{heuristic}$ denotes the algebraic connectivity of the solution found by the heuristic and λ_2^* represents the optimum. The results shown in Table I are for 10 random instances generated for networks with 8 nodes. Based on the results in Table I, we observed that the average run time to obtain an optimal solution was around 371 seconds. Also, it can be seen that the k-opt ($k = 2, 3$) and tabu search heuristic found optimal solutions for all problem instances with 8 nodes. Each of the heuristics ran within a fraction of a second for each of the instances with 8 nodes. Instance 1 of Table I is pictorially shown in Figure 4.

For more than 8 nodes, we used the heuristics to generate feasible solutions for the algebraic connectivity problem. In the remainder of this section, we compare the results from the $k-$opt and tabu search heuristics. As mentioned in the article previously, choosing an initial solution for the heuristics is relatively straightforward; we choose a star graph with the largest algebraic connectivity ($\lambda_2^{initial}$) as an initial feasible solution. A sample initial feasible solution is shown in part (b) of Figure 4.

For more than 8 nodes, we inferred the deviation of a solution for an instance by comparing it with the solution obtained by using the 3-opt heuristic for the same instance. Specifically, the deviation of a solution is defined as $\frac{\lambda_2^{3opt} - \lambda_2^{heuristic}}{\lambda_2^{3opt}} \times 100$ where λ_2^{3opt} is the algebraic connectivity of a solution obtained by using the 3-opt heuristic. For each problem size, the *average* deviation and computation time obtained for the 10 instances is shown in table II. For each of the instances, the solution obtained by the 3-opt

heuristic had the largest algebraic connectivity.

From the results in table II, it is clear that the 2-opt heuristic performed better than the tabu search for medium sized instances (up to 30 nodes). For larger problems (40, 45, 55 and 60 nodes), we observed that the quality of the solutions found by the tabu search was better compared to the 2-opt heuristic. In summary, we observe that the 2-opt heuristic provides a good trade-off between obtaining good feasible solutions and the required computation time. Figure 5 illustrates the solutions obtained using the k-opt and tabu search heuristic for a 40 nodes problem. Also, as shown in table III, we observed that the *maximum* deviation (out of 10 instances) of the 2-opt solutions remained lower than that of tabu search for instances up to 30 nodes. For large instances (35, 40, 45, 55 and 60 nodes), the *maximum* deviation for tabu search remained lower that that of 2-opt solutions. Overall, the tabu search heuristic performed better than the 2-opt search heuristic for most of the large instances.

TABLE I: Comparison of the deviation of the solutions found by the heuristics for networks with 8 nodes. λ_2^* is the optimal algebraic connectivity.

No.	Optimal solution		k-opt search (k=2,3)		Tabu search	
	λ_2^*	Time (sec)	λ_2^{kopt}	Deviation (%)	λ_2^{tabu}	Deviation (%)
1	3.9712	180.57	3.9712	0.00	3.9712	0.00
2	4.3101	408.10	4.3101	0.00	4.3101	0.00
3	3.9297	621.85	3.9297	0.00	3.9297	0.00
4	3.5275	216.79	3.5275	0.00	3.5275	0.00
5	3.8753	470.63	3.8753	0.00	3.8753	0.00
6	3.7972	342.14	3.7972	0.00	3.7972	0.00
7	3.7125	377.47	3.7125	0.00	3.7125	0.00
8	3.9205	313.12	3.9205	0.00	3.9205	0.00
9	3.7940	341.84	3.7940	0.00	3.7940	0.00
10	3.8923	316.86	3.8923	0.00	3.8923	0.00
Average				0.00		0.00

TABLE II: Comparison of 3-opt, 2-opt and tabu search heuristic solutions for various problem sizes. Here, the deviation was averaged over ten random instances for each n.

n	3-opt		2-opt		Tabu	
	Deviation (%)	Time (sec)	Deviation (%)	Time (sec)	Deviation (%)	Time (sec)
10	0	0.29	0.00	0.10	0.00	0.77
15	0	3.06	0.01	0.52	1.52	9.36
20	0	16.32	0.27	1.65	1.78	25.58
25	0	60.38	0.60	3.59	1.88	60.19
30	0	274.93	2.07	12.13	3.12	144.92
35	0	480.15	2.02	37.85	2.77	313.20
40	0	1016.99	5.62	57.28	2.48	3470.71
45	0	2309.60	7.10	116.09	5.10	4272.58
50	0	4219.17	1.38	139.99	3.55	6675.32
55	0	6798.34	6.98	350.83	4.90	8654.29
60	0	8974.46	7.97	505.36	6.06	10049.10

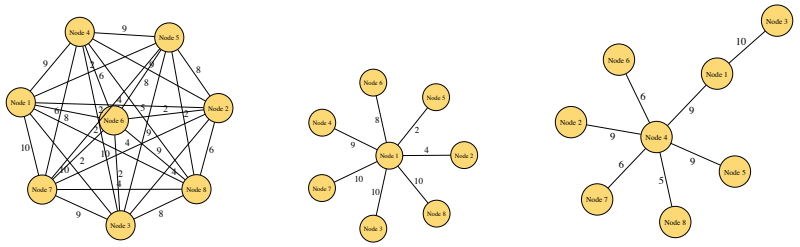
VI. Conclusions

We studied design methods to maximize the robustness of a network in the presence of diameter constraints. Algebraic connectivity is used to measure the robustness of the network. Our design methods mainly involve the development of heuristics. Our first method, the k-opt heuristic, aims to iteratively

TABLE III: Comparison of maximum deviations computed over ten random instances for each n for 2-opt and tabu search heuristic.

n	Maximum Deviation (%)	
	2-opt	Tabu
10	0.00	0.00
15	0.08	6.67
20	2.73	10.86
25	4.92	6.36
30	7.56	9.07
35	12.59	7.64
40	17.89	8.91
45	15.41	12.23
50	5.10	8.68
55	17.56	11.47
60	16.92	13.91

search for better solutions by performing an exchange of edges in each iteration. Unlike the k-opt heuristic, our second method (the tabu search heuristic) iteratively moves to solutions that may have a lower algebraic connectivity in the short run but could find better solutions as the number of iterations increases. Computational results suggested that the 3-opt heuristic performed the best while the 2-opt heuristic provided a good trade-off between finding good solutions and the required computation time. There are several future directions for this work. One research direction would be do generalize the network synthesis problem to include more connectivity constraints. Another research direction would be to compare the performance of the developed algorithms in a simulation setting similar to the one in [4] and test the performance in the presence of node or edge-failures.



(a) Complete network (b) $\lambda_2^{initial}$ = 2.2045 (c) λ_2^{tabu},λ_2^{2opt},λ_2^{3opt} = 3.9712

Fig. 4: 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, 3-opt and Tabu search methods.

REFERENCES

[1] H. Nagarajan, S. Rathinam, S. Darbha, and K. R. Rajagopal, "Algorithms for identifying stiff structures with maximal natural frequencies," *Nonlinear analysis: real world applications*, 2012.

[2] R. Guimera and L. Amaral, "Modeling the world-wide airport network," *European Physical Journal B*, vol. 38, pp. 381–385, 2004.

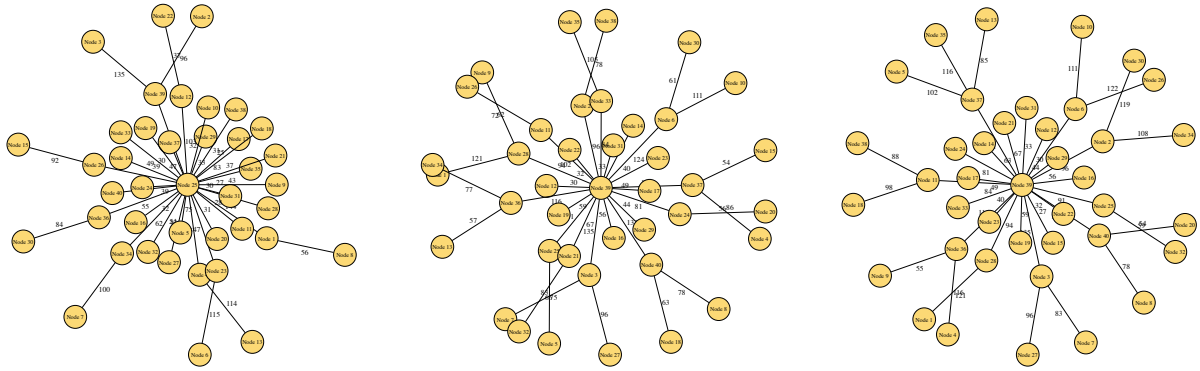
[3] R. Kincaid, N. Alexandrov, and M. Holroyd, "An investigation of synchrony in transport networks," *Complexity*, vol. 14, no. 4, pp. 34–43, 2008.

[4] E. Vargo, R. Kincaid, and N. Alexandrov, "Towards optimal transport networks," *Systemics, Cybernetics and Informatics*, vol. 8, no. 4, pp. 59–64, 2010.

[5] S. Conway, "Scale-free networks and commercial air carrier transportation in the united states," in *24th International Congress of the Aeronautical Sciences*, Yokohama, Japan, 2004.

[6] N. Alexandrov, "Transportation network topologies," NASA Langley Research Center, Tech. Rep., 2004.

[7] A. Jamakovic and S. Uhlig, "On the relationship between the algebraic connectivity and graph's robustness to node and link failures," in *Next Generation Internet Networks, 3rd EuroNGI conference*, 2007, pp. 96–102.



(a) 2-opt exchange, $\lambda_2^{2opt} = 21.4088$ (b) Tabu search, $\lambda_2^{tabu} = 25.0408$ (c) 3-opt search, $\lambda_2^{3opt} = 26.0731$

Fig. 5: 2-opt, 3-opt exchange and Tabu search solutions for a network with 40 nodes where $D = 4$.

[8] A. Jamakovic and P. V. Mieghem, "On the robustness of complex networks by using the algebraic connectivity," in *Networking 2008 Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet*, A. D. et al, Ed., 2008, pp. 183–194.

[9] R. H. Byrne, J. T. Feddema, and C. T. Abdallah, "Algebraic connectivity and graph robustness," Sandia National Laboratories, Albuquerque, New Mexico 87185, Tech. Rep., July 2009.

[10] D. Mosk-Aoyama, "Maximum algebraic connectivity augmentation is NP-hard," *Operations Research Letters*, vol. 36, no. 6, pp. 677–679, 2008.

[11] H. Nagarajan, S. Rathinam, S. Darbha, and K.R.Rajagopal, "Algorithms for finding diameter-constrained graphs with maximum algebraic connectivity," *Dynamics of Information Systems: Mathematical Foundations*, 2011.

[12] A. Ghosh and S. Boyd, "Growing well-connected graphs," in *Proceedings of the 45th IEEE Conference on Decision and Control*, December 2006, pp. 6605–6611.

[13] P. Wei and D. Sun, "Weighted algebraic connectivity: An application to airport transportation network," in *the 18th IFAC World Congress*, Milan, Italy, Sep 2011.

[14] T. Kotegawa, D. DeLaurentis, K. Noonan, and J. Post, "Impact of commercial airline network evolution on the u.s. air transportation system," in *Ninth USA/Europe Air Traffic Management Research and Development Seminar (ATM2011)*, Berlin, Germany, 2011.

[15] M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak Mathematics Journal*, vol. 23, pp. 298–305, 1973.

[16] G. A. Croes, "A method for solving Traveling-Salesman problems," *Operations Research*, vol. 6, no. 6, pp. 791–812, Nov. 1958, ArticleType: research-article / Full publication date: Nov. - Dec., 1958 / Copyright 1958 INFORMS. [Online]. Available: http://www.jstor.org/stable/167074

[17] R. W. Floyd, "Algorithm 97: shortest path," *Communications of the ACM*, vol. 5, no. 6, p. 345, 1962.

[18] "IBM - ILOG: CPLEX optimization studio 12.2. url: http://www.ilog.com/products/cplex." [Online]. Available: http://www.ilog.com/products/cplex

VII. Appendix

For completeness, we summarize the algorithm in [11] for finding an optimal solution to the algebraic connectivity problem. There are two issues that need to be addressed before one can solve the formulation in (2). Firstly, the diameter constraint, $\delta_{ij}(x) \leq D$, must be converted to a more tractable form so that standard optimization tools can be used to solve it. Secondly, one must develop an algorithm to find an optimal solution to solve the resulting algebraic connectivity problem.

A. Handling the diameter constraint

We assume D is even. A similar approach can be used when D is odd. To impose the the diameter constraint, we add a source node (r) to the graph (V, E) and add an edge joining r to each vertex in V, *i.e.*, $\tilde{V} = V \cup \{r\}$ and $\tilde{E} = E \cup (r, j) \quad \forall j \in V$. We then construct a tree spanning all the nodes in $\tilde{V}$ while restricting the length of the path from r to any other node in $\tilde{V}$. The additional edges emanating from the root vertex are used only to formulate the diameter constraints, and they do not play any role in determining the algebraic connectivity of the original graph. Hence, the diameter constraints in (2d) can be equivalently formulated using the multi-commodity flow constraints as shown below:

$$\sum_{j\in\tilde{V}\setminus\{r\}}(f_{ij}^k - f_{ji}^k) = 1 \quad \forall k \in V \text{ and } i = r, \tag{5a}$$

$$\sum_{j\in\tilde{V}}(f_{ij}^k - f_{ji}^k) = 0 \quad \forall i,k \in V \text{ and } i \neq k, \tag{5b}$$

$$\sum_{j\in\tilde{V}}(f_{ij}^k - f_{ji}^k) = -1 \quad \forall i,k \in V \text{ and } i = k, \tag{5c}$$

$$f_{ij}^k + f_{ji}^k \leq x_e \quad \forall\ e := (i,j) \in \tilde{E}, \forall k \in V, \tag{5d}$$

$$\sum_{e\in\tilde{E}} x_e = |\tilde{V}| - 1, \tag{5e}$$

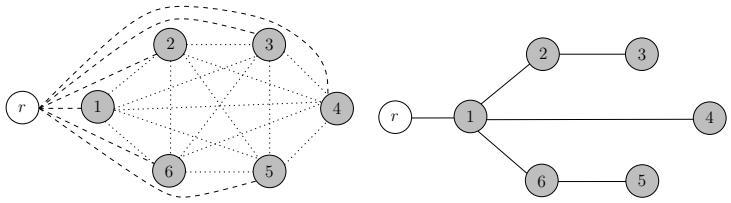
$$0 \leq \ f_{ij}^k \ \leq 1 \quad \forall i,j \in \tilde{V}, \forall k \in V, \tag{5f}$$

$$x_e \in \{0,1\} \quad \forall e \in \tilde{E}, \tag{5g}$$

$$\sum_{(i,j)\in\tilde{E}} f_{ij}^k \leq (D/2+1) \quad \forall k \in V, \tag{5h}$$

$$\sum_{j\in V} x_{rj} = 1, \tag{5i}$$

where f_{ij}^k is the k^{th} commodity flowing from node i to node j. Constraints (5a) through (5c) state that each commodity must originate at the root node and terminate at its corresponding vertex. Equation (5d) states that the flow of commodities between two vertices is possible only if there is an edge joining the two vertices. Constraint (5e) 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 r to k and hence (5h) represents the diameter constraint. Therefore, the constraints in (5) can be used to replace the diameter (2d) and spanning tree constraints (2b) in the formulation.



(a) Original graph augmented with a dummy node, r

(b) Feasible solution

Fig. 6: Illustration of an addition of a root node (r) 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 root node to $(D/2) + 1 = 3$, and allowing only one incident edge on r will suffice as shown in (b).

B. Optimal algorithm

The second issue is to reduce the problem in (2) to a pure Binary Semi-Definite Programming problem (BSDP) as the tools associated with the construction of valid inequalities are more abundant for BSDPs as compared to mixed-integer programs. Also, with the further relaxation of the semi-definite constraint, it can be solved using CPLEX [18], a high performance solver for Integer Linear Programs (ILP). Therefore, we adopt a different approach for finding an optimal solution by casting the algebraic connectivity problem

as the following decision problem: Is there an augmented network such that the algebraic connectivity of the network is at least equal to a pre-specified value ($\hat{\lambda}_2$) and the diameter of the graph is at most equal to D? This problem can be posed as a BSDP by choosing to find a spanning tree that minimizes the degree of the root vertex r. This can be mathematically written as follows:

$$\begin{aligned} & \min \textstyle\sum_{e\in\delta(r)} x_e \\ \text{s.t.} \quad & L(x) \succeq \hat{\lambda}_2(\tilde{I}_n), \\ & \text{Diameter and spanning tree} \\ & \quad \text{constraints as in (5).} \end{aligned} \tag{6}$$

In this formulation, $\delta(r)$ denotes a cutset defined as $\delta(r) = \{e = (r, j) : \ j \in V \setminus r\}$. The above BSDP can be efficiently solved by a cutting plane procedure and then a bisection algorithm can be used to maximize the algebraic connectivity. In the cutting plane procedure, the semi-definite constraint is replaced by a finite subset of the infinite number of linear constraints and successively tighter polyhedral approximations are constructed by augmenting valid inequalities until a feasible solution is obtained by satisfying a desired level of connectivity. Based on the notation defined in this article, a detailed pseudo code of this procedure is outlined in Algorithm (3). Steps 4 though 16 of Algorithm (3) employs the cutting plane procedure until the semi-definite constraint is satisfied and step 17 is the bisection step where $\hat{\lambda}$ is incremented until the optimization problem becomes infeasible.

Algorithm 3 : Synthesizing robust networks with maximum algebraic connectivity subject to diameter constraints

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

1: Input: A graph $G = (V, E)$, w_e, r, D, and a finite number of unit vectors, $v_i, i = 1 \dots M$
2: Choose any spanning tree satisfying the diameter constraint as an initial feasible solution, x^*
3: $\hat{\lambda} \leftarrow \lambda_2(L(x^*))$
4: **loop**
5: $\quad \mathfrak{F} \leftarrow \emptyset$
6: $\quad$ Replace the semi-definite constraint in (6) with $({v_i}^T(L(x))v_i) \ \geq \ \hat{\lambda}({v_i}^T\tilde{I}_n v_i) \quad \forall i = 1, .., M$, and with additional constraint, x satisfying $\mathfrak{F}$ and solve the ILP.
7: $\quad$ **if** the above ILP is infeasible **then**
8: $\quad\quad$ **break loop** $\{x^*$ is the optimal solution$\}$
9: $\quad$ **else**
10: $\quad\quad$ $x^* \leftarrow$ solution to the above ILP
11: $\quad\quad$ $\gamma^* \leftarrow \lambda_2(L(x^*))$
12: $\quad\quad$ **if** $L(x^*) \not\succeq \gamma^*(\tilde{I}_n)$ **then**
13: $\quad\quad\quad$ Augment $\mathfrak{F}$ with a constraint $({v^*}^T L(x^*)v^*) \ \geq \ \gamma^*({v^*}^T\tilde{I}_n v^*)$ where v^* is the eigenvector corresponding to a negative eigenvalue of $L(x^*) - \gamma^*\tilde{I}_n$.
14: $\quad\quad\quad$ Go to step 6.
15: $\quad\quad$ **end if**
16: $\quad$ **end if**
17: $\quad \hat{\lambda} \leftarrow \hat{\lambda} + \epsilon$ {let ϵ be a small number}
18: **end loop**