Air transportation network robustness optimization under limited legs itinerary constraint

Harsha Nagarajan, Peng Wei, Sivakumar Rathinam and Dengfeng Sun

***Abstract*—In air transportation networks the robustness of a network regarding node and link failures is a key factor for its design. At the same time, travelling passengers usually prefer the itinerary with fewer legs. In this paper the authors propose three design methods, which can maximize the air transportation network robustness and make sure every itinerary within this network have fewer legs than a given number. In the first design method, the problem is formulated as a mixed-integer, semi-definite program, and an algorithm to obtain optimal solutions is applied based on cutting plane and bisection methods. The second and third method is to apply local search heuristics like tabu search and 2-opt respectively to find a potentially global optimal design solution. The simulation results present the performance of our proposed algortihms.**

I. Introduction

An air transportation network consists of distinct airports (cities) and direct flight routes between airport pairs [1]. Usually a graph $G(V, E)$ is used to describe an air transportation network [2], [3], where the node set V represents all the n airports and the edge set E represents all the m direct flight routes between airports. With the fact that if a direct flight route from airport a to airport b exists, normally the direct return route from airport b to airport a also exists [4], $G(V, E)$ is constructed as an undirected simple graph, where the airports are indexed as $\{v_i | i = 1, 2, \ldots, n\}$ and the direct flight routes are named as $\{e_{ij} | \text{if there is a direct route between airports } i \text{ and } j\}$.

According to the definition of the simple graph, it is required that no more than one edge exists between any two nodes. However, in reality multiple flight routes operated by one or more airlines exist between two airports. Thus we introduce non-negative edge weights to record multiple fight routes information to maintain G as a simple graph, i.e., the edge weight w_{ij} could be the number of distinct direct flight routes between airports i and j.

Connectivity is the metric which is used to measure how well a network is connected. Usually the more connected is a network, the more robust it is [5]. The air traffic demand is expected to continue its rapid growth in the future. The Federal Aviation Administration (FAA) estimated that the number of passengers is projected to increase by an average of 3% every year until 2025 [6], so the expanding load on current air transportation network will cause more and more flight delays and cancellations with the limited airspace and sector capacities. As a result, a robust air transportation network design scheme is necessary to handle the increasing traffic demand. That is the major motivation of this work.

Traditionally, the *node connectivity* and the *edge connectivity* are two metrics to evaluate how well a graph is connected [7]. The node (edge) connectivity of a graph G is the minimum number of node (edge) deletions sufficient to disconnect G.

Fig. 1: N-node line topology.

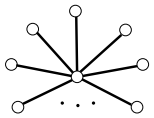


Fig. 2: N-node star topology.

In order to show the limitation of node (edge) connectivity evaluation, two different topologies are shown, where Fig. 1 is a N-node line topology and Fig. 2 is a N-node star topology. Obviously, the node connectivity for both topology formations is 1 and so is the edge connectivity. However, the star topology should be more robust than the line topology because in Fig. 2 the network is disconnected only when the central node fails, while in Fig. 1 any node failure can cause the network disconneted except the two end nodes. The robustness features of two topologies are intuitively different. Since neither node connectivity nor edge connectivity can observe the difference between these two topologies, we need to introduce a new metric.

The second smallest Laplacian eigenvalue of a graph is related to many graph invariants. Fiedler defines it as the *algebraic connectivity* [8] of graph G. According to the definition, when $N = 4$, the algebraic connectivities of Fig. 1 and Fig. 2 are 0.588 and 1 respectively, which show that the star topology is more robust than the line topology.

Now we see the algebraic connectivity in unweighted graphs provides better resolution on how well a graph or network is connected and can be considered as a measurement of the robustness in complex network models which is proved in [5]. So the algebraic connectivity may also be considered to evaluate the robustness of the air transportation network in weighted graphs.

As we mentioned, in order to maintain the graphic description of the air transportation network as a simple graph and obtain more network information, it is necessary to introduce the weighted graphs to describe the air transportation network.

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.

The authors in [9] first introduced the weighted air transportation network and they studied the correlations of betweenness-degree, degree-degree and clustering-degree of the network. In [10], Wei and Sun built a different network and studied the weighted algebraic connectivity optimization in the robustness evaluation of air transportation networks with the constraint of k additional edges. In this paper, the authors first introduce the limited leg number constraint and then two algorithms are proposed to solve the new formulated problem.

From the passenger perspective, it is natural to introduce the limited leg number constraint. When a passenger plans his or her itinerary, no one wants to have too many stops for connecting flights. When we construct an air transportation network, besides the k additional edges constraint, we also require that betweeen any two airports, all the itineraries must have their leg numbers fewer than a given value D.

The *weighted adjacent matrix* A of graph G has the ith row and jth column entry a_{ij}. The diagonal items are all zeros and the off-diagonal item a_{ij} $(i \neq j)$ is equal to weight w_{ij}:

$$a_{ij} = \begin{cases} w_{ij}, & \text{if node } i \text{ and node } j \text{ are connected} \\ & \text{by an edge } e_{ij} \text{ with weight } w_{ij}; \\ 0, & \text{if node } i \text{ and } j \text{ are not connected.} \end{cases} \tag{1}$$

where the weights are usually bounded by an upper limit W because a weighting scheme without an upper bound is normally not applicable in practice.

Then the *weighted Laplacian matrix* L can be obtained from the adjacent matrix A. Each item l_{ij} of L can be written as:

$$l_{ij} = \begin{cases} -a_{ij}, & \text{if } i \neq j; \\ \sum_{i=1}^{n} a_{ij}, & \text{if } i = j. \end{cases} \tag{2}$$

The second smallest eigenvalue λ_2 of L is the *weighted algebraic connectivity*, which is the focus of this paper.

In fact [11] shows that λ_2 of the weighted algebraic connectivity is the performance metric for a weighted graph. If we want to improve the robustness of the air transportation network, we only need to increase λ_2 of its corresponding weighted graph. Therefore the aim of this paper is to maximize λ_2 under the given constraints.

The methods developed or improved in this work are expected to be implemented as the network performance measurement and the guidance to enhance the robustness in air transportation networks.

The rest of the paper is organized as follows. In Section II we formulate the problem and show the problem is NP-hard. In Section III an optimal algorithm is demonstrated, which is based on cutting plane and bisection methods. Local search heuristics like Tabu search and 2-opt exchange search are presented in Sections IV and V respectively. In Section VI we evaluate the performance of our algorithms via simulations. Section VII concludes this paper.

II. Problem formulation and its NP-hardness

A. Problem formulation

Given the graphic description $G(V, E_0)$ of an existed weighted air transportation network, where the node set V is the collection of all the airports in this network and the edge set E_0 contains the existed routes between the airport pairs. The size of set V is n and the size of E_0 is m. The objective is to maximize λ_2 with a fixed number k of edge additions or deletions based on E_0 while the edges to be added or deleted are given in a pre-determined set P (for addition) or Q (for deletion). In this work only the addition case is considered for the ease of illustration. We denote the routes to be added as a set of ΔE. δ_{ij} is the number of legs on the shortest path between v_i and v_j. Thus the *flight routes addition problem with limited legs itinerary* is:

$$\begin{aligned} & \max \lambda_2(G(E_0 + \Delta E)) \\ \text{s.t.} \quad & |\Delta E| = k, \\ & \Delta E \subseteq P, P \cap E_0 = \emptyset, \\ & \delta_{ij} \leq D. \end{aligned} \tag{3}$$

B. Alternative problem formulation

We can also denote $\lambda_2(G)$ as $\lambda_2(L)$, in which L is the weighted Laplacian matrix of graph G. According to [12], the weighted Laplacian matrix L can be represented by the dot product summation of *edge vectors*. For an edge e connecting two nodes i and j, we define the edge vector $h_e \in \mathbb{R}^n$ as $h_e(i) = 1$, $h_e(j) = -1$, and all other entries 0. w_e is the non-negative weight on e. Suppose there are m edges in graph G. The weighted Laplacian matrix L of G is the $n \times n$ matrix:

$$L = \sum_{e=1}^{m} w_e h_e h_e^T. \tag{4}$$

which is equivalent to the weighted Laplacian matrix definition in Eq. (2).

The reason that algebraic connectivity is a good metric for how well a network keeps connected is that $\lambda_2(L)$ is monotone increasing with the edge set: if $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ are such that $E_1 \subseteq E_2$, then $\lambda_2(G_1) \leq \lambda_2(G_2)$ [8]. That is, the more connected graph (on the same vertex set) has the greater algebraic connectivity.

On the other hand, the air transportation network robustness exactly means how well-connected the network is. A network's link may be broken down by severe weather conditions, aviation regulation changes or economics issues. Despite these natural and human effects, the airports should maintain robust connections with each other to make sure the air traffic flows smoothly.

According to the edge vector description of L in Eq. (4), the flight routes addition problem (3) can be written as the following mixed integer semi-definite programing problem based on the proof shown in [13]:

$$\begin{aligned} & \max \lambda_2 \\ \text{s.t.} \quad & L(x) \succeq \lambda_2(I_n - e_0 e_0^T) \\ & \mathbf{1}^T x = k, \\ & x \in \{0, 1\}^{|P|}, \\ & \delta_{ij} \leq D. \end{aligned} \tag{5}$$

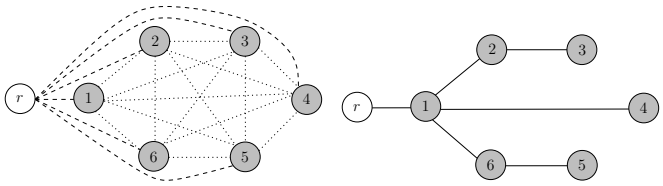
where $L(x) = L_0 + \sum_{e=1}^{|P|} x_e w_e h_e h_e^T$ is the weighted Laplacian matrix of the augmented network, L_0 is the weighted Laplacian matrix of the existed network $G(V, E_0)$ to which a fixed number of k edges are to be added, $I_n \in \mathbb{R}^{n \times n}$

is an identity matrix and e_0 is the normalized eigenvector corresponding to the first eigenvalue of $L(x)$. "$\succeq$" refers to the positive-semidefiniteness of a matrix. P is the pre-determined set with size $|P|$ which contains the candidate edges to be added. e is the index for candidate edges in P. x_e is a boolean variable, in which 1 means that edge e in P is in the set of ΔE in (3) and 0 means that edge e in P is not in ΔE. x is a vector consisting of all x_e's whose length is $|P|$, illustrating which candidate edges are to be added and which are not.

When the value D is set to N, the problem in (5) is reduced to the proved NP-hard formulation in [10]. Therefore (5) is NP-hard.

III. An algorithm to compute optimal solutions

The proposed algorithm in this section is the modified version of the algorithms presented in [13], [14]. Before we discuss the idea behind this algorithm, two important issues need to be addressed in the formulation (5). First issue is to handle the diameter constraint or the limited legs itinerary constraint. To simplify the problem, we restrict our search for optimal networks from the set of all spanning trees. Also spanning trees help to minimize the overall cost of construction and maintenance of the air transportation network but still keeping the network connected. It is a well known result that any spanning tree has a diameter not more than an even integer (D) if and only if there exists a central node p such that any path from p to other nodes consists of at most $D/2$ edges. Since p is not known a priori, we augment the network with a dummy root vertex, r connecting all other vertices of the network with dummy edges as shown in Figure 3. Hence the constraints representing the spanning tree with diameter constraints is posed as a network flow model using the multi commodity flow formulation.



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

(b) Feasible solution

Fig. 3: Illustration of an addition of dummy 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).

The second issue is to reduce the problem in (5) to a pure binary semi-definite programming problem (BSDP) since the tools associated with construction of valid inequalities are more abundant when compared to mixed-integer programs. Also, with further relaxation of the semi-definite constraint, it can be solved using CPLEX, a high performance solver for integer linear programs. 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 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:

$$\begin{aligned} & \min \textstyle\sum_{e\in\delta(r)} x_e \\ \text{s.t.} \quad & L(x) \succeq \lambda_2(I_n - e_0 e_0^T), \\ & \mathbf{1}^T x = k,\ x \in \{0,1\}^{|P|}, \\ & \delta_{ij} \leq D. \end{aligned} \tag{6}$$

In this formulation, the only decision variables would be the binary variables denoted by x_e and $\delta(r)$ denotes a cutset defined as $\delta(r) = \{e = (r, j) :\ j \in V \setminus r\}$. The above BSDP can be efficiently solved by cutting plane procedure and then 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 till a feasible solution is obtained 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 (1). Steps 4 though 16 of Algorithm (1) does the cutting plane procedure till 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 1 : 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, $v_i, i = 1 \ldots M$
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 $(v_i \cdot (L(x))v_i) \geq \hat{\lambda}(v_i \cdot (I_n - e_0 e_0^T)v_i) \quad \forall i = 1, .., M$, 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** $L(x^*) \not\succeq \gamma^*(I_n - e_0 e_0^T)$ **then**
13: $\quad\quad\quad$ Augment $\mathfrak{F}$ with a constraint $(v^* \cdot L(x^*)v^*) \geq \gamma^*(v^* \cdot (I_n - e_0 e_0^T)v^*)$ where v^* is the eigenvector corresponding to a negative eigenvalue of $L(x^*) - \gamma^*(I_n - e_0 e_0^T)$.
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**

IV. Tabu search for limited legs

Mixed Integer Semi-Definite Programming is usually slow, therefore applying a heuristic search can provide computa-

tional efficiency. We are interested in finding k edges to add to G, which gives the maximal $\lambda_2(G(V,E))$.

A. Tabu search details

Now we elaborate the details of the tabu search implemented in this work. The search is an iterative process and the next round solution s' is generated from the neighbor of the current solution s supervised by a dynamically updating tabu list T.

1) Dynamic neighbor: The tabu search looks for the next round solution s' inside the neighbor $N(s)$ of the current solution s. After s' is chosen and checked to be feasible, the following round solution will be selected from its neighbor $N(s')$.

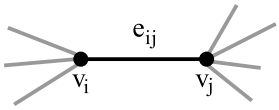


Fig. 4: The neighbor of the pth edge e_{ij} in current solution s.

Instead of the swapping operation in [2], [3], we define $N(s)$ for our problem. Inside a given initial solution s, there are k edges to be added. The pth ($1 \leq p \leq k$) edge e_{ij} in solution s connects two nodes v_i and v_j, which is shown in Fig. 4. Thus all the edges incident to v_i or v_j and in set P constitute the sub-neighbor $N(s,p)$ of solution s at the pth edge. The edges which already exist in G or are not in candidate set P are not displayed in Fig. 4. To prevent $N(s,p)$ from being empty, a random jump inside P is also included in $N(s,p)$, which jumps to any edge in P but not current edges 1 to k in solution s. If the random jump gives out an existing edge in $N(s,p)$, then we execute another random jump. The neighbor of current solution s is $N(s)$, which is the union of the sub-neighbors of every edge in s:

$$N(s) = \bigcup_{p=1}^{k} N(s,p). \tag{7}$$

Notice that each of the k new edges will be picked from its own sub-neighbor $N(s,p)$ and form s' together. Each solution s' needs to be checked if it satisfies the limited legs itinerary constraint (graph diameter constraint). According to [7], given k edges by s', whether the diameter of the graph $D(s')$ is fewer than a given value D is checked using the product summation of adjacency matrix A. If $\sum_{i=1}^{D} A^i$ is an all one matrix, $D(s') \leq D$.

2) Tabu list: The tabu list T records the most recent $|T|$ moves. For each edge p ($1 \leq p \leq k$) in the current solution s, all the candidate moves from edge p to its neighbor $N(s,p)$, which passed the feasible check, are checked with the tabu list. If a candidate moving repeats one of the moves in T, this candidate will not be selected.

3) Aspiration criteria: The tabu search offers an opportunity where it can violate tabu. When a searching move is improving the solution value and leading to the best observed value of λ_2, this move can be performed. So the best observed value λ_2^* needs to be updated throughout the searching process.

Algorithm 2 : Tabu search with limited legs constraint

```
1: given G and all the edge weights
2: construct initial solution s_0
3: s = s_0, λ_2^* = 0, s^* = s_0, T is set to a empty queue with the
   pre-fixed size |T|
4: for iteration = 1 to Φ do
5:    for p = 1 to k do
6:       construct N(s,p) of the pth edge in s
7:    end for
8:    while 1 do
9:       pick one edge p' from each N(s,p) to construct s'
10:      if D(s') > D then
11:         continue
12:      end if
13:      if λ_2(s') > λ_2^* then
14:         s = s', update T
15:         λ_2^* = λ_2(s), s^* = s
16:         break
17:      end if
18:      if s' is not in T then
19:         s = s', update T
20:         break
21:      end if
22:   end while
23: end for
24: output λ_2^* and s^*
```

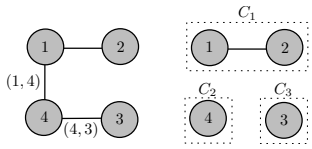
4) Complete tabu search algorithm: The complete tabu search with limited legs constraint is shown in Algorithm 2. Line 2 sets an initial solution s_0. Line 3 initializes the parameters, where s is the solution in the current round, λ_2^* and s^* record the best λ_2 and its corresponding s respectively. Line 4 shows that the algorithm terminates after Φ rounds. Line 5 to 7 constructs the sub-neighbors of the current solution. Line 9 forms s' from $N(s)$. Line 10 to 12 checks the feasibility for limited legs. Line 13 to 17 checks the aspiration criteria. Line 18 to 21 checks whether the move from s to s' is in the tabu list T.

V. 2-OPT SEARCH FOR LIMITED LEGS

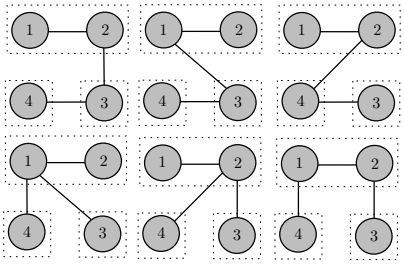
In this section, we discuss another local search heuristic called 2-opt exchange for obtaining sub-optimal solutions. This heuristic in a more general form called k-opt search was initially proposed to solve traveling salesman type problems as discussed in [15]. In [16], authors apply the 2-opt search to obtain good suboptimal solutions for the basic problem of maximizing algebraic connectivity subject to spanning tree constraints. In this paper, we extend this idea to with an additional constraint on the diameter of the graph.

Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be any two feasible solutions for problem (6), then the feasible solution, $\mathcal{T}_2$ 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 diameter constraint in our problem, 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. An illustration of such a procedure on one such pair of edges of a random spanning tree with 4 nodes is shown in Figure 5. An important issue to be addressed is to

make sure that the solutions resulting after 2-opt exchanges are also feasible. Ensuring feasibility for problem (6) is relatively easier since we are looking for spanning trees which satisfy the given diameter constraint. In the case of spanning trees, 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, $\mathcal{T}_2$. Since evaluating diameter of the graph is also computationally efficient, one may compute the feasibility of the new solution. The new feasible $\mathcal{T}_2$ is considered for replacing $\mathcal{T}_1$ if it has a better algebraic connectivity than $\mathcal{T}_1$. A pseudo-code of 2-opt heuristic is outlined 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. This procedure is repeated on the current feasible solution iteratively until no improvement is possible.



(a) Left: Remove chosen edges Right: Corresponding connected components



(b) All 2-opt feasible solutions without power constraint

Fig. 5: 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 (spanning trees).

Algorithm 3 : 2-opt exchange heuristic

1: $\mathcal{T}_0 \leftarrow$ Initial feasible solution satisfying diameter constraints
2: $\lambda_0 \leftarrow \lambda_2(L(\mathcal{T}_0)))$
3: Input: D (positive integer)
4: **for** each pair of edges $\{(u_1, v_1), (u_2, v_2)\} \in \mathcal{T}_0$ **do**
5: 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.
6: **if** $\lambda_2(L(\mathcal{T}_{opt})) > \lambda_0$ and diameter$(\mathcal{T}_{opt}) \leq D$ **then**
7: $\mathcal{T}_0 \leftarrow \mathcal{T}_{opt}$
8: $\lambda_0 \leftarrow \lambda_2(L(\mathcal{T}_{opt}))$
9: **end if**
10: **end for**
11: $\mathcal{T}_0$ is the best spanning tree in the solution space with respect to the initial feasible solution

VI. Computational results

The computational results in this section are based on the proposed algorithm (1) for finding optimal solution and heuristics given in algorithms (3) and (2). Algorithm (1) was implemented in C++ programming language and the resulting ILP's were solved using CPLEX 12.2 with all the solver options set to default. Both, 2-opt and tabu search heuristics were implemented in Matlab. All computational results in this section were implemented on a Dell Precision T5500 workstation (Intel Xeon E5630 processor @ 2.53GHz, 12GB RAM).

The semi-definite programming toolboxes (Yalmip and Sedumi) in Matlab could not solve the proposed formulation (6) with the semi-definite and multi commodity flow constraints even for instances with 5 airports 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 airports. Since we are interested in spanning trees as feasible solutions, there are $8^6 = 262144$ combinatorial possibilities (for a graph with n nodes, there are n^{n-2} possible feasible solutions).

Construction of an initial feasible solution for heuristics: For the airport transportation network problem in this paper, we consider spanning trees as feasible solutions with maximum number of itinerary legs not greater than 4 ($D \leq 4$). We chose $D \leq 4$ since it would be undesirable for any traveller to have an itinerary with more than 4 stops. Constructing a feasible solution for this problem is simple since any star graph whose diameter is equal to 2 would trivially satisfy any diameter constraint. Hence we chose to pick the star graph with maximum algebraic connectivity ($\lambda_2^{initial}$) as an initial feasible solution whose computational complexity would be $O(n)$. A sample initial feasible solution is shown in part (b) of Figure 6.

We shall now compare the solution quality of heuristics with respect to the optimal solution (λ_2^*) obtained from the proposed algorithm 1. We define solution quality as $\frac{\lambda_2^* - \lambda_2^{heuristic}}{\lambda_2^*} * 100$ where $\lambda_2^{heuristic}$ holds for either 2-opt or tabu search heuristic. The results shown in Table I are for 10 random instances generated for networks with 8 airports. Based on the results in Table I, we observed that the average run time to obtain optimal solution was around 370.89 seconds. Also it can be seen that both 2-opt and tabu search heuristics' solution qualities were very good and gave optimal solutions for these 10 random instances. Instance 1 of Table I is pictorially shown in Figure 6.

Scalability of 2-opt and tabu search heuristics: We also compared the performance of proposed heuristics for larger instances with respect to the initial feasible solution since we could not compute optimal solutions for larger instances. In Table II, all the values were averaged over 10 random instances for each size of the problem. We define % gap as $\frac{\lambda_2^{heuristic} - \lambda_2^{initial}}{\lambda_2^{initial}} * 100$. Based on this definition, higher % gap implies that the heuristic solution is better. Hence from Table II, we observed that, though 2-opt heuristic performed better

for smaller instances, tabu search performed better than 2-opt for larger instances except a few instances. 2-opt and tabu search heuristic solutions for a 25 nodes problem is shown in Figure 7 where tabu search solution quality is better than that of 2-opt.

TABLE I: Comparison of solution quality of the heuristics with the optimal solution obtained from the proposed algorithm for $D \leq 4$ of networks with 8 airports. λ_2^* is the optimal algebraic connectivity.

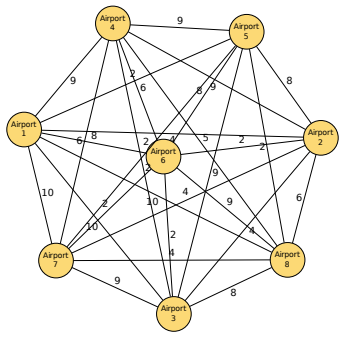
No.	Optimal solution		2-opt search		Tabu search	
	λ_2^*	Time (sec)	λ_2^{2opt}	Solution quality	λ_2^{tabu}	Solution quality
1	3.9712	180.57	3.9712	0.00	3.9712	0.00
2	4.3101	408.10	4.3101	0.00	4.3101	0.00
3	3.9297	621.85	3.9297	0.00	3.9297	0.00
4	3.5275	216.79	3.5275	0.00	3.5275	0.00
5	3.8753	470.63	3.8753	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 2-opt and tabu search heuristic solutions for various problem sizes.

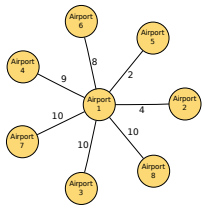
n	Average $\lambda_2^{initial}$	2-opt search		Tabu search	
		Average λ_2^{2opt}	Average % gap	Average λ_2^{tabu}	Average % gap
5	1.15	1.38	19.38	1.15	0.00
7	2.87	4.07	48.44	4.07	48.44
8	2.39	3.87	65.43	3.87	65.43
9	2.46	3.91	63.77	3.89	62.93
10	2.47	3.89	63.01	3.89	63.01
15	1.79	4.92	211.26	5.11	225.67
20	1.98	6.32	232.70	6.73	253.40
25	2.27	9.62	331.20	9.95	346.79

VII. Conclusions

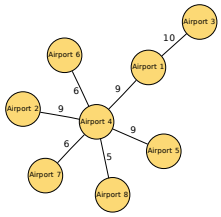
We study three air transportation network design methods to maximize the network robustness and keep every itinerary length fewer than D legs. Algebraic connectivity is used to measure the robustness of the network. The authors formulate the problem as a mixed-integer, semi-definite program and provide an algorithm to find optimal solutions based on cutting plane and bisection methods. The algebraic connectivity of a network is posed using a semi-definite constraint and the diameter of the graph is formulated using a multi-commodity flow formulation. Our second method, tabu search constructs the dynamic searching neighbor for each evolving solution, establishes the tabu list and aspiration criteria, which seeks a potentially global solution and the optimal robust design. In third method, 2-opt exchange searches for a better feasible solution by performing an exchange on every pair of edges on an initial feasible solution. Computational results are provided to evaluate the performance of the proposed algorithms and we empirically observed that tabu search method performed better than 2-opt search for larger instances.



(a) Complete network



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



(c) λ_2^*,λ_2^{tabu},λ_2^{2opt} = 3.9712

Fig. 6: Complete network with all possible routes connecting 8 airports with distinct number of flight routes representing the edge weights are shown in (a). (b) represents the initial feasible solution which is the star graph used in both the heuristics. (c) represents the optimal network which also happens to be the solutions of 2-opt and Tabu search methods. Maximum number of itinerary legs were taken to be 4 ($D \leq 4$).

References

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

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

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

[4] ICAO, *Procedures for Air Navigation Services - Rules of the air and air traffic services*, International Civil Aviation Organization, 2010, doc 4444-RAC/501.

[5] 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.

[6] FAA, *FAA Aerospace Forecasts FY 2008-2025*, Federal Aviation Administration, December 2010.

[7] A. Gibbons, *Algorithmic Graph Theory*. Cambridge University Press, July 1985.

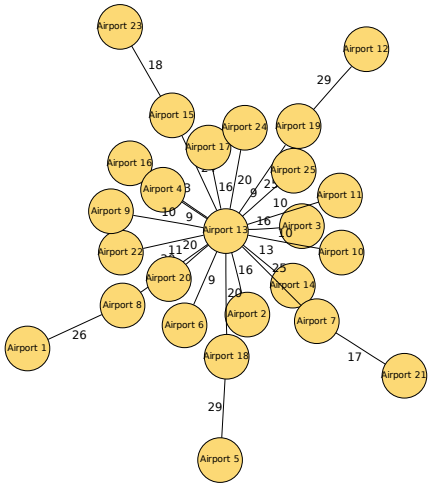
[8] M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak Mathematics Journal*, vol. 23, pp. 298–305, 1973.

[9] J. Huang and Y. Dang, "Research on the complexity of weighted air transportation network of express enterprise," in *The 1st International Conference on Information Science and Engineering (ICISE 2009)*, Dec 2009, pp. 5077–5080.

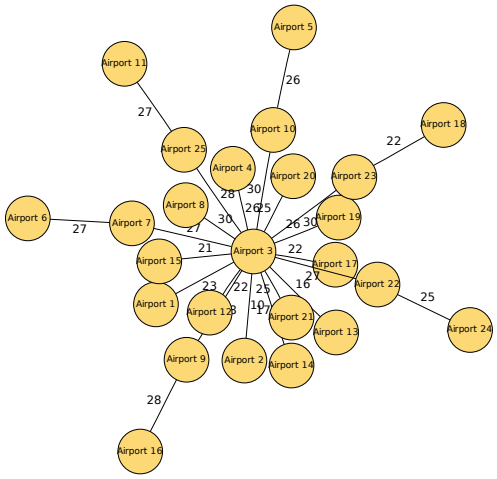
[10] P. Wei and D. Sun, "Weighted algebraic connectivity: An application to airport transportation network," in *the 18th IFAC World Congress*, Milan, Italy, Sep 2011.

[11] B. Mohar, "The Laplacian spectrum of graphs," *Graph Theory, Combinatorics, and Applications*, vol. 2, pp. 871–898, 1991.

[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.



(a) 2-opt exchange, $\lambda_2^{2opt} = 8.3411$



(b) Tabu search, $\lambda_2^{tabu} = 10.0197$

Fig. 7: 2-opt exchange and Tabu search solutions for a network with 25 airports where maximum number of itinerary legs were taken to be 4 ($D \leq 4$). Since $\lambda_2^{tabu} > \lambda_2^{2opt}$, the feasible network corresponding to the Tabu search method is more robust against the failure of functionality of any airport.

[13] 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.

[14] 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.

[15] 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

[16] H. Nagarajan, S. Rathinam, S. Darbha, and K. Rajagopal, "Synthesizing robust communication networks for uavs," in *American Control Conference, 2012. ACC'12*. IEEE, 2012.