Algebraic Connectivity Maximization for Air Transportation Networks

Peng Wei, *Member, IEEE*, Gregoire Spiers, and Dengfeng Sun, *Member, IEEE*

***Abstract*—It is necessary to design a robust air transportation network. An experiment based on the real air transportation network is performed to show that algebraic connectivity is a fair measure for network robustness under random failures. Therefor, the goal of this paper is to maximize algebraic connectivity. Some researchers solve the maximization of the algebraic connectivity by choosing the weights for the edges in the graph. Others focus on the best way to add edges in a network in order to optimize the connectivity. In this paper, the authors formulate a new air transportation network model and show that the corresponding algebraic connectivity optimization problem is interesting because the two subproblems of adding edges and choosing edge weights cannot be treated separately. The new problem is formulated and exactly solved in a small air transportation network case. The authors also propose the approximation algorithm in order to achieve better efficiency. For large networks, the semidefinite programming with cluster decomposition is first presented. Moreover, the algebraic connectivity maximization for directed networks is discussed. Simulations are performed for a small-scale case, large-scale problem, and directed network problem.**

***Index Terms*—Author, please supply index terms/keywords for your paper. To download the IEEE Taxonomy go to http://www.ieee.org/documents/Taxonomy_v101.pdf.** AQ1

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 (link) set E represents all the m direct flight routes between airports. 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} if there is a direct route between airports v_i and v_j. There are many factors to be considered when designing an air transportation network, such as traffic demand, operating cost, airport hubs, market competition, multiairport systems, and scheduling [5]–[14]. In this paper, we focus on investigating the network robustness maximization, particularly the algebraic connectivity maximization.

Manuscript received November 6, 2012; revised April 5, 2013 and August 20, 2013; accepted October 1, 2013. The Associate Editor for this paper was J.-P. B. Clarke.

P. Wei is with the Operations Research and Advanced Analytics, American Airlines, Fort Worth, TX 76155 USA (e-mail: peng.wei@aa.com).

G. Spiers was with the Department of Applied Mathematics, École Polytechnique, 91120 Palaiseau, France. He is now with Amadeus S.A.S., 06902 Sophia Antipolis, France (e-mail: gregoire.spiers@polytechnique.org). AQ2

D. Sun is with the School of Aeronautics and Astronautics, Purdue University, West Lafayette, IN 47907 USA (e-mail: dsun@purdue.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITS.2013.2284913

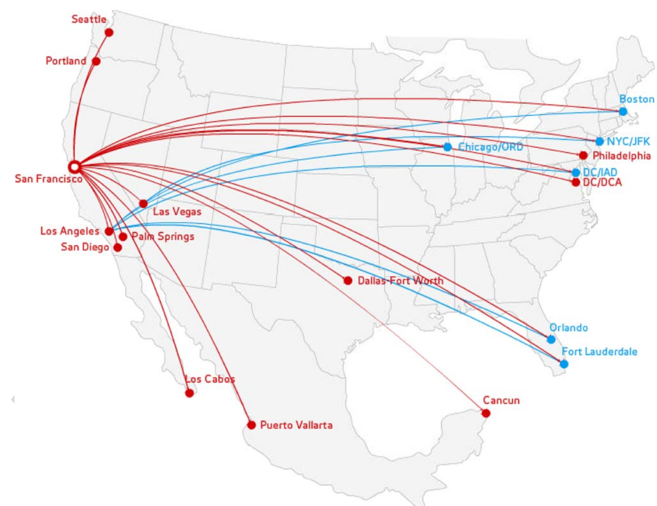


Fig. 1. Air transportation network route map for Virgin America Airlines.

A. *Algebraic Connectivity and Air Transportation Network Robustness*

In order to illustrate the relationship between the algebraic connectivity and air transportation network robustness, a real air transportation network of Virgin America is studied. The following experiment shows that algebraic connectivity is a fair measurement for the network robustness with regard to random link failures under the current Virgin America network topology.

According to the current route map of Virgin America in Fig. 1, we consider the 16 airports in the United States and obtain the adjacency matrix as Table I. The 16 United States airports include Boston (BOS), New York City/John F. Kennedy (JFK), Philadelphia, Washington Dulles International Airport (DC/IAD), Ronald Reagan Washington National Airport (DC/DCA), Chicago O'Hare International Airport (ORD), Orlando, Fort Lauderdale, Dallas/Fort Worth, Seattle, Portland, San Francisco, Los Angeles, Las Vegas, San Diego, and Palm Springs and are indexed as numbers 1–16. San Francisco International Airport (SFO) and Los Angeles International Airport (LAX) are two major hubs of the entire network. Both have at least one direct flight to almost all the other airports.

In order to show how well the algebraic connectivity can measure the robustness of an air transportation network, we

TABLE I
ADJACENCY MATRIX CONSISTS OF 16 VIRGIN AMERICA AIRLINES AIRPORTS IN THE USA

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0
3	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
12	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
13	1	1	1	1	0	1	1	1	1	1	1	1	0	0	0	0
14	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

created six different *weighted air transportation networks* with the same topology in Table I by randomly assigning one of the three types of *link weights* to each route. Each link weight is an indication of link strength. A larger weight represents a stronger link and a smaller weight shows that the corresponding link is easier to fail. There are many reasons for route failure, such as weather disturbance, long ground delay program, long airspace flow program (AFP), aircraft mechanical problem, and upline flight delay/cancelation. The route failure rate statistics are published by each origin–destination pair (route) and different routes have different features [15]. For example, the route failure rate between JFK and BOS during summer is higher than that between SFO and LAX because of the crowded northeastern airspace (AFP is more frequent) and more summer thunderstorms. Another example is that a shorter route is easier to fail than a longer transcontinental route because: 1) airlines usually put larger aircraft on transcontinental route, and these aircraft are more robust to weather disturbance and 2) airlines are more likely to cancel shorter route flights because the flight frequency on a shorter route is higher; therefor, the passengers on the canceled flight are easier to protect (be reaccommodated to later flights). In summary, each route has its own features and thus in our model we consider that they have different possibilities for failure.

The three types of link weights are mapped to different link failure probabilities (see Table II). The link failure probability range [%0, %5] is obtained from the historical flight cancelation rate between September 15, 2012, and November 15, 2012 [15].

A network failure is defined as the existence of at least one pair of nodes that cannot access each other through any one or multiple links. For each one of the six weighted networks, 1000 trials are performed. In each trial, every link fails randomly according to the failure probabilities listed in Table II. The total number of network failures is counted in 1000 random trials. The results are shown in Table III with algebraic connectivity sorted in ascending order.

TABLE II
MAPPING BETWEEN LINK WEIGHTS AND LINK FAILURE PROBABILITIES

link weight w_{ij}	1	2	3
link failure probability	5%	3%	1%

TABLE III
NETWORK FAILURE NUMBERS WITH DIFFERENT ALGEBRAIC CONNECTIVITY VALUES

algebraic connectivity	total failures in 10000 trials
1.0306	1113
1.7586	991
1.8661	763
1.9711	571
2.3128	423
2.7393	355

We can see that with higher algebraic connectivity, the network is more robust and has fewer network failures. With lower weighted algebraic connectivity, the network is easier to break down. Therefor, algebraic connectivity is a fair robustness measure for the air transportation network, and we need to find the maximized algebraic connectivity.

The air traffic demand is expected to continue its rapid growth in the future. The Federal Aviation Administration estimated that the number of passengers is projected to increase by an average of 3% every year until 2025 [16]. The expanding traffic demand on the current air transportation networks of different airlines will cause more and more flight cancelations with the limited airport and airspace capacities. As a result, more robust air transportation networks are desired to sustain the increasing traffic demand for each airline and for the entire National Airspace System (NAS). This is the major motivation of this paper.

B. *Related Work*

An air transportation network and its robustness have been studied over the last several years. Guimera and Amaral first studied the scale-free graphical model of the air transportation network [1]. Conway showed that it was better to describe the national air transportation system or the commercial air carrier transportation network as a system of systems [17]. Bonnefoy showed that the air transportation network was scale free with aggregating multiple airport nodes into meganodes [18]. Alexandrov defined that on-demand transportation networks would require robustness in system performance [19]. The robustness of an on-demand network would depend on the tolerance of the network to variability in temporal and spatial dynamics of weather, equipment, facility, crew positioning, etc. Kotegawa *et al.* surveyed different metrics for air transportation network robustness, including betweenness, degree, centrality, connectivity, etc. [20]. They selected clustering coefficient and eigenvector centrality as the network robustness metrics in their machine learning approach. Bigdeli *et al.* compared algebraic connectivity, network criticality, average degree, average node betweenness, and other metrics [21]. Jamakovic *et al.* found that algebraic connectivity was an important metric in the analysis of various robustness problems in several typical network models [22], [23]. Byrne *et al.* showed that algebraic connectivity was the efficient measure for the robustness of both small and large networks [24]. Vargo *et al.* in [3] chose algebraic connectivity as the robustness metric and built the optimization problem solved by the edge swapping-based tabu search algorithm.

In this paper, we measure the robustness of air transportation network by computing the algebraic connectivity, which is usually considered as one of the most reasonable and efficient evaluation methods [24], [25]. Although the maximized value of algebraic connectivity is abstract, the optimized air transportation network structure and weighting assignment provide us the applicable design.

There are some literature on algebraic connectivity maximization. The problems studied can be divided into two categories, namely, the edge addition problem and the variable weights problem.

1) **Edge addition problem:** The goal is to add or remove a given number of k edges on a graph in order to get the best algebraic connectivity. The edges to be added or removed are selected from a candidate set. The algorithms that have been developed to solve the problem include tabu search [25], greedy algorithms [25], [26], and rounded semidefinite programming (SDP) [26].
2) **Variable weights problem:** The edges of the graph are fixed and the goal is to determine the edge weights in order to maximize the algebraic connectivity. This is a convex optimization problem that is often solved by using an SDP formulation [27]–[29] or a subgradient algorithm [30].

C. *Contribution*

The major contribution of this paper compared with what has been studied is that we find that, in order to maximize the algebraic connectivity, the edge addition problem and the variable weights problem cannot be studied separately. Solving one of them independently will only result in a suboptimal solution. Therefor, we propose a new algorithm to solve both problems at the same time. How to choose the edges of the graph is demonstrated, as well as how to assign their weights. In addition, we are the first to present the cluster decomposition method to achieve better computation efficiency for large-scale networks. We are also the first to discuss the algebraic connectivity maximization for directed air transportation networks.

The rest of this paper is structured as follows. Section II shows why this problem naturally arises in air transportation networks and how it can be formulated. In Section III, the problem is exactly solved for small networks, and the fact that the two problems are not independent is highlighted. Then, the authors present the SDP formulation and the more efficient full algorithm, which includes relaxed SDP, and solution rounding is proposed. In Section IV, the problem for large networks is solved, the computational efficiency is analyzed, and the numerical results are provided. The algebraic connectivity optimization for directed air transportation networks is presented in Section V. Section VI concludes this paper.

II. Problem Formulation

A graph G with n nodes and m edges is used to define an air transportation network. Let $A = (a_{ij})$ be the adjacency matrix of G. The Laplacian matrix $L = (l_{ij})$ of G is defined by

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

The eigenvalues of L are sorted $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$. L is a semidefinite positive matrix; thus, for all i, $\lambda_i \geq 0$. It is also known that $\lambda_1 = 0$ since $Le = 0$ with $e = (1, \ldots, 1)$ [31].

Definition: The second smallest eigenvalue $\lambda_2(L)$ is the algebraic connectivity of G.

Now, recall the two key properties of the algebraic connectivity that will be used in this paper.

Property 1: Let $e = (1, \ldots, 1) \in \mathbb{R}^n$ and

$$\Omega = \left\{ x \in \mathbb{R}^n | \|x\| = 1, \quad e^T x = 0 \right\}.$$

The Courant–Fischer principle [32] states that

$$\lambda_2 = \min_{x \in \Omega} x^T L x. \tag{1}$$

Property 2: Function $w \to \lambda_2(w)$ is concave with w denoting the edge weight vector. This can be proven by seeing that $\lambda_2(w)$ is the pointwise infimum of a family of linear functions of w (see [27])

$$\lambda_2(w) = \inf_{\|v\|=1, e^T v = 0} v^T L v,$$

$$\lambda_2(w) = \inf_{\|v\|=1, e^T v = 0} \sum_{(i,j) \in E} w_{ij} (v_i - v_j)^2.$$

The goal of this paper is to maximize the algebraic connectivity of the network under several constraints.

There are $m = (n(n-1))/2$ edges in the complete symmetric graph. Each has a weight w_{ij} representing the link strength, as described in Section I. The following constraints are considered.

The edge weight representing link strength must be within the range between the lower bound α and the upper bound β

$$\forall (i,j) \in E, \quad \alpha \leq w_{ij} \leq \beta.$$

When there is no edge connecting v_i and v_j, the corresponding $w_{ij} = 0$.

There exists an operating cost c_{ij} for each link. In a real air transportation network, the cost for a route contains the fuel cost, aircraft maintenance cost, crew/flight attendant labor cost, cost for arrival/departure slots at runways, cost for gates at origin/destination airports, and cost for flying through airspace (international flights). In this paper, we use one link cost to represent the integrated operating cost. The operating cost is higher for a stronger link for several practical reasons. For example, we know that the most effective way to avoid a mechanical problem cancelation is to have spare parts or even a spare aircraft. Similarly, the most effective way to avoid a cancelation caused by crew legality or crew scheduling is to have enough standby crew. Both approaches can increase link strength; at the same time, they introduce higher costs. As for weather disturbances, to load extra fuel will give an aircraft more flight plan options with which it can be rerouted to avoid weather problems and prevent the cancelation. However, extra fuel also introduces higher cost. In addition, larger aircraft are more robust to weather disturbances. Nevertheless, to operate a larger aircraft costs more because of more fuel needed, more flight attendants, and even more crew (for international flights). Therefor, in this paper, we consider the linear cost for link strength. The total operating cost budget for all the links is limited by

$$\sum_{ij} w_{ij} c_{ij} \leq C.$$

In summary, the complete problem that the authors aim at solving is

$$\max_{w} \lambda_2 \left(L(w)\right) \quad \text{s.t.} \begin{cases} \sum_{ij} w_{ij} c_{ij} \leq C \\ w_{ij} \in \{0, [\alpha, \beta]\} \end{cases}. \tag{P}$$

A. Alternative Formulation

In order to be able to solve the problem, the authors need to reformulate it by adding decision variables. The idea is to add, for each edge (i, j), a binary variable x_{ij} stating if there exists an edge between v_i and v_j

$$x_{ij} = 1 \Leftrightarrow w_{ij} \neq 0.$$

This is useful since now the domain of w can be expressed as

$$\forall (i,j), \quad \alpha x_{ij} \leq w_{ij} \leq \beta x_{ij}.$$

The problem now becomes

$$\max_{x,w} \lambda_2 \left(L(w)\right) \quad \text{s.t.} : \begin{cases} x_{ij} \in \{0,1\} \\ \sum_{ij} w_{ij} c_{ij} \leq C \\ \alpha x_{ij} \leq w_{ij} \leq \beta x_{ij}. \end{cases}$$

Then, variable k is added, which determines the number of edges in the graph. The final formulation of the problem is

$$\max_{x,w,k} \lambda_2 \left(L(w)\right) \quad \text{s.t.} : \begin{cases} \sum_{i,j} x_{ij} = k \\ x_{ij} \in \{0,1\} \\ \sum_{i,j} w_{ij} c_{ij} \leq C \\ \alpha x_{ij} \leq w_{ij} \leq \beta x_{ij}. \end{cases} \tag{2}$$

B. Difficulty

An important remark is that the problem cannot really be split into two steps of first deciding whether $w = 0$ or not and followed by choosing the appropriate weights. This is due to the fact that there are lower bound and upper bound constraints on w. However, assuming that the two steps are independent, a decoupled approach can be tried first. The first step is to choose edges for the empty graph that corresponds to the edge addition problem introduced in [25]

$$\max_{x} \lambda_2(L(x))$$
$$\text{s.t.} : \sum_i x_i = k, \quad x_i \in \{0,1\}, \quad \sum_i x_i c_i \leq \frac{C}{\alpha}$$

and the second step is to choose the weights on them

$$\max_{w} \lambda_2 \left(L(w)\right)$$
$$\text{s.t.} : \sum_i w_i c_i \leq C, \quad \alpha y_i \leq w_i \leq \beta y_i, \quad y = x_{\text{opt}}.$$

Later, it will be seen that, if this approach is used, the result will not be optimal.

C. Relaxation

The relaxation (R) of the problem is obtained by allowing noninteger values for x

$$\forall (i,j) \in E, \quad x_{ij} \in [0,1].$$

This is the same as choosing $w \in [0, \beta]$ without variables x and k. However, these variables will be necessary in order to be able to get the integer solution from this relaxed one. It is noticed that the solution of (R) is a concave function of k.

D. Concavity in k

The first important property is that the solution of (R) is concave in k, which will be used in the golden section search algorithm in Algorithm 1. More precisely, Λ is defined such that

$$\Lambda(k) = \max_{x,w} \lambda_2 \left(L(w)\right) \quad \text{s.t.} \begin{cases} \sum_i x_i = k \\ x_i \in [0,1] \\ \sum_i w_i c_i \leq C \\ \alpha x_i \leq w_i \leq \beta x_i. \end{cases}$$

Property 3: $\Lambda(k)$ is a concave function.

This property is important since it shows that the maximization of the algebraic connectivity is related to the number of edges in the graph. By solving the problem for very few values of k, a good knowledge on k_{opt} can be obtained.

Proof: Consider k_1 and k_2 in $\mathbb{R}$ and $\gamma \in [0,1]$ such that $\Lambda(k_i) > 0$ for $i = 1, 2$. The idea is to use the fact that $w \to \lambda_2(L(w))$ is concave (please see Property 2).

$$
\begin{aligned}
&\Lambda\left(\gamma k_1 + (1-\gamma)k_2\right)\\
&= \max_{x,w} \lambda_2 \quad \text{s.t.} \begin{cases} \sum_i x_i = \gamma k_1 + (1-\gamma)k_2 \\ x_i \in [0,1] \\ \sum_i w_i c_i \le C \\ \alpha x_i \le w_i \le \beta x_i \end{cases}\\
&\ge \max_{x,w} \lambda_2 \quad \text{s.t.} \begin{cases} \sum_i x_i^{(1)} = k_1, & \sum_i x_i^{(2)} = k_2 \\ x_i^{(j)} \in [0,1], & j = 1,2 \\ x = \gamma x^{(1)} + (1-\gamma)x^{(2)} \\ \sum_i w_i c_i \le C \\ \alpha x_i \le w_i \le \beta x_i \end{cases}\\
&\ge \max_{x,w} \lambda_2 \quad \text{s.t.} \begin{cases} \sum_i x_i^{(1)} = k_1, & \sum_i x_i^{(2)} = k_2 \\ x_i^{(j)} \in [0,1], & j = 1,2 \\ w = \gamma w^{(1)} + (1-\gamma)w^{(2)} \\ \sum_i w_i c_i \le C \\ \alpha x_i^{(j)} \le w_i^{(j)} \le \beta x_i^{(j)} \end{cases}\\
&\ge \gamma \max_{x,w} \lambda_2 \quad \text{s.t.} \begin{cases} \sum_i x_i = k_1 \\ x_i \in [0,1] \\ \sum_i w_i c_i \le C \\ \alpha x_i \le w_i \le \beta x_i \end{cases}\\
&\quad + (1-\gamma) \max_{x,w} \lambda_2 \quad \text{s.t.} \begin{cases} \sum_i x_i = k_2 \\ x_i \in [0,1] \\ \sum_i w_i c_i \le C \\ \alpha x_i \le w_i \le \beta x_i \end{cases}\\
&\ge \gamma \Lambda(k_1) + (1-\gamma)\Lambda(k_2)
\end{aligned}
$$

which proves that Λ is concave in k. ■

III. Small-Scale Air Transportation Networks

A. *Exact Solution for Small Networks*

If all weights have to be chosen within an interval, the problem becomes a convex optimization problem and it can be solved using an SDP solver. The idea here is to try all the possible configurations for which all the weights are either 0 or in $[\alpha, \beta]$. Then, each configuration can be independently optimized and the one that leads to the best result can be found.

Consider that n nodes are chosen randomly. There are $m = (n(n-1))/2$ edges and 2^m configurations to test. For each configuration, if Y is the set of the edges that are actually in the graph, the following problem needs to be solved:

$$
\max_w \lambda_2\left(L(w)\right) \quad \text{s.t.} \begin{cases} \sum_{ij} w_{ij} c_{ij} \le C \\ \alpha \le w_{ij} \le \beta, & \forall (i,j) \in Y \\ w_{ij} = 0, & \forall (i,j) \notin Y. \end{cases}
$$

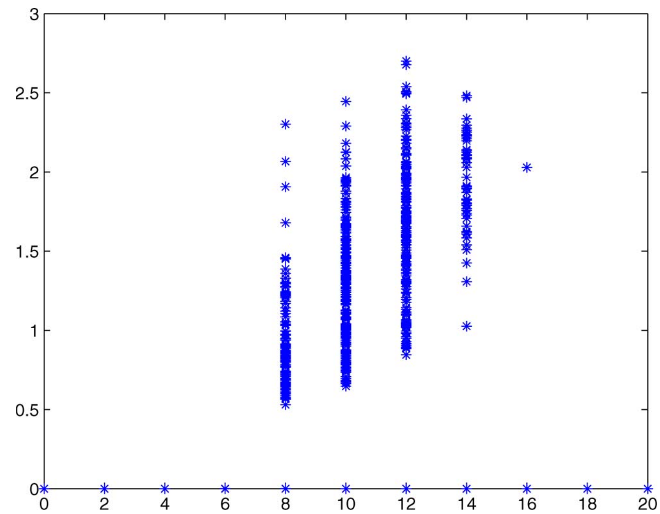


Fig. 2. Results of (k, λ_2) for all the configurations of $n = 5$ and $C = 6.5$.

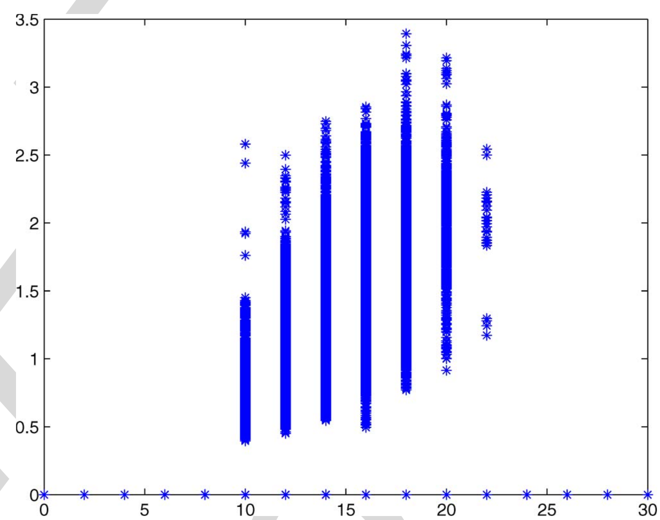


Fig. 3. Results of (k, λ_2) for all the configurations of $n = 6$ and $C = 8$.

This can be done by solving the SDP corresponding to the weight optimization problem (see [27] for details)

$$
\min_w \sum_i w_i c_i \quad \text{s.t.} \begin{cases} \alpha \le w_i \le \beta, & \forall (i,j) \in Y \\ w_{ij} = 0, & \forall (i,j) \notin Y \\ L(w) \succeq I - \frac{1}{n} e e^T. \end{cases}
$$

It becomes impossible to exactly solve the problem when n is large. Therefor, the authors assume n to be small in this section.

1) Small-Scale Exact Solution Results: For each configuration, the number of edges k in the graph is computed. The results of (k, λ_2) are plotted in Figs. 2 and 3 for two different networks.

It is noticed that the best connectivity is not reached at the maximum number of edges; therefor, the choice of the edges and the choice of the weights are not independent.

It is also noticed that, unlike the continuous case, the discrete shape given by $k \to \Lambda(k)$ in Figs. 2 and 3 is not exactly concave. However, it has almost the shape of a concave function; hence, the authors will be able to consider it concave in the approximate case later on.

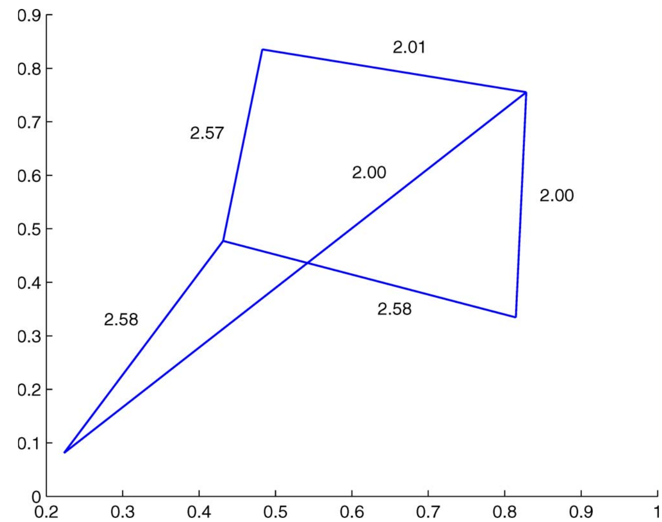


Fig. 4. Optimal network for $n = 5$ with the weights. $p = 0$, $\alpha = 2$, and $C = 6.5$.

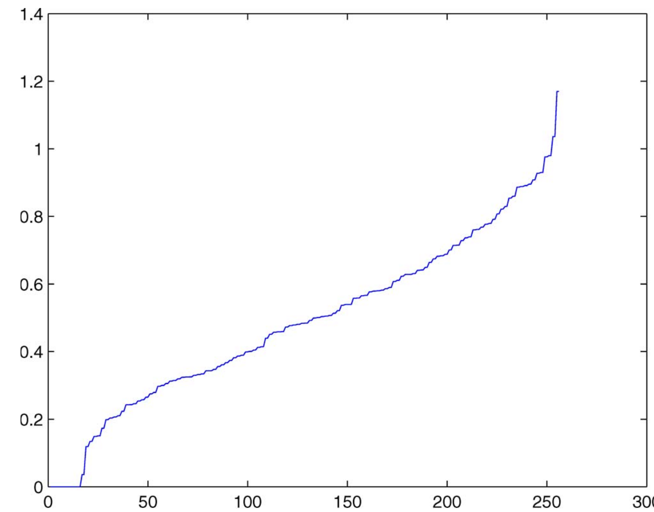


Fig. 5. Function g for random points in $[0, 1]^2$.

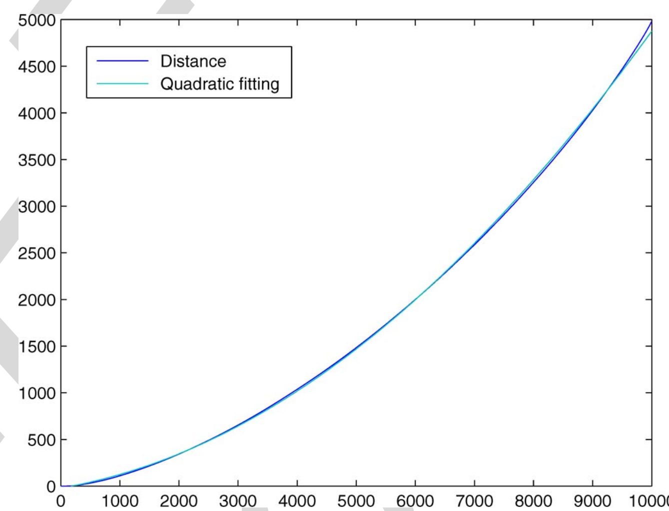


Fig. 6. C as a function of k_{lim} with the quadratic fitting.

The exact solution for a network of size $n = 5$ is shown in Fig. 4. As it is impossible to exactly solve this problem for networks with a larger size, an algorithm is going to be designed, which solves it approximately. The main idea is to use the quasi-concave shape of function $\Lambda(k)$.

For the practical problem with a larger size, the first step is to choose a value for k. The authors are able to solve the relaxed version (R) of the problem where x is a noninteger variable. Then, the result can be rounded to obtain a feasible solution for the original problem (P).

2) Maximum Number of Edges: Because of the minimum value α for the weights, there exists a limit in the number of edges in the graph. Here, the maximal number of edges k_{lim} needs to be found. Regardless of the performance of the network, edges are added until the operating cost constraint is reached. Consider that $w = \alpha x$, which is the minimum weight, and the problem is to solve a trivial form of the knapsack problem

$$\max_{x \in \{0,1\}} \sum_i x_i$$
$$\text{s.t. } \sum_i x_i c_i \leq \frac{C}{\alpha}.$$

Indeed, if k_{lim} is the solution of this problem, it can be guaranteed that there will not be any solution of $k > k_{\mathrm{lim}}$.

When the cost c_i is sorted by increasing order, it can be obtained that

$$\sum_{s=1}^{k_{\mathrm{lim}}} c_s \approx \frac{C}{\alpha}$$

which, for large-enough n, can be approximated by the following formula:

$$\int_1^{k_{\mathrm{lim}}} g(s)ds = \frac{C}{\alpha}$$

where

$$\forall s \in [1, k_{\mathrm{lim}}], \quad g(s) = c_{\lfloor s \rfloor}.$$

If the n nodes are randomly chosen in a square, function g is very close to a linear function (except at the very beginning and at the very end). This can be verified in Fig. 5. Using this information, it can be obtained that

$$a_2 k_{\mathrm{lim}}^2 + a_1 k_{\mathrm{lim}} + a_0 = \frac{C}{\alpha} \tag{3}$$

where a_0, a_1, and a_2 are constant parameters. Finally

$$k_{\mathrm{lim}} = \frac{-a_1 + \sqrt{{a_1}^2 - 4a_2\left(a_0 - \frac{C}{\alpha}\right)}}{2a_2}. \tag{4}$$

It is shown in Fig. 6 that the quadratic result obtained by (4) is a very good approximation.

B. SDP Formulation

To solve the larger size problem, the relaxation of the problem is expressed as an SDP that will be solved efficiently. When v is not normalized, recall (1) in Property 1, which can be then transformed as

$$\begin{cases} \lambda_2 = \max_\lambda \lambda \\ \lambda v^T v \le v^T L v \\ \forall v \in \mathbb{R}^n, \qquad v^T e = 0. \end{cases}$$

Variable μ is added, which allows any $v \in \mathbb{R}^n$

$$\begin{cases} \lambda_2 = \max_{\lambda,\mu} \lambda \\ \forall v \in \mathbb{R}^n, \qquad v^T(\mu e e^T)v + v^T L v - \lambda v^T v \ge 0. \end{cases}$$

It can be written using Loewner's order [33]

$$\begin{cases} \lambda_2 = \max_{\lambda,\mu} \lambda \\ \mu e e^T + L - \lambda I \succeq 0. \end{cases} \tag{5}$$

The relaxation of the problem that needs to be solved is

$$\max_{x,w,k} \lambda_2(L(w)) \quad \text{s.t.} \begin{cases} \sum_i x_i = k \\ x_i \in [0,1] \\ \sum_i w_i c_i \le C \\ \alpha x_i \le w_i \le \beta x_i \end{cases}$$

which now becomes with (5)

$$\max_{x,w,k,\lambda,\mu} \lambda \quad \text{s.t.} \begin{cases} \sum_i x_i = k \\ x_i \in [0,1] \\ \sum_i w_i c_i \le C \\ \alpha x_i \le w_i \le \beta x_i \\ \mu e e^T + L - \lambda I \succeq 0. \end{cases} \tag{6}$$

This problem is an SDP since there is a semidefinite constraint and all the other constraints are linear. It can be solved efficiently by an SDP solver. SeDuMi [34] is used in this paper.

1) Optimality Conditions: The primal SDP is

$$\max_{x,w,k,\lambda,\mu} \lambda \quad \text{s.t.} \begin{cases} \sum_i x_i = k \\ x_i \in [0,1] \\ \sum_i w_i c_i \le C \\ \alpha x_i \le w_i \le \beta x_i \\ \mu e e^T + L - \lambda I \succeq 0. \end{cases}$$

The variables are rescaled by dividing w, k, and x by λ and C. The dual SDP problem is

$$\min_{x,w,k,\lambda} \sum_i w_i c_i \quad \text{s.t.} \begin{cases} \sum_i x_i = k \\ x_i \in [0,1] \\ \alpha x_i \le w_i \le \beta x_i \\ L \succeq I - \frac{1}{n} J \end{cases}$$

where J is the all-one matrix. When x_{ij} is relaxed (please see Section II-C), the relaxed dual SDP formulation is

$$\min_{x,w,k,\lambda} \sum_i w_i c_i \quad \text{s.t.} \begin{cases} 0 \le w_i \le \beta \\ L \succeq I - \frac{1}{n} J. \end{cases}$$

X is the matrix of the operating costs. The matrix format relaxed dual SDP is therefor

$$\max_X \left\langle I - \frac{1}{n} J, X \right\rangle \quad \text{s.t.} \begin{cases} X \succeq 0 \\ \langle E, X \rangle = c_{ij} \end{cases}$$

where E is the matrix with $E_{ii} = E_{jj} = 1$ and $E_{ij} = E_{ji} = -1$.

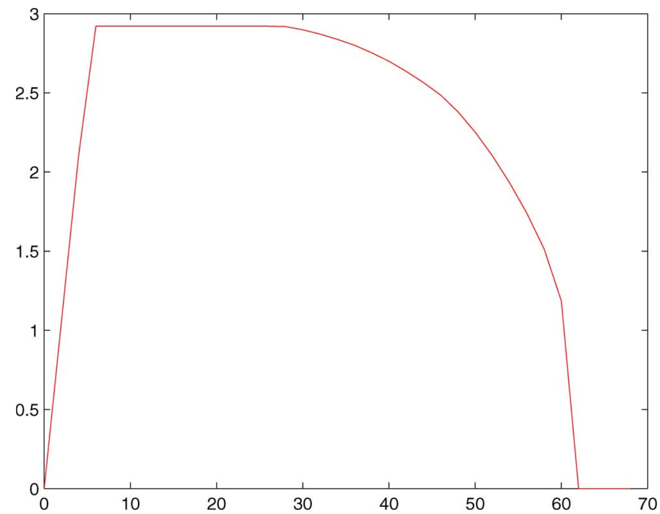


Fig. 7. Upper bound: $\lambda_2 = f(k)$.

The Karush–Kuhn–Tucker optimality conditions are

$$\begin{cases} SX = XS = 0 \\ S \succeq 0, \qquad X \succeq 0 \\ \langle E, X \rangle = c_{ij} \\ L - I + \frac{1}{n} J = S. \end{cases}$$

If $w_i = w$ for all i, it can be obtained that

$$S = (nw - 1)\left(I - \frac{1}{n} J\right).$$

When $w = 1/n$, $S = 0$ and the conditions are satisfied.

Reciprocally, if the optimality conditions are satisfied, it can be obtained that

$$\left(L - \left(I - \frac{1}{n} J\right)\right) X = 0.$$

X has rank n; hence

$$L = I - \frac{1}{n} J.$$

Thus, w is constant for all i and $w = 1/n$.

For edge i and edge j, the optimality condition is finally obtained

$$\begin{cases} \forall (i,j), w_i = w_j \\ \sum_i w_i c_i = C. \end{cases}$$

2) Upper Bound: With the SDP formulation, the relaxation can now be solved. When the values of the optimal connectivity for different values of k are computed, the upper bound is plotted in Fig. 7.

The relaxed problem reached its maximum for several values of k contained in an interval $[k_{\min}, k_{\max}]$. Indeed, the optimality conditions give

$$\begin{cases} \forall (i,j), w_i = w_j \\ \sum_{i=1}^m w_i c_i = C. \end{cases}$$

All the weights are equal and their value is $\forall i,\ w_i = (C/(\sum_j c_j)) = \Omega$. If $w = \beta x$, all the elements of x are equal and their value is $\forall i,\ x_i = (k/(n^2 - n))$, which leads to

$$k_{\min} = \frac{\Omega(n^2 - n)}{\beta}.$$

By doing the same computation, it is proved that the optimal value is also reached with

$$k_{\max} = \frac{\Omega(n^2 - n)}{\alpha}$$

and $\forall k \in [k_{\min}, k_{\max}]$, the optimal value is achieved.

However, it needs to be pointed out that when the solution is rounded, infeasible solution may appear. For example, if $k = k_{\min}$, there is often no solution since

$$\sum_i x_i = \frac{n\Omega}{\beta} < n - 1$$

if $(\Omega/\beta) \ll 1$, which is often the case. In addition, because at least $n - 1$ edges are needed to connect an n node graph, there is no positive solution. Therefor, the upper bound is not a very good bound for small values of k.

C. Rounding Techniques for SDP Solution

1) Description of the Methods: In this section, suppose that the relaxed optimal solution s_0 has been found. k edges are going to be selected from s_0, which means that $x_i = 1$ for k values and $x_i = 0$ for the others. There are several ways to do so. The methods that have been studied and implemented are listed.

1) *Greedy:* Choose the k biggest elements $s_0(x_i)$ in the relaxed solution. Then, find the optimal weights by solving the corresponding SDP.
2) *Random fast:* Randomly choose the rounding. For each $i \in \{1, \ldots, m\}$, $x_i = 1$ with probability $s_0(x_i)$ and $x_i = 0$ otherwise. Then, the weights are affected with the following value:

$$\frac{s_0(w_i)}{s_0(x_i)}.$$

These two steps are repeated many times. The average value $\overline{x_i}$ of x_i is $s_0(x_i)$; therefor

$$\overline{\sum_i x_i} = \sum_i s_0(x_i) = k$$

and for the same reason

$$\overline{\sum_i w_i c_i} = \sum_i \frac{s_0(w_i)}{s_0(x_i)} \overline{x_i} c_i = \sum_i s_0(w_i) c_i \leq C.$$

Thus, on average, the solution satisfies the constraints. At the end, keep the best solution that satisfies all the constraints.
3) *Random:* In addition, randomly choose the rounding. If $\sum_i x_i = k$, which is the case on average, evaluate the weights by solving the SDP formulation. The steps are repeated several times, and keep the best value.

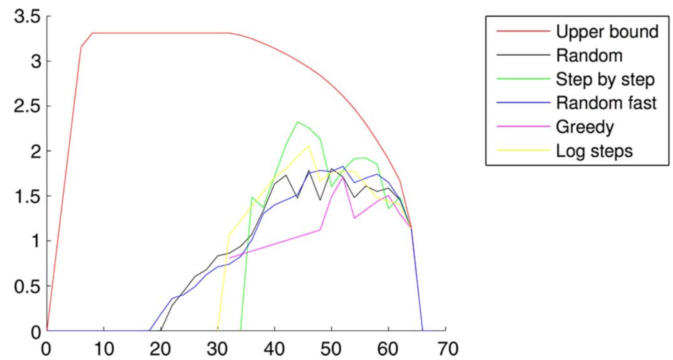


Fig. 8. λ_2 as a function of k. The results from several rounding methods are represented for a 20-node graph.

4) *Step by step:* Select the biggest element $s_0(x_i) < 1$ and affect its value to 1 in the SDP formulation. Then, solve the SDP again and repeat k times for these two steps.
5) *Log step by step:* This is the same idea as the "step by step" except that, at each step, choose the best half of the remaining elements. Thus, there are only $\log(k)$ SDPs that have to be solved.

2) Numerical Results: The simulation is set up with 20 nodes randomly generated in a square. The results are presented in Fig. 8 with λ_2 as a function of k. The upper bound obtained by the relaxation is plotted, as well as all the rounding techniques.

It turns out that some techniques may fail to find a solution. In this case, the corresponding values are removed from the figure.

It can be seen that the algorithms can achieve the upper bound at the maximum number of edges k_{lim}. This shows that any algorithm based on edge addition without considering the variable weights is not adapted.

Pros and Cons: Each of the methods presented has some advantages and drawbacks. The one that gives the best result is the *step-by-step* method. The fastest is *random fast*. In addition, the method that gives the best tradeoff between speed and value is *log step by step*.

D. Relaxed SDP With Rounding Algorithm

This algorithm is used to solve relatively small-scale problem when the exhaustive search described in Section III-A fails.

1) Golden Section Search: For a given k, a well-connected network with k edges can now be found. Instead of testing all the possible values of k, the search can be speeded up by considering that algebraic connectivity is a concave function of k.

This approximation leads to better results with rounding methods that have good regularity. For large networks, the rounding methods with lower regularity can be used. Instead of computing the value for a given k, a local average value is computed based on three values

$$\forall k \in \mathbb{N}, \quad \tilde{f}(k) = \frac{f(k-1) + f(k) + f(k+1)}{3}.$$

As only the value of the connectivity for integer values of k can be computed, it is not possible to use continuous optimization principles. Thus, the golden section search [35] is adopted. It consists in creating a decreasing set of intervals containing the optimal value

$$\forall i \in \mathbb{N}, \quad [a_{i+1}, b_{i+1}] \subset [a_i, b_i]$$

and $k_{\text{opt}} \in [a_i, b_i]$. Two test values $c_i < d_i$ in $[a_i, b_i]$ facilitate the search. The rules used to update the interval are: 1) $f(c_i) < f(d_i) \Rightarrow [a_{i+1}, b_{i+1}] = [c_i, b_i]$ and 2) $f(c_i) > f(d_i) \Rightarrow [a_{i+1}, b_{i+1}] = [a_i, d_i]$.

At each step, only one new value of f needs to be computed. This new value is usually chosen so that the test values are at the golden ratio $\phi = (1 + \sqrt{5})/2$. Here, the value is rounded to get an integer. This method allows, on average, to divide the length of the interval by ϕ at each step.

2) Relaxed SDP With Rounding Algorithm: All the steps of the algorithm are summed up. The SDP is solved using the SeDuMi solver [34] and the golden section search. This algorithm that leads to the approximation of the optimum is listed in Algorithm 1.

Algorithm 1 Relaxed SDP with step-by-step rounding

1: Initialize a, b and d
2: **while** $b - a > 2$ **do**
3: Choose c in $\{(a + d/2), (d + b/2)\}$
4: Solve the relaxed SDP with $k = c$
5: **for** $p = 1$ to k **do**
6: $j \leftarrow \arg\max_i \{x_i | x_i < 1\}$
7: Impose $x_j = 1$
8: Solve the SDP
9: **end for**
10: **if** $f(c) < f(d)$ **then**
11: $a \leftarrow c$
12: **else**
13: $b \leftarrow d, d \leftarrow c$
14: **end if**
15: **end while**
16: **return** λ_2

The complexity of this algorithm can be analyzed, which depends on several parameters of the problem. The algorithm uses the step-by-step rounding technique and requires to solve $k+1$ SDPs for each value of k selected. Each step has a different value for k, and most of them are close to k_{opt}.

In addition, there are U such steps. U is defined by $k_{\text{lim}} \phi^{-U} = 1$ since at each step the length of the interval is divided by ϕ. It is obtained that

$$U = \frac{\log(1/k_{\text{lim}})}{\log(1/\phi)}.$$

Complexity T also needs to be considered to solve the SDP. T is a polynomial in the size of the entry, which is equivalent to n^2. Therefor, T is a polynomial function of n.

Therefor, the complexity of the whole algorithm can be approximated by $O(k_{\text{opt}} UT)$.

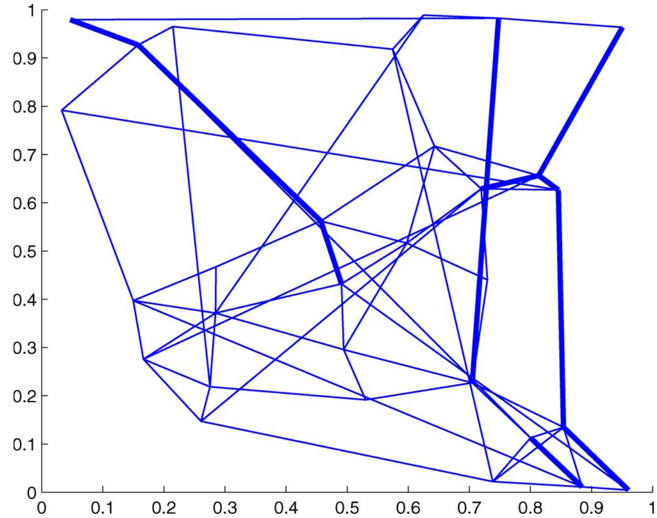


Fig. 9. Optimal result for a 30-node graph.

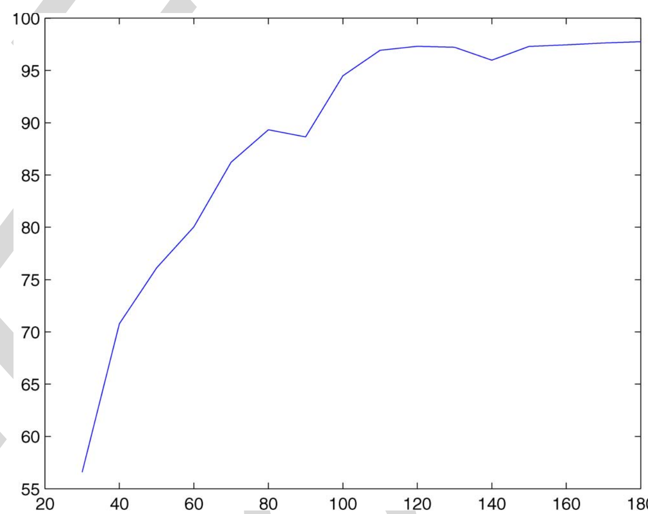


Fig. 10. Values of r for different values of C.

E. Numerical Results

1) Optimal Network: In order to test the full algorithm, a set of random nodes is generated in a square. An example of the optimal network for 30 nodes is shown in Fig. 9.

The edges that have a weight greater than the lower bound α are represented with a thicker line. In the example in Fig. 9, there are ten edges with a larger weight value than α.

2) Efficiency: Now, the efficiency of the result is going to be evaluated by comparing the optimal algebraic connectivity to the upper bound. For a given set of nodes, the problem is solved for different values of the total operating cost budget C and the percentage of the solution is computed compared to the bound

$$r = 100 \times \frac{val(P)}{val(R)} \%.$$

The result, as illustrated in Fig. 10, shows that for small values of C, the best result found is very far from the upper

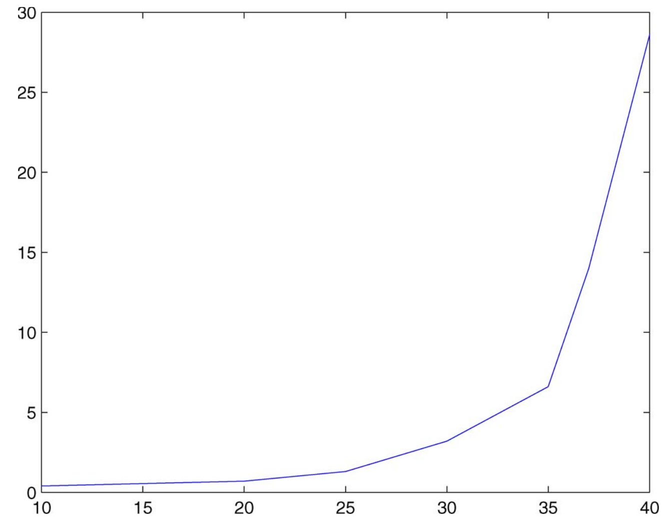


Fig. 11. Time (in seconds) to solve the SDP formulation of the problem for a given number of n nodes.

bound. However, when increasing C, the objective value of the problem P quickly increases to reach the value of its relaxed problem R.

IV. Large-Scale Air Transportation Networks

A. Necessity

The method in the previous section is going to be applied to large networks. The most time-consuming computation in the process is solving the SDP. Fig. 11 shows the computational time of solving one SDP for n nodes. It is observed that the running time increases very rapidly. In fact, for n nodes, there are $n(n-1)+2 \sim n^2$ variables in the SDP. As several SDPs need to be computed in order to solve the problem, it becomes impossible for $n \geq 35$ on the authors' workstation.

However, it is necessary to get some results for large values of n because real networks are usually large. For example, the air transportation network contains several hundred nodes when considering the entire USA.

B. Cluster Decomposition

Since the key factor for operating cost for each link is the route distance (a longer distance route consumes more fuel), the idea in this section is to divide the airports into $g \in \mathbb{N}$ clusters based on the distance between the nodes. These clusters can be solved independently with the relaxed SDP method (Algorithm 1) developed in the previous section and can be connected afterward.

To connect the cluster, choose k major nodes in each cluster that will be connected to each other. Then, the problem P has to be solved for these $g \times k$ nodes, except that links between two nodes from the same cluster are not allowed; hence, the graph is not complete.

At the end, $g+1$ problems of type (P) need to be solved to get the final result. Fig. 12 shows the idea of the decomposition into several clusters and the selection of major nodes.

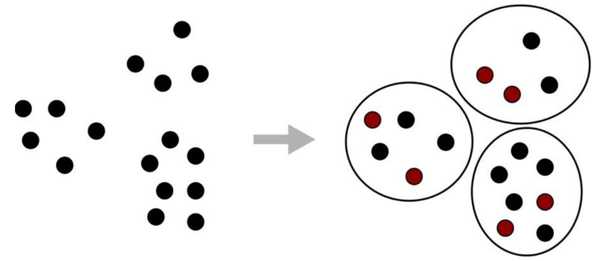


Fig. 12. Set of 16 nodes separated in three clusters with two major nodes in each cluster (in red).

There are several parameters whose values have to be chosen to apply this idea. First, choose the number of clusters and how many major nodes are used in each cluster to connect to other clusters. Second, choose which nodes are kept as major nodes among each cluster. Naturally, it is decided here to take the airports that have the largest traffic demand.

In addition, to solve the problem for each small problem $1 \leq i \leq g+1$, the value of the maximum operating cost budget C_i in each cluster has to be chosen. A natural option is to choose C_i proportional to the sum s_i of all the costs of the edges in the cluster i and such that $\sum_{i=1}^{g+1} C_i = C$

$$s_i = \sum_{(x,y)\in E} c_{xy},$$

$$C_i = \frac{s_i}{\sum_{j=1}^{g+1} s_j} C.$$

The separation of the nodes into several clusters is made by k-means algorithm [36]. This algorithm has the advantages of being fast, easy to implement, and generally giving good results.

To sum up the method described above, the full cluster decomposition algorithm is listed in Algorithm 2.

Algorithm 2 Large-scale cluster decomposition

1: Initialize g, C_i
2: k-means algorithm gives g clusters
3: **for** $p = 1$ to g **do**
4:     Solve the cluster problem (with Algorithm 1)
5: **end for**
6: Solve the major node problem (Algorithm 1)
7: Build the resulting network
8: **return** $\lambda_2(L)$

C. Evaluation of Efficiency

The goal here is to show that if all $g+1$ clusters are well connected, the resulting graph is well connected too. This depends on the values of some parameters that characterize how each cluster is linked to the others.

g clusters are considered. Each cluster has n nodes and k of its nodes are used to connect to other clusters. Let G be the matrix of the graph and F be the vector defined by the

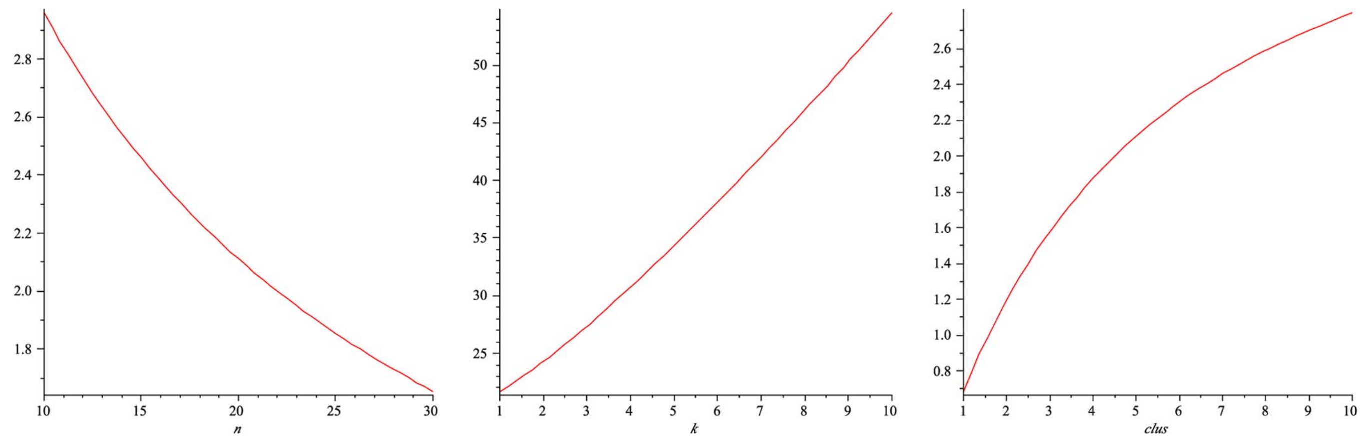


Fig. 13. (a) $\lambda_2 = f(n)$; (b) $\lambda_2 = f(k)$; (c) $\lambda_2 = f(g)$.

expression below. If, for instance, $g = 3$, matrix G can be put into the following form:

$$G = \left(\begin{array}{ccc|ccc|ccc} & & & E & 0 & 0 & E & 0 & 0 \\ & A_1 & & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline E & 0 & 0 & & & & E & 0 & 0 \\ 0 & 0 & 0 & & A_2 & & 0 & 0 & 0 \\ 0 & 0 & 0 & & & & 0 & 0 & 0 \\ \hline E & 0 & 0 & E & 0 & 0 & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & & A_3 & \\ 0 & 0 & 0 & 0 & 0 & 0 & & & \end{array}\right)$$

$$F = \left(\begin{array}{c} \alpha e \\ \beta e \\ \beta e \\ \hline 0 \\ 0 \\ 0 \\ \hline -\alpha e \\ -\beta e \\ -\beta e \end{array}\right)$$

with the following notation. e is the all-one vector. α and β are constants that will be computed in the next paragraph. A_1, A_2, and A_3 represent the adjacency matrices of the three clusters. E is a $k \times k$ matrix with all elements equal to 1.

1) Fiedler Vector: The Fiedler vector is the vector solution of the minimization problem

$$\min_{x \in \mathbb{R}^n} \left\{ x^T L x \| \|x\| = 1, \quad xe = 0 \right\}.$$

It is known to be an indicator on how to split a graph into two smaller graphs. In fact, the nodes that have the same sign in this vector form a cut of the graph (see [37]).

Here, the optimal cut will naturally be found between two clusters. Since some of the nodes play the same role, the Fiedler vector has a shape close to F where α and β are constants that need to be determined.

This assumption has been verified by numerical experiments and it seems to be a very good approximation of the real Fiedler vector.

2) Computing the Connectivity: Consider that the Fiedler vector has the form of F and matrices are full, which means all nondiagonal elements are equal to 1. The matrix products give

$$\begin{aligned} \lambda_2 &= F^T L F, \\ \lambda_2 &= 2k\alpha X + 2(n-k)\beta Y \end{aligned}$$

with

$$\begin{aligned} X &= \alpha\,(n - 1 + k(g-1)) - (k-1)\alpha - (n-k)\beta + k\alpha, \\ Y &= \beta(n-1) - k\alpha - (n-k-1)\beta. \end{aligned}$$

It is also known that $\|F\| = 1$; thus

$$2k\alpha^2 + (2n - 2k)\beta^2 = 1,$$

$$\alpha = \sqrt{\frac{1 - (2n-2k)\beta}{2k}}.$$

Substitute α with this expression and β is given by the

$$\frac{d\,\lambda_2}{d\,\beta} = 0. \tag{7}$$

With a computation software package like Maple, this gives us the expression of function f such that

$$\lambda_2 = f(n, k, g).$$

3) Resulted Curves: By solving (7), the values of α, β, and λ_2 are obtained. The following figures have been obtained with Maple. Among the three parameters k, g, and n, fix two of them and let the third one vary to see its influence on connectivity.

Fig. 13 provides a clearer idea on how to choose the value of each parameter. For example, connectivity is almost linear regarding k but has a concave shape when represented as a function of the number of clusters g.

There exists a tradeoff: If g is too large, kg will be too large to be solved. On the contrary, if g is too small, each of the cluster will have too many nodes to be solved.

4) Numerical Results: The data used are the 100 largest cities in the United States. Fig. 14 shows the 100 biggest cities without any link. The cluster decomposition is used to divide these 100 cities into $g = 5$ groups. In each group, we selected $k = 5$. The lower bound $\alpha = 2$ and the upper bound

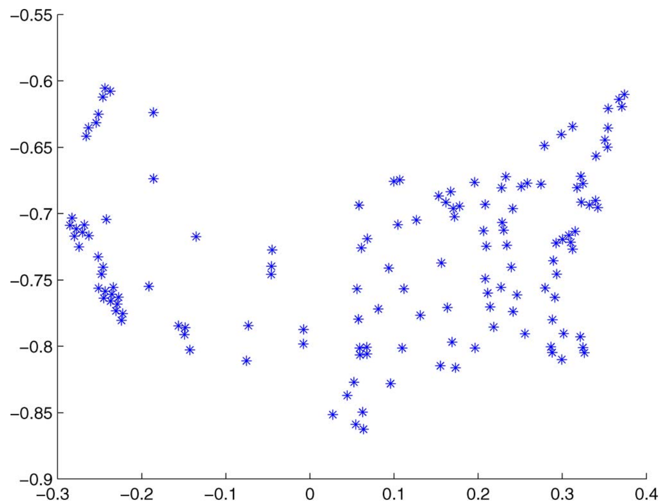


Fig. 14. Shown are the 100 largest cities in the USA.

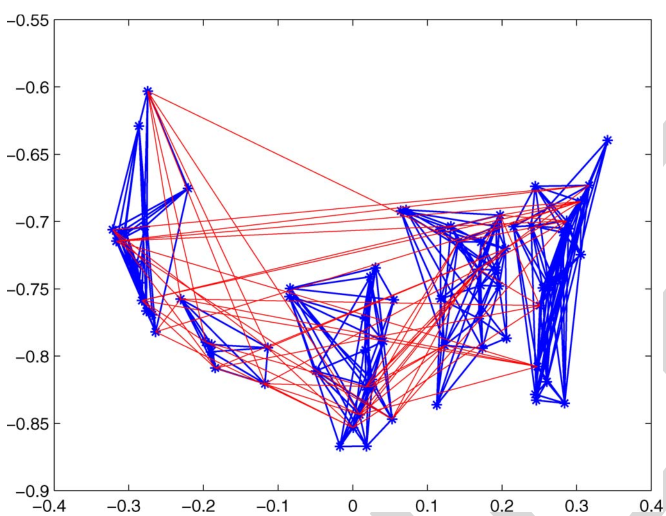


Fig. 15. Result for the 100 largest cities in the USA.

$\beta = 10$. The total running time is 317 s with MATLAB on our workstation. The algebraic connectivity that we have achieved is $\lambda_2 = 2.6$.

The optimal network found is illustrated in Fig. 15. The blue lines represent the edges inside each cluster and the red lines represent the edges that connect nodes from different clusters.

In a real network, most airlines use spoke-hub planning, in which the regional airports are clustered and connected to their regional hub airport. Fig. 15 shows the same behavior that regional airports are clustered together. In a real network in the United States, the number of the major nodes k for each cluster is usually 1, which means that the hub airport is the only major node for its cluster. In a real network in Europe, the number of the major nodes k is usually bigger than 1. In fact, the number k in a European network is those hub airports in each country. For example, Air France has hubs at Paris and Lyon $(k = 2)$ and Air Berlin has its hubs at Berlin, Düsseldorf, Hamburg, and Munich $(k = 4)$. In summary, the result in Fig. 15 is a practical design, and more importantly, the computation is very efficient for large-scale network planning.

V. Directed Air Transportation Network

A. From Undirected Graph to Directed Graph

In this section, the methods are extended in directed graphs. In order to be consistent with the undirected case, the graphs need to be balanced, which means that the number of aircraft that comes in is the same as the number of aircraft that leaves the airport

$$\forall i \in \{1, \ldots, n\}, \quad \sum_{j=1}^{n} w_{ij} = \sum_{k=1}^{n} w_{ki}.$$

It is clear that the set of undirected graphs is included in the set of directed balanced graphs. Therefor, the results should be at least as good as in the previous section.

Definition: According to [38], if $\Omega = \{x \in \mathbb{R}^n, xe = 0, \|x\| = 1\}$, the definition of the algebraic connectivity can be extended for directed balanced graphs with

$$\min_{x \in \Omega} x^T L x = \lambda_2 \left(\frac{1}{2}(L + L^T) \right).$$

Property 4: In the directed case and with this definition of the algebraic connectivity, the upper bound given by the continuous relaxation is the same as in the undirected case.

Proof: Given the optimum directed balanced graph in the relaxed problem and its incidence matrix G, H can be created

$$H = \frac{G + G^T}{2}$$

where H is symmetric and satisfies all the constraints of the problem since they are linear. In addition with the definition of the connectivity for directed graphs, the connectivity of H is clearly the same as the connectivity of G.

Since it is also known that undirected graphs are a subset of directed balanced graphs, the bounds are equal in both cases. This will allow us to easily evaluate the improvement brought by directed graphs. ■

B. Results

The same method is used as in Section III. The results are impressively better with directed graphs, as shown in Fig. 16. The best value for the directed balanced case almost reached the upper bound. It is also noticed that the optimal result has less edges than the optimal network in the undirected case (an edge in the undirected case is counted in both ways).

It is shown in Fig. 17 that most of the edges in the optimal solution are oriented in only one way, which shows that the solution is very different from the undirected case.

However, there is an important drawback. Indeed, two times as many variables are needed for directed networks. Thus, the problem takes a much longer time to be solved and is only applicable on smaller networks.

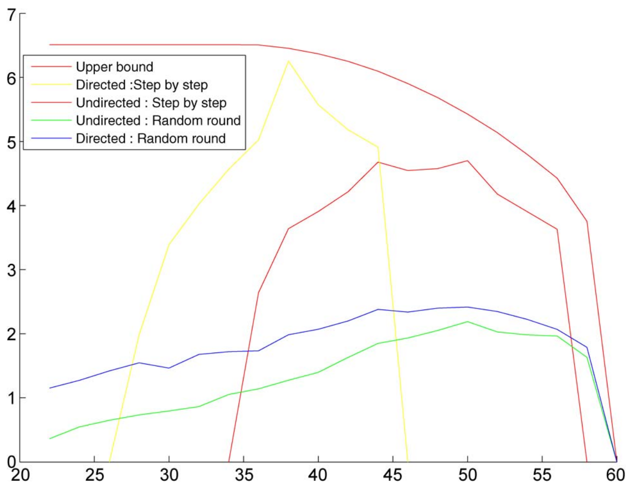


Fig. 16. $\lambda = f(k)$ for the same graph with different approaches.

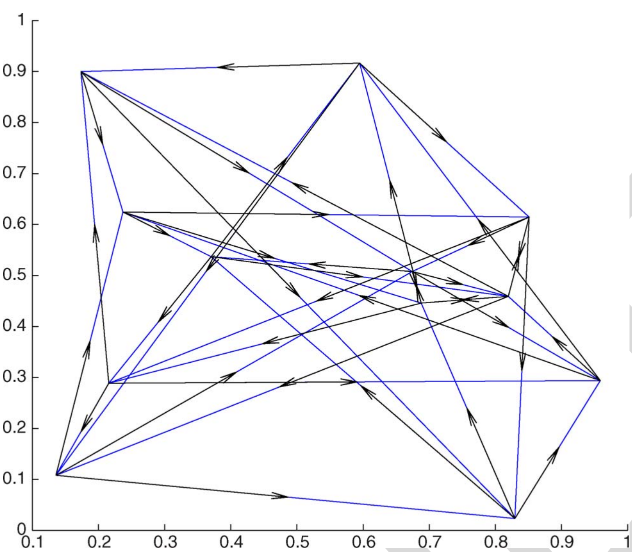


Fig. 17. Optimal directed graph.

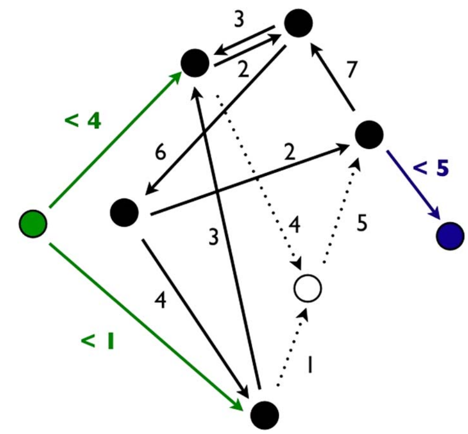


Fig. 18. Example of a graph with a node failure.

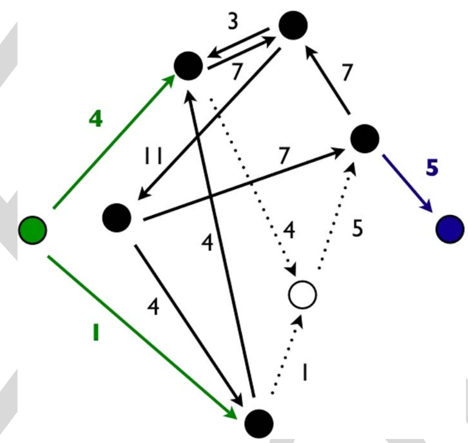


Fig. 19. Example of a graph after maximization of the flow.

C. Failure Case

If an edge or a node is removed, the graph is not balanced anymore. This can cause important problems in practice; hence, the remaining weights in the graph need to be changed to handle this problem.

This operation can be done by using a flow algorithm. The first step is to link the nodes with positive aircraft balance to a virtual source and those with negative balance to a sink. The capacities of these links are equal to the absolute value of the difference in the balance flow for the node. The capacity of the other links of the graph is β.

Then, consider the problem of the maximization of the flow from the source to the sink. This problem can be solved by using Edmonds–Karp algorithm [39], which has efficient complexity, i.e., $O(nm^2)$. This algorithm maintains the balance of the flow at each node.

At the end, the graph is the graph solution of the flow problem when the source and the sink are removed. Figs. 18 and 19 show an example of a graph in which a node is removed, before and after maximization of the flow. The graph in Fig. 19 is now balanced by Edmonds–Karp algorithm after removing the source (green node) and the sink (blue node).

VI. Conclusion

In this paper, a new problem concerning the maximization of the algebraic connectivity of a network has been presented and studied. This problem consists of finding both the edges of the graph and their weight assignment under several constraints. First, the problem in small networks is exactly solved, and it is shown that the problem cannot be separated into two already studied independent problems. Then, a relaxed SDP method with step-by-step rounding is presented to solve the problem approximately for better computational efficiency. With the method developed for small-scale network, the cluster decomposition method is proposed to solve the large-scale network problem and successfully find the near-optimal solution for a network of the 100 largest cities in the United States. Finally, the study is extended to directed graphs. Numerical experiments and analysis are performed for all the proposed algorithms. The developed methods are able to model and maximize robustness in air transportation networks for airlines and for the NAS. They also provide an option to improve current ways of the generic network design.

References

[1] R. Guimera and L. Amaral, "Modeling the world-wide airport network," *Eur. Phys. J. B, Condens. Matter Complex Syst.*, vol. 38, no. 2, pp. 381–385, Mar. 2004.

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

[3] E. Vargo, R. Kincaid, and N. Alexandrov, "Towards optimal transport networks," *J. Syst., Cybern. Inf.*, vol. 8, no. 4, pp. 59–64, Jul. 2010.

[4] International Civil Aviation Organization (ICAO), "Procedures for air navigation services—Rules of the air and air traffic services," Montréal, QC, Canada, doc 4444-RAC/501, 2010.

[5] P. Kostiuk, D. Lee, and D. Long, "Closed loop forecasting of air traffic demand and delay," presented at the 3rd USA/Europe Air Traffic Management R&D Seminar, Napoli, Italy, Jun. 2010, Paper 80.

[6] K. Nam and T. Schaefer, "Forecasting international airline passenger traffic using neural networks," *Logist. Transp. Rev.*, vol. 31, no. 3, pp. 239–251, Sep. 1995.

[7] F. Alamdari and S. Fagan, "Impact of the adherence to the original low cost model on the profitability of low cost airlines," *Transp. Rev.*, vol. 25, no. 3, pp. 377–392, May 2005.

[8] M. Dresner, J.-S. C. Lin, and R. Windle, "The impact of low-cost carriers on airport and route competition," *J. Transp. Econ. Policy*, vol. 30, no. 3, pp. 309–328, Sep. 1996.

[9] K. Button and S. Lall, "The economics of being an airport hub city," *Res. Transp. Econ.*, vol. 5, pp. 75–105, 1999.

[10] T. Oum, A. Zhang, and Y. Zhang, "A note on optimal airport pricing in a hub-and-spoke system," *Transp. Res. B, Methodol.*, vol. 30, no. 1, pp. 11–18, Feb. 1996.

[11] X.-B. Hu, W.-H. Chen, and E. Di Paolo, "Multiairport capacity management: Genetic algorithm with receding horizon," *IEEE Trans. Intell. Transp. Syst.*, vol. 8, no. 2, pp. 254–263, Jun. 2007.

[12] X.-B. Hu and E. Di Paolo, "Binary-representation-based genetic algorithm for aircraft arrival sequencing and scheduling," *IEEE Trans. Intell. Transp. Syst.*, vol. 9, no. 2, pp. 301–310, Jun. 2008.

[13] Y. Eun, I. Hwang, and H. Bang, "Optimal arrival flight sequencing and scheduling using discrete airborne delays," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 2, pp. 359–373, Jun. 2010.

[14] Z.-H. Zhan, J. Zhang, Y. Li, O. Liu, S. Kwok, W. Ip, and O. Kaynak, "An efficient ant colony system based on receding horizon control for the aircraft arrival sequencing and scheduling problem," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 2, pp. 399–412, Jun. 2010.

[15] *FlightStats-Global Flight Tracker, Status Tracking and Airport Information*, FlightStats, Inc., Portland, OR, USA, 2013, retrieved on March 16, 2013. [Online]. Available: http://www.flightstats.com

[16] Federal Aviation Administration (FAA), "FAA Aerospace Forecasts FY 2008-2025," Washington, DC, USA, Dec. 2010.

[17] S. Conway, "Scale-free networks and commercial air carrier transportation in the United States," in *Proc. 24th Int. Congr. Aeron. Sci.*, Yokohama, Japan, 2004, pp. 1–12.

[18] P. A. Bonnefoy, "Scalability of the air transportation system and development of multi-airport systems: A worldwide perspective," Ph.D. dissertation, MIT, Cambridge, MA, USA, Jun. 2008.

[19] N. Alexandrov, "Transportation network topologies," NASA Langley Res. Center, Hampton, VA, USA, Tech. Rep., 2004.

[20] T. Kotegawa, D. DeLaurentis, K. Noonan, and J. Post, "Impact of commercial airline network evolution on the US air transportation system," in *Proc. 9th USA/Europe ATM Res. Develop. Semin.*, Berlin, Germany, 2011.

[21] A. Bigdeli, A. Tizghadam, and A. Leon-Garcia, "Comparison of network criticality, algebraic connectivity, and other graph metrics," presented at the 1st Annu. Workshop Simplifying Complex Network Practitioners, Venice, Italy, 2009, Paper 4.

[22] A. Jamakovic and S. Uhlig, "On the relationship between the algebraic connectivity and graph's robustness to node and link failures," in *Proc. 3rd EuroNGI Netw. Conf.*, May 2007, pp. 96–102.

[23] A. Jamakovic and P. V. Mieghem, "On the robustness of complex networks by using the algebraic connectivity," in *Proc. Netw. Ad Hoc Sensor Netw., Wireless Netw., Next Gen. Internet*, A. Das, H. K. Pung, F. Bu Sung Lee, and L. Wong Wai Choong, Eds., 2008, pp. 183–194.

[24] R. H. Byrne, J. T. Feddema, and C. T. Abdallah, "Algebraic connectivity and graph robustness," Sandia Nat. Lab., Albuquerque, NM, USA, Tech. Rep., Jul. 2009.

[25] P. Wei and D. Sun, "Weighted algebraic connectivity: An application to airport transportation network," in *Proc. 18th IFAC World Congress*, Milan, Italy, Aug. 2011, pp. 13 864–13 869.

[26] A. Ghosh and S. Boyd, "Growing well-connected graphs," in *Proc. 45th IEEE Conf. Decis. Control*, Dec. 2006, pp. 6605–6611.

[27] J. Sun, S. Boyd, L. Xiao, and P. Diaconis, "The fastest mixing Markov process on a graph and a connection to a maximum variance unfolding problem," *SIAM Rev.*, vol. 48, no. 4, pp. 681–699, Nov. 2006.

[28] F. Goring, C. Helmberg, and M. Wappler, "Embedded in the shadow of the separator," *SIAM J. Optim.*, vol. 19, no. 1, pp. 472–501, Feb. 2008.

[29] S. Boyd, "Convex optimization of graph Laplacian eigenvalues," in *Proc. Int. Congr. Math.*, 2006, vol. 3, pp. 1311–1319.

[30] S. Boyd, P. Diaconis, and L. Xiao, "Fastest mixing Markov chain on a graph," *SIAM Rev.*, vol. 46, no. 4, pp. 667–689, Dec. 2004.

[31] M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak Math. J.*, vol. 23, no. 98, pp. 298–305, 1973.

[32] B. Mohar, "The Laplacian spectrum of graphs," in *Graph Theory, Combinatorics, and Applications*. New York, NY, USA: Wiley, 1991, pp. 871–898.

[33] M. Siotani, "Some applications of Loewner's ordering on symmetric matrices," *Ann. Inst. Stat. Math.*, vol. 19, no. 1, pp. 245–259, Dec. 1967.

[34] J. F. Sturm, "Using SeDuMi 1.02, A Matlab toolbox for optimization over symmetric cones," *Optim. Methods Softw.*, vol. 11, no. 1–4, pp. 625–653, Jan. 1999.

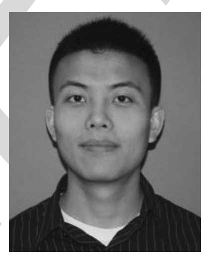
[35] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *The Art of Scientific Computing,* 3rd ed. Cambridge, U.K.: Cambridge Univ. Press, 2007. ch. Section 10.2. Golden Section Search in One Dimension, pp. 492–496.

[36] J. Kleinberg and E. Tardos, *Algorithm Design*. Reading, MA, USA: Addison-Wesley, 2005, ch. 4.7 Clustering, pp. 157–161.

[37] S. Martinez, G. Chatterji, D. Sun, and A. Bayen, "A weighted graph approach for dynamic airspace configuration," presented at the AIAA Conf. Guidance, Control Dynamics, Hilton Head, SC, USA, Aug. 2007, Paper AIAA 2007-6448.

[38] C. Wu, "Algebraic connectivity of directed graphs," *Linear Multilinear Algebra*, vol. 53, no. 3, pp. 203–223, Jun. 2005.

[39] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, MA, USA: MIT Press, 2009, ch. 26.2 The Ford-Fulkerson method, pp. 727–730.



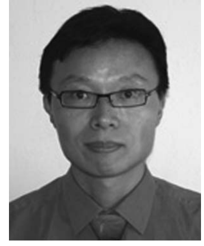
Peng Wei (M'12) received the Bachelor's degree in automation from Tsinghua University, Beijing, China; the Master's degree in electrical engineering from Stony Brook University, Stony Brook, NY, USA; and the Ph.D. degree in aerospace engineering from Purdue University, West Lafayette, IN, USA.

He is an Operations Research Consultant with American Airlines, Fort Worth, TX, USA. His research interests include next-generation air transportation system, environmental impact of aviation, and unmanned aerial vehicle integration in the National Airspace System.

Dr. Wei received the Purdue College of Engineering Outstanding Research Award in 2013.

Gregoire Spiers received the Bachelor's degree in applied mathematics from École Polytechnique, Palaiseau, France, in 2012.

He was a Visiting Student with Purdue University, West Lafayette, IN, USA, in 2011. He is currently a Research Engineer with Amadeus S.A.S., Sophia Antipolis, France.



Dengfeng Sun (M'08) received the Bachelor's degree in precision instruments and mechanology from Tsinghua University, Beijing, China; the Master's degree in industrial and systems engineering from The Ohio State University, Columbus, OH, USA; and the Ph.D. degree in civil engineering from University of California, Berkeley, CA, USA.

He is an Assistant Professor with the School of Aeronautics and Astronautics, Purdue University, West Lafayette, IN, USA. His research interests include control and optimization, with an emphasis on applications in air traffic flow management, dynamic airspace configuration, and studies for the next-generation air transportation system.

AUTHOR QUERIES

AUTHOR PLEASE ANSWER ALL QUERIES

AQ1 = Please provide keywords.
AQ2 = Please check if changes made in the affiliation note of author Gregoire Spiers are appropriate, with reference to the given biography. Please also check accuracy of postal code and location.
AQ3 = Please provide page range in Ref. [20].
AQ4 = Please check accuracy of provided location.

END OF ALL QUERIES

Algebraic Connectivity Maximization for Air Transportation Networks

Peng Wei, *Member, IEEE*, Gregoire Spiers, and Dengfeng Sun, *Member, IEEE*

***Abstract*—It is necessary to design a robust air transportation network. An experiment based on the real air transportation network is performed to show that algebraic connectivity is a fair measure for network robustness under random failures. Therefor, the goal of this paper is to maximize algebraic connectivity. Some researchers solve the maximization of the algebraic connectivity by choosing the weights for the edges in the graph. Others focus on the best way to add edges in a network in order to optimize the connectivity. In this paper, the authors formulate a new air transportation network model and show that the corresponding algebraic connectivity optimization problem is interesting because the two subproblems of adding edges and choosing edge weights cannot be treated separately. The new problem is formulated and exactly solved in a small air transportation network case. The authors also propose the approximation algorithm in order to achieve better efficiency. For large networks, the semidefinite programming with cluster decomposition is first presented. Moreover, the algebraic connectivity maximization for directed networks is discussed. Simulations are performed for a small-scale case, large-scale problem, and directed network problem.**

***Index Terms*—Author, please supply index terms/keywords for your paper. To download the IEEE Taxonomy go to http: //www.ieee.org/documents/Taxonomy_v101.pdf.** AQ1

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 (link) set E represents all the m direct flight routes between airports. 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} if there is a direct route between airports v_i and v_j. There are many factors to be considered when designing an air transportation network, such as traffic demand, operating cost, airport hubs, market competition, multiairport systems, and scheduling [5]–[14]. In this paper, we focus on investigating the network robustness maximization, particularly the algebraic connectivity maximization.

Manuscript received November 6, 2012; revised April 5, 2013 and August 20, 2013; accepted October 1, 2013. The Associate Editor for this paper was J.-P. B. Clarke.

P. Wei is with the Operations Research and Advanced Analytics, American Airlines, Fort Worth, TX 76155 USA (e-mail: peng.wei@aa.com).

G. Spiers was with the Department of Applied Mathematics, École Polytechnique, 91120 Palaiseau, France. He is now with Amadeus S.A.S., 06902 Sophia Antipolis, France (e-mail: gregoire.spiers@polytechnique.org). AQ2

D. Sun is with the School of Aeronautics and Astronautics, Purdue University, West Lafayette, IN 47907 USA (e-mail: dsun@purdue.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITS.2013.2284913

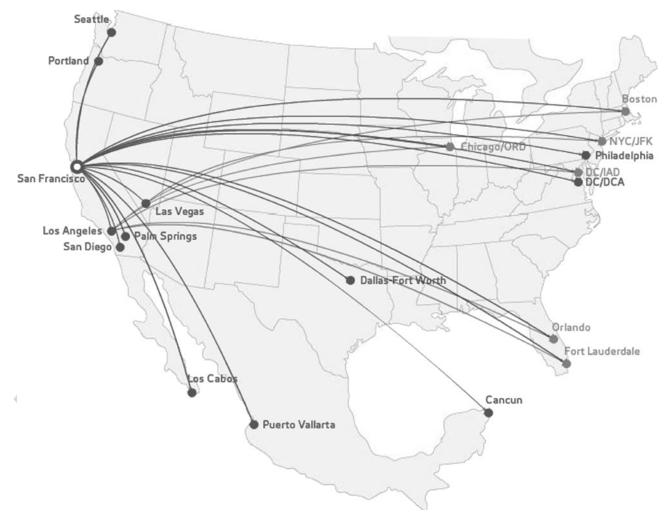


Fig. 1. Air transportation network route map for Virgin America Airlines.

A. Algebraic Connectivity and Air Transportation Network Robustness

In order to illustrate the relationship between the algebraic connectivity and air transportation network robustness, a real air transportation network of Virgin America is studied. The following experiment shows that algebraic connectivity is a fair measurement for the network robustness with regard to random link failures under the current Virgin America network topology.

According to the current route map of Virgin America in Fig. 1, we consider the 16 airports in the United States and obtain the adjacency matrix as Table I. The 16 United States airports include Boston (BOS), New York City/John F. Kennedy (JFK), Philadelphia, Washington Dulles International Airport (DC/IAD), Ronald Reagan Washington National Airport (DC/DCA), Chicago O'Hare International Airport (ORD), Orlando, Fort Lauderdale, Dallas/Fort Worth, Seattle, Portland, San Francisco, Los Angeles, Las Vegas, San Diego, and Palm Springs and are indexed as numbers 1–16. San Francisco International Airport (SFO) and Los Angeles International Airport (LAX) are two major hubs of the entire network. Both have at least one direct flight to almost all the other airports.

In order to show how well the algebraic connectivity can measure the robustness of an air transportation network, we

TABLE I
ADJACENCY MATRIX CONSISTS OF 16 VIRGIN AMERICA AIRLINES AIRPORTS IN THE USA

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0
3	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
12	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
13	1	1	1	1	0	1	1	1	1	1	1	1	0	0	0	0
14	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

created six different *weighted air transportation networks* with the same topology in Table I by randomly assigning one of the three types of *link weights* to each route. Each link weight is an indication of link strength. A larger weight represents a stronger link and a smaller weight shows that the corresponding link is easier to fail. There are many reasons for route failure, such as weather disturbance, long ground delay program, long airspace flow program (AFP), aircraft mechanical problem, and upline flight delay/cancelation. The route failure rate statistics are published by each origin–destination pair (route) and different routes have different features [15]. For example, the route failure rate between JFK and BOS during summer is higher than that between SFO and LAX because of the crowded northeastern airspace (AFP is more frequent) and more summer thunderstorms. Another example is that a shorter route is easier to fail than a longer transcontinental route because: 1) airlines usually put larger aircraft on transcontinental route, and these aircraft are more robust to weather disturbance and 2) airlines are more likely to cancel shorter route flights because the flight frequency on a shorter route is higher; therefor, the passengers on the canceled flight are easier to protect (be reaccommodated to later flights). In summary, each route has its own features and thus in our model we consider that they have different possibilities for failure.

The three types of link weights are mapped to different link failure probabilities (see Table II). The link failure probability range [%0, %5] is obtained from the historical flight cancelation rate between September 15, 2012, and November 15, 2012 [15].

A network failure is defined as the existence of at least one pair of nodes that cannot access each other through any one or multiple links. For each one of the six weighted networks, 1000 trials are performed. In each trial, every link fails randomly according to the failure probabilities listed in Table II. The total number of network failures is counted in 1000 random trials. The results are shown in Table III with algebraic connectivity sorted in ascending order.

TABLE II
MAPPING BETWEEN LINK WEIGHTS AND LINK FAILURE PROBABILITIES

link weight w_{ij}	1	2	3
link failure probability	5%	3%	1%

TABLE III
NETWORK FAILURE NUMBERS WITH DIFFERENT ALGEBRAIC CONNECTIVITY VALUES

algebraic connectivity	total failures in 10000 trials
1.0306	1113
1.7586	991
1.8661	763
1.9711	571
2.3128	423
2.7393	355

We can see that with higher algebraic connectivity, the network is more robust and has fewer network failures. With lower weighted algebraic connectivity, the network is easier to break down. Therefor, algebraic connectivity is a fair robustness measure for the air transportation network, and we need to find the maximized algebraic connectivity.

The air traffic demand is expected to continue its rapid growth in the future. The Federal Aviation Administration estimated that the number of passengers is projected to increase by an average of 3% every year until 2025 [16]. The expanding traffic demand on the current air transportation networks of different airlines will cause more and more flight cancelations with the limited airport and airspace capacities. As a result, more robust air transportation networks are desired to sustain the increasing traffic demand for each airline and for the entire National Airspace System (NAS). This is the major motivation of this paper.

B. Related Work

An air transportation network and its robustness have been studied over the last several years. Guimera and Amaral first studied the scale-free graphical model of the air transportation network [1]. Conway showed that it was better to describe the national air transportation system or the commercial air carrier transportation network as a system of systems [17]. Bonnefoy showed that the air transportation network was scale free with aggregating multiple airport nodes into meganodes [18]. Alexandrov defined that on-demand transportation networks would require robustness in system performance [19]. The robustness of an on-demand network would depend on the tolerance of the network to variability in temporal and spatial dynamics of weather, equipment, facility, crew positioning, etc. Kotegawa *et al.* surveyed different metrics for air transportation network robustness, including betweenness, degree, centrality, connectivity, etc. [20]. They selected clustering coefficient and eigenvector centrality as the network robustness metrics in their machine learning approach. Bigdeli *et al.* compared algebraic connectivity, network criticality, average degree, average node betweenness, and other metrics [21]. Jamakovic *et al.* found that algebraic connectivity was an important metric in the analysis of various robustness problems in several typical network models [22], [23]. Byrne *et al.* showed that algebraic connectivity was the efficient measure for the robustness of both small and large networks [24]. Vargo *et al.* in [3] chose algebraic connectivity as the robustness metric and built the optimization problem solved by the edge swapping-based tabu search algorithm.

In this paper, we measure the robustness of air transportation network by computing the algebraic connectivity, which is usually considered as one of the most reasonable and efficient evaluation methods [24], [25]. Although the maximized value of algebraic connectivity is abstract, the optimized air transportation network structure and weighting assignment provide us the applicable design.

There are some literature on algebraic connectivity maximization. The problems studied can be divided into two categories, namely, the edge addition problem and the variable weights problem.

1) **Edge addition problem:** The goal is to add or remove a given number of k edges on a graph in order to get the best algebraic connectivity. The edges to be added or removed are selected from a candidate set. The algorithms that have been developed to solve the problem include tabu search [25], greedy algorithms [25], [26], and rounded semidefinite programming (SDP) [26].
2) **Variable weights problem:** The edges of the graph are fixed and the goal is to determine the edge weights in order to maximize the algebraic connectivity. This is a convex optimization problem that is often solved by using an SDP formulation [27]–[29] or a subgradient algorithm [30].

C. Contribution

The major contribution of this paper compared with what has been studied is that we find that, in order to maximize the algebraic connectivity, the edge addition problem and the variable weights problem cannot be studied separately. Solving one of them independently will only result in a suboptimal solution. Therefor, we propose a new algorithm to solve both problems at the same time. How to choose the edges of the graph is demonstrated, as well as how to assign their weights. In addition, we are the first to present the cluster decomposition method to achieve better computation efficiency for large-scale networks. We are also the first to discuss the algebraic connectivity maximization for directed air transportation networks.

The rest of this paper is structured as follows. Section II shows why this problem naturally arises in air transportation networks and how it can be formulated. In Section III, the problem is exactly solved for small networks, and the fact that the two problems are not independent is highlighted. Then, the authors present the SDP formulation and the more efficient full algorithm, which includes relaxed SDP, and solution rounding is proposed. In Section IV, the problem for large networks is solved, the computational efficiency is analyzed, and the numerical results are provided. The algebraic connectivity optimization for directed air transportation networks is presented in Section V. Section VI concludes this paper.

II. Problem Formulation

A graph G with n nodes and m edges is used to define an air transportation network. Let $A = (a_{ij})$ be the adjacency matrix of G. The Laplacian matrix $L = (l_{ij})$ of G is defined by

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

The eigenvalues of L are sorted $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$. L is a semidefinite positive matrix; thus, for all i, $\lambda_i \geq 0$. It is also known that $\lambda_1 = 0$ since $Le = 0$ with $e = (1, \ldots, 1)$ [31].

Definition: The second smallest eigenvalue $\lambda_2(L)$ is the algebraic connectivity of G.

Now, recall the two key properties of the algebraic connectivity that will be used in this paper.

Property 1: Let $e = (1, \ldots, 1) \in \mathbb{R}^n$ and

$$\Omega = \left\{ x \in \mathbb{R}^n | \|x\| = 1, \quad e^T x = 0 \right\}.$$

The Courant–Fischer principle [32] states that

$$\lambda_2 = \min_{x \in \Omega} x^T L x. \tag{1}$$

Property 2: Function $w \to \lambda_2(w)$ is concave with w denoting the edge weight vector. This can be proven by seeing that $\lambda_2(w)$ is the pointwise infimum of a family of linear functions of w (see [27])

$$\lambda_2(w) = \inf_{\|v\|=1, e^T v = 0} v^T L v,$$

$$\lambda_2(w) = \inf_{\|v\|=1, e^T v = 0} \sum_{(i,j) \in E} w_{ij} (v_i - v_j)^2.$$

The goal of this paper is to maximize the algebraic connectivity of the network under several constraints.

There are $m = (n(n-1))/2$ edges in the complete symmetric graph. Each has a weight w_{ij} representing the link strength, as described in Section I. The following constraints are considered.

The edge weight representing link strength must be within the range between the lower bound α and the upper bound β

$$\forall (i,j) \in E, \quad \alpha \le w_{ij} \le \beta.$$

When there is no edge connecting v_i and v_j, the corresponding $w_{ij} = 0$.

There exists an operating cost c_{ij} for each link. In a real air transportation network, the cost for a route contains the fuel cost, aircraft maintenance cost, crew/flight attendant labor cost, cost for arrival/departure slots at runways, cost for gates at origin/destination airports, and cost for flying through airspace (international flights). In this paper, we use one link cost to represent the integrated operating cost. The operating cost is higher for a stronger link for several practical reasons. For example, we know that the most effective way to avoid a mechanical problem cancelation is to have spare parts or even a spare aircraft. Similarly, the most effective way to avoid a cancelation caused by crew legality or crew scheduling is to have enough standby crew. Both approaches can increase link strength; at the same time, they introduce higher costs. As for weather disturbances, to load extra fuel will give an aircraft more flight plan options with which it can be rerouted to avoid weather problems and prevent the cancelation. However, extra fuel also introduces higher cost. In addition, larger aircraft are more robust to weather disturbances. Nevertheless, to operate a larger aircraft costs more because of more fuel needed, more flight attendants, and even more crew (for international flights). Therefor, in this paper, we consider the linear cost for link strength. The total operating cost budget for all the links is limited by

$$\sum_{ij} w_{ij} c_{ij} \le C.$$

In summary, the complete problem that the authors aim at solving is

$$\max_{w} \lambda_2\left(L(w)\right) \quad \text{s.t.} \begin{cases} \sum_{ij} w_{ij} c_{ij} \le C \\ w_{ij} \in \{0, [\alpha, \beta]\} \end{cases}. \tag{P}$$

A. Alternative Formulation

In order to be able to solve the problem, the authors need to reformulate it by adding decision variables. The idea is to add, for each edge (i,j), a binary variable x_{ij} stating if there exists an edge between v_i and v_j

$$x_{ij} = 1 \Leftrightarrow w_{ij} \neq 0.$$

This is useful since now the domain of w can be expressed as

$$\forall (i,j), \quad \alpha x_{ij} \le w_{ij} \le \beta x_{ij}.$$

The problem now becomes

$$\max_{x,w} \lambda_2\left(L(w)\right) \quad \text{s.t. :} \begin{cases} x_{ij} \in \{0,1\} \\ \sum_{ij} w_{ij} c_{ij} \le C \\ \alpha x_{ij} \le w_{ij} \le \beta x_{ij}. \end{cases}$$

Then, variable k is added, which determines the number of edges in the graph. The final formulation of the problem is

$$\max_{x,w,k} \lambda_2\left(L(w)\right) \quad \text{s.t. :} \begin{cases} \sum_{i,j} x_{ij} = k \\ x_{ij} \in \{0,1\} \\ \sum_{i,j} w_{ij} c_{ij} \le C \\ \alpha x_{ij} \le w_{ij} \le \beta x_{ij}. \end{cases} \tag{2}$$

B. Difficulty

An important remark is that the problem cannot really be split into two steps of first deciding whether $w = 0$ or not and followed by choosing the appropriate weights. This is due to the fact that there are lower bound and upper bound constraints on w. However, assuming that the two steps are independent, a decoupled approach can be tried first. The first step is to choose edges for the empty graph that corresponds to the edge addition problem introduced in [25]

$$\max_{x} \lambda_2(L(x))$$
$$\text{s.t. :} \sum_i x_i = k, \quad x_i \in \{0,1\}, \quad \sum_i x_i c_i \le \frac{C}{\alpha}$$

and the second step is to choose the weights on them

$$\max_{w} \lambda_2\left(L(w)\right)$$
$$\text{s.t. :} \sum_i w_i c_i \le C, \quad \alpha y_i \le w_i \le \beta y_i, \quad y = x_{\text{opt}}.$$

Later, it will be seen that, if this approach is used, the result will not be optimal.

C. Relaxation

The relaxation (R) of the problem is obtained by allowing noninteger values for x

$$\forall (i,j) \in E, \quad x_{ij} \in [0,1].$$

This is the same as choosing $w \in [0, \beta]$ without variables x and k. However, these variables will be necessary in order to be able to get the integer solution from this relaxed one. It is noticed that the solution of (R) is a concave function of k.

D. Concavity in k

The first important property is that the solution of (R) is concave in k, which will be used in the golden section search algorithm in Algorithm 1. More precisely, Λ is defined such that

$$\Lambda(k) = \max_{x,w} \lambda_2\left(L(w)\right) \quad \text{s.t.} \begin{cases} \sum_i x_i = k \\ x_i \in [0,1] \\ \sum_i w_i c_i \le C \\ \alpha x_i \le w_i \le \beta x_i. \end{cases}$$

Property 3: $\Lambda(k)$ is a concave function.

This property is important since it shows that the maximization of the algebraic connectivity is related to the number of edges in the graph. By solving the problem for very few values of k, a good knowledge on k_{opt} can be obtained.

Proof: Consider k_1 and k_2 in $\mathbb{R}$ and $\gamma \in [0,1]$ such that $\Lambda(k_i) > 0$ for $i = 1, 2$. The idea is to use the fact that $w \to \lambda_2(L(w))$ is concave (please see Property 2).

$$
\begin{aligned}
&\Lambda\left(\gamma k_1 + (1-\gamma) k_2\right)\\
&= \max_{x,w} \lambda_2 \quad \text{s.t.} \begin{cases} \sum_i x_i = \gamma k_1 + (1-\gamma) k_2 \\ x_i \in [0,1] \\ \sum_i w_i c_i \le C \\ \alpha x_i \le w_i \le \beta x_i \end{cases}\\
&\ge \max_{x,w} \lambda_2 \quad \text{s.t.} \begin{cases} \sum_i x_i^{(1)} = k_1, & \sum_i x_i^{(2)} = k_2 \\ x_i^{(j)} \in [0,1], & j = 1,2 \\ x = \gamma x^{(1)} + (1-\gamma) x^{(2)} \\ \sum_i w_i c_i \le C \\ \alpha x_i \le w_i \le \beta x_i \end{cases}\\
&\ge \max_{x,w} \lambda_2 \quad \text{s.t.} \begin{cases} \sum_i x_i^{(1)} = k_1, & \sum_i x_i^{(2)} = k_2 \\ x_i^{(j)} \in [0,1], & j = 1,2 \\ w = \gamma w^{(1)} + (1-\gamma) w^{(2)} \\ \sum_i w_i c_i \le C \\ \alpha x_i^{(j)} \le w_i^{(j)} \le \beta x_i^{(j)} \end{cases}\\
&\ge \gamma \max_{x,w} \lambda_2 \quad \text{s.t.} \begin{cases} \sum_i x_i = k_1 \\ x_i \in [0,1] \\ \sum_i w_i c_i \le C \\ \alpha x_i \le w_i \le \beta x_i \end{cases}\\
&\quad + (1-\gamma) \max_{x,w} \lambda_2 \quad \text{s.t.} \begin{cases} \sum_i x_i = k_2 \\ x_i \in [0,1] \\ \sum_i w_i c_i \le C \\ \alpha x_i \le w_i \le \beta x_i \end{cases}\\
&\ge \gamma \Lambda(k_1) + (1-\gamma)\Lambda(k_2)
\end{aligned}
$$

which proves that Λ is concave in k. ∎

III. Small-Scale Air Transportation Networks

A. Exact Solution for Small Networks

If all weights have to be chosen within an interval, the problem becomes a convex optimization problem and it can be solved using an SDP solver. The idea here is to try all the possible configurations for which all the weights are either 0 or in $[\alpha, \beta]$. Then, each configuration can be independently optimized and the one that leads to the best result can be found.

Consider that n nodes are chosen randomly. There are $m = (n(n-1))/2$ edges and 2^m configurations to test. For each configuration, if Y is the set of the edges that are actually in the graph, the following problem needs to be solved:

$$
\max_w \lambda_2\left(L(w)\right) \quad \text{s.t.} \begin{cases} \sum_{ij} w_{ij} c_{ij} \le C \\ \alpha \le w_{ij} \le \beta, & \forall (i,j) \in Y \\ w_{ij} = 0, & \forall (i,j) \notin Y. \end{cases}
$$

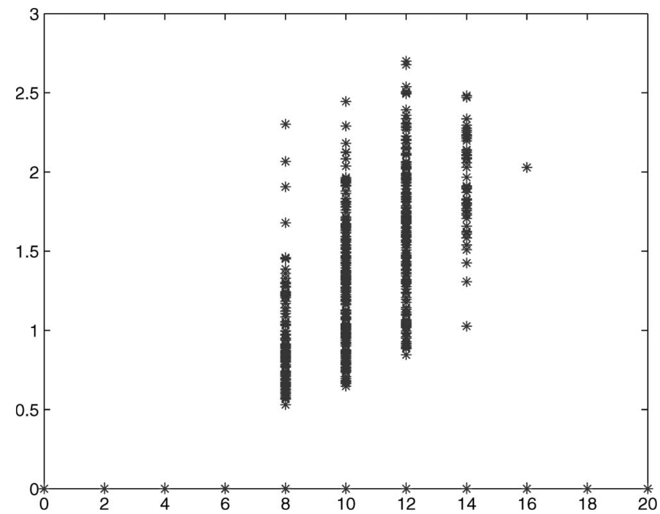


Fig. 2. Results of (k, λ_2) for all the configurations of $n = 5$ and $C = 6.5$.

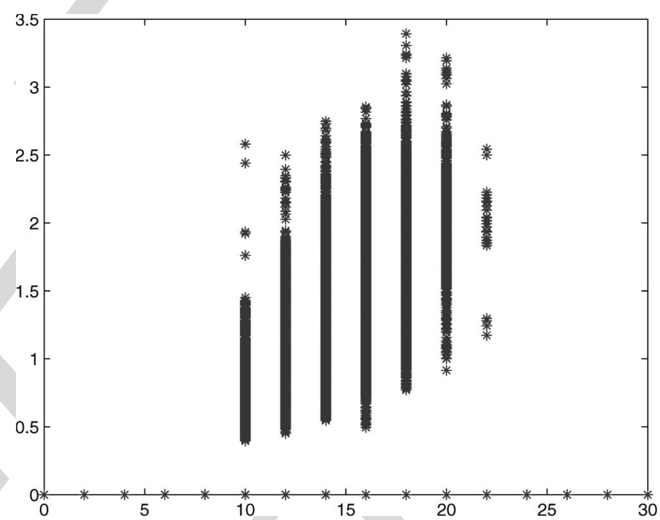


Fig. 3. Results of (k, λ_2) for all the configurations of $n = 6$ and $C = 8$.

This can be done by solving the SDP corresponding to the weight optimization problem (see [27] for details)

$$
\min_w \sum_i w_i c_i \quad \text{s.t.} \begin{cases} \alpha \le w_i \le \beta, & \forall (i,j) \in Y \\ w_{ij} = 0, & \forall (i,j) \notin Y \\ L(w) \succeq I - \frac{1}{n} e e^T. \end{cases}
$$

It becomes impossible to exactly solve the problem when n is large. Therefor, the authors assume n to be small in this section.

1) Small-Scale Exact Solution Results: For each configuration, the number of edges k in the graph is computed. The results of (k, λ_2) are plotted in Figs. 2 and 3 for two different networks.

It is noticed that the best connectivity is not reached at the maximum number of edges; therefor, the choice of the edges and the choice of the weights are not independent.

It is also noticed that, unlike the continuous case, the discrete shape given by $k \to \Lambda(k)$ in Figs. 2 and 3 is not exactly concave. However, it has almost the shape of a concave function; hence, the authors will be able to consider it concave in the approximate case later on.

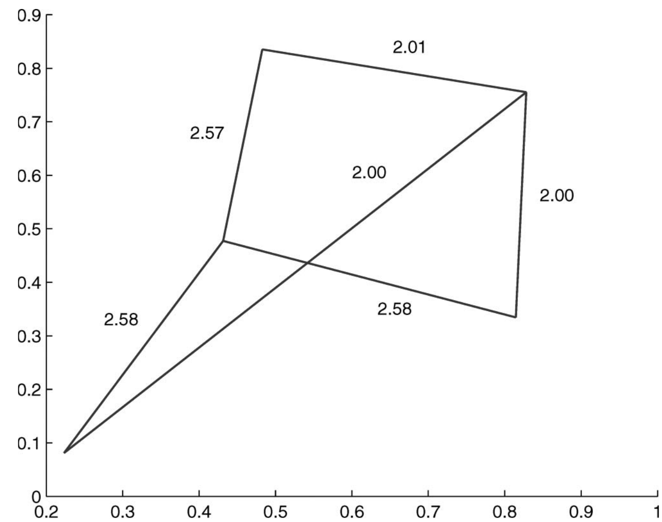


Fig. 4. Optimal network for $n = 5$ with the weights. $p = 0$, $\alpha = 2$, and $C = 6.5$.

The exact solution for a network of size $n = 5$ is shown in Fig. 4. As it is impossible to exactly solve this problem for networks with a larger size, an algorithm is going to be designed, which solves it approximately. The main idea is to use the quasi-concave shape of function $\Lambda(k)$.

For the practical problem with a larger size, the first step is to choose a value for k. The authors are able to solve the relaxed version (R) of the problem where x is a noninteger variable. Then, the result can be rounded to obtain a feasible solution for the original problem (P).

2) Maximum Number of Edges: Because of the minimum value α for the weights, there exists a limit in the number of edges in the graph. Here, the maximal number of edges $k_{\lim}$ needs to be found. Regardless of the performance of the network, edges are added until the operating cost constraint is reached. Consider that $w = \alpha x$, which is the minimum weight, and the problem is to solve a trivial form of the knapsack problem

$$\max_{x \in \{0,1\}} \sum_i x_i$$
$$\text{s.t.} \sum_i x_i c_i \leq \frac{C}{\alpha}.$$

Indeed, if $k_{\lim}$ is the solution of this problem, it can be guaranteed that there will not be any solution of $k > k_{\lim}$.

When the cost c_i is sorted by increasing order, it can be obtained that

$$\sum_{s=1}^{k_{\lim}} c_s \approx \frac{C}{\alpha}$$

which, for large-enough n, can be approximated by the following formula:

$$\int_1^{k_{\lim}} g(s) ds = \frac{C}{\alpha}$$

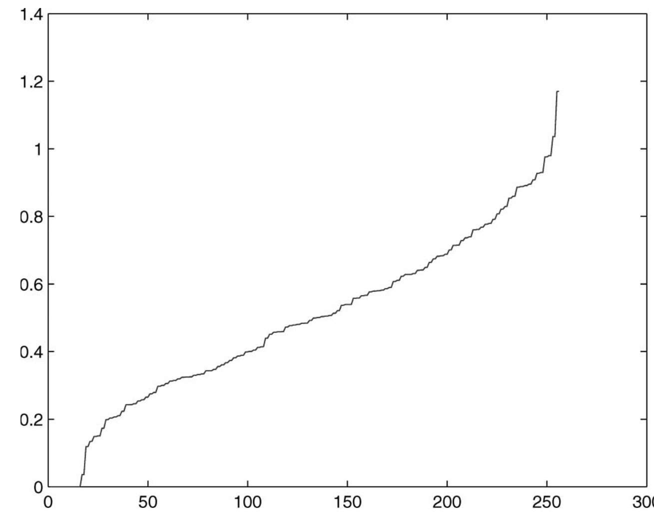


Fig. 5. Function g for random points in $[0, 1]^2$.

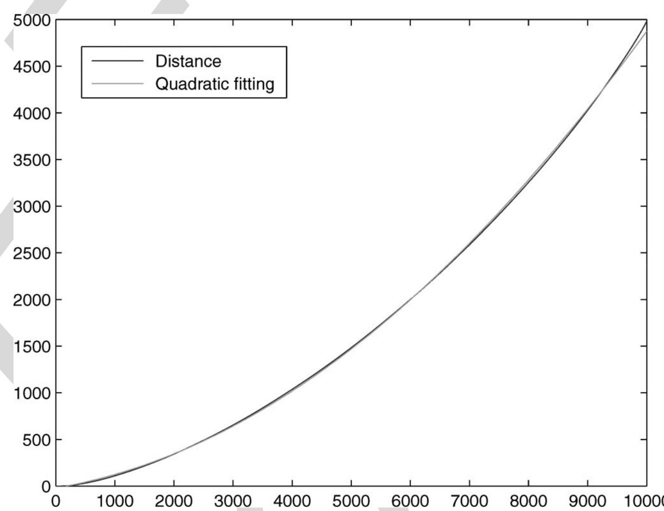


Fig. 6. C as a function of $k_{\lim}$ with the quadratic fitting.

where

$$\forall s \in [1, k_{\lim}], \quad g(s) = c_{\lfloor s \rfloor}.$$

If the n nodes are randomly chosen in a square, function g is very close to a linear function (except at the very beginning and at the very end). This can be verified in Fig. 5. Using this information, it can be obtained that

$$a_2 k_{\lim}^2 + a_1 k_{\lim} + a_0 = \frac{C}{\alpha} \tag{3}$$

where a_0, a_1, and a_2 are constant parameters. Finally

$$k_{\lim} = \frac{-a_1 + \sqrt{{a_1}^2 - 4a_2\left(a_0 - \frac{C}{\alpha}\right)}}{2a_2}. \tag{4}$$

It is shown in Fig. 6 that the quadratic result obtained by (4) is a very good approximation.

B. SDP Formulation

To solve the larger size problem, the relaxation of the problem is expressed as an SDP that will be solved efficiently. When v is not normalized, recall (1) in Property 1, which can be then transformed as

$$\begin{cases} \lambda_2 = \max_\lambda \lambda \\ \lambda v^T v \leq v^T L v \\ \forall v \in \mathbb{R}^n, \qquad v^T e = 0. \end{cases}$$

Variable μ is added, which allows any $v \in \mathbb{R}^n$

$$\begin{cases} \lambda_2 = \max_{\lambda,\mu} \lambda \\ \forall v \in \mathbb{R}^n, \qquad v^T(\mu e e^T)v + v^T L v - \lambda v^T v \geq 0. \end{cases}$$

It can be written using Loewner's order [33]

$$\begin{cases} \lambda_2 = \max_{\lambda,\mu} \lambda \\ \mu e e^T + L - \lambda I \succeq 0. \end{cases} \tag{5}$$

The relaxation of the problem that needs to be solved is

$$\max_{x,w,k} \lambda_2 \left(L(w)\right) \quad \text{s.t.} \begin{cases} \sum_i x_i = k \\ x_i \in [0,1] \\ \sum_i w_i c_i \leq C \\ \alpha x_i \leq w_i \leq \beta x_i \end{cases}$$

which now becomes with (5)

$$\max_{x,w,k,\lambda,\mu} \lambda \quad \text{s.t.} \begin{cases} \sum_i x_i = k \\ x_i \in [0,1] \\ \sum_i w_i c_i \leq C \\ \alpha x_i \leq w_i \leq \beta x_i \\ \mu e e^T + L - \lambda I \succeq 0. \end{cases} \tag{6}$$

This problem is an SDP since there is a semidefinite constraint and all the other constraints are linear. It can be solved efficiently by an SDP solver. SeDuMi [34] is used in this paper.

1) Optimality Conditions: The primal SDP is

$$\max_{x,w,k,\lambda,\mu} \lambda \quad \text{s.t.} \begin{cases} \sum_i x_i = k \\ x_i \in [0,1] \\ \sum_i w_i c_i \leq C \\ \alpha x_i \leq w_i \leq \beta x_i \\ \mu e e^T + L - \lambda I \succeq 0. \end{cases}$$

The variables are rescaled by dividing w, k, and x by λ and C. The dual SDP problem is

$$\min_{x,w,k,\lambda} \sum_i w_i c_i \quad \text{s.t.} \begin{cases} \sum_i x_i = k \\ x_i \in [0,1] \\ \alpha x_i \leq w_i \leq \beta x_i \\ L \succeq I - \frac{1}{n} J \end{cases}$$

where J is the all-one matrix. When x_{ij} is relaxed (please see Section II-C), the relaxed dual SDP formulation is

$$\min_{x,w,k,\lambda} \sum_i w_i c_i \quad \text{s.t.} \begin{cases} 0 \leq w_i \leq \beta \\ L \succeq I - \frac{1}{n} J. \end{cases}$$

X is the matrix of the operating costs. The matrix format relaxed dual SDP is therefor

$$\max_X \left\langle I - \frac{1}{n} J, X \right\rangle \quad \text{s.t.} \begin{cases} X \succeq 0 \\ \langle E, X \rangle = c_{ij} \end{cases}$$

where E is the matrix with $E_{ii} = E_{jj} = 1$ and $E_{ij} = E_{ji} = -1$.

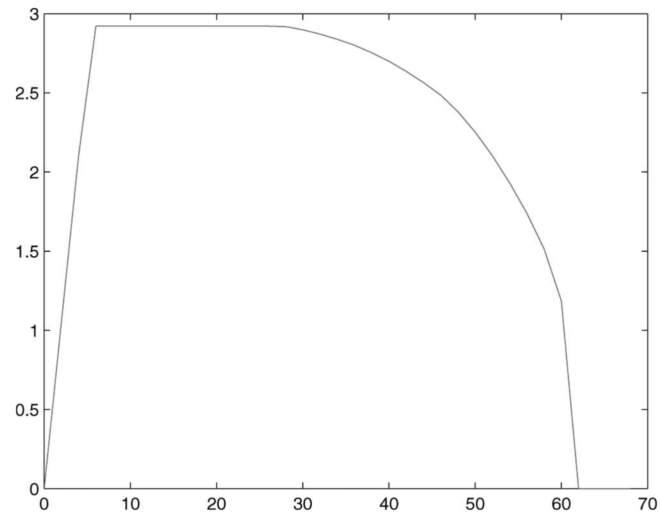


Fig. 7. Upper bound: $\lambda_2 = f(k)$.

The Karush–Kuhn–Tucker optimality conditions are

$$\begin{cases} SX = XS = 0 \\ S \succeq 0, \qquad X \succeq 0 \\ \langle E, X \rangle = c_{ij} \\ L - I + \frac{1}{n} J = S. \end{cases}$$

If $w_i = w$ for all i, it can be obtained that

$$S = (nw - 1)\left(I - \frac{1}{n} J\right).$$

When $w = 1/n$, $S = 0$ and the conditions are satisfied.

Reciprocally, if the optimality conditions are satisfied, it can be obtained that

$$\left(L - \left(I - \frac{1}{n} J\right)\right) X = 0.$$

X has rank n; hence

$$L = I - \frac{1}{n} J.$$

Thus, w is constant for all i and $w = 1/n$.

For edge i and edge j, the optimality condition is finally obtained

$$\begin{cases} \forall (i,j), w_i = w_j \\ \sum_i w_i c_i = C. \end{cases}$$

2) Upper Bound: With the SDP formulation, the relaxation can now be solved. When the values of the optimal connectivity for different values of k are computed, the upper bound is plotted in Fig. 7.

The relaxed problem reached its maximum for several values of k contained in an interval $[k_{\min}, k_{\max}]$. Indeed, the optimality conditions give

$$\begin{cases} \forall (i,j), w_i = w_j \\ \sum_{i=1}^m w_i c_i = C. \end{cases}$$

All the weights are equal and their value is $\forall i,\ w_i = (C/(\sum_j c_j)) = \Omega$. If $w = \beta x$, all the elements of x are equal and their value is $\forall i,\ x_i = (k/(n^2-n))$, which leads to

$$k_{\min} = \frac{\Omega(n^2-n)}{\beta}.$$

By doing the same computation, it is proved that the optimal value is also reached with

$$k_{\max} = \frac{\Omega(n^2-n)}{\alpha}$$

and $\forall k \in [k_{\min}, k_{\max}]$, the optimal value is achieved.

However, it needs to be pointed out that when the solution is rounded, infeasible solution may appear. For example, if $k = k_{\min}$, there is often no solution since

$$\sum_i x_i = \frac{n\Omega}{\beta} < n-1$$

if $(\Omega/\beta) \ll 1$, which is often the case. In addition, because at least $n-1$ edges are needed to connect an n node graph, there is no positive solution. Therefor, the upper bound is not a very good bound for small values of k.

C. Rounding Techniques for SDP Solution

1) Description of the Methods: In this section, suppose that the relaxed optimal solution s_0 has been found. k edges are going to be selected from s_0, which means that $x_i = 1$ for k values and $x_i = 0$ for the others. There are several ways to do so. The methods that have been studied and implemented are listed.

1) *Greedy:* Choose the k biggest elements $s_0(x_i)$ in the relaxed solution. Then, find the optimal weights by solving the corresponding SDP.

2) *Random fast:* Randomly choose the rounding. For each $i \in \{1, \ldots, m\}$, $x_i = 1$ with probability $s_0(x_i)$ and $x_i = 0$ otherwise. Then, the weights are affected with the following value:

$$\frac{s_0(w_i)}{s_0(x_i)}.$$

These two steps are repeated many times. The average value $\overline{x_i}$ of x_i is $s_0(x_i)$; therefor

$$\overline{\sum_i x_i} = \sum_i s_0(x_i) = k$$

and for the same reason

$$\overline{\sum_i w_i c_i} = \sum_i \frac{s_0(w_i)}{s_0(x_i)} \overline{x_i} c_i = \sum_i s_0(w_i) c_i \leq C.$$

Thus, on average, the solution satisfies the constraints. At the end, keep the best solution that satisfies all the constraints.

3) *Random:* In addition, randomly choose the rounding. If $\sum_i x_i = k$, which is the case on average, evaluate the weights by solving the SDP formulation. The steps are repeated several times, and keep the best value.

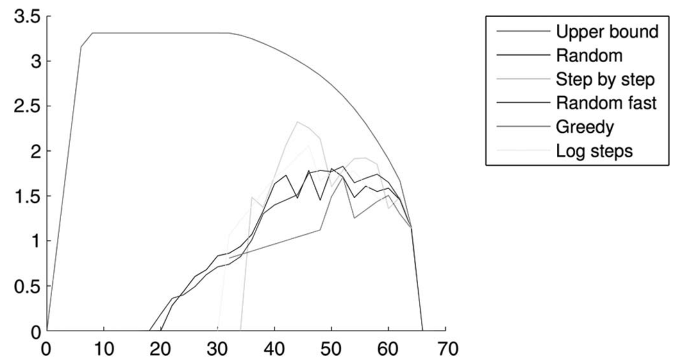


Fig. 8. λ_2 as a function of k. The results from several rounding methods are represented for a 20-node graph.

4) *Step by step:* Select the biggest element $s_0(x_i) < 1$ and affect its value to 1 in the SDP formulation. Then, solve the SDP again and repeat k times for these two steps.

5) *Log step by step:* This is the same idea as the "step by step" except that, at each step, choose the best half of the remaining elements. Thus, there are only $\log(k)$ SDPs that have to be solved.

2) Numerical Results: The simulation is set up with 20 nodes randomly generated in a square. The results are presented in Fig. 8 with λ_2 as a function of k. The upper bound obtained by the relaxation is plotted, as well as all the rounding techniques.

It turns out that some techniques may fail to find a solution. In this case, the corresponding values are removed from the figure.

It can be seen that the algorithms can achieve the upper bound at the maximum number of edges k_{lim}. This shows that any algorithm based on edge addition without considering the variable weights is not adapted.

Pros and Cons: Each of the methods presented has some advantages and drawbacks. The one that gives the best result is the *step-by-step* method. The fastest is *random fast*. In addition, the method that gives the best tradeoff between speed and value is *log step by step*.

D. Relaxed SDP With Rounding Algorithm

This algorithm is used to solve relatively small-scale problem when the exhaustive search described in Section III-A fails.

1) Golden Section Search: For a given k, a well-connected network with k edges can now be found. Instead of testing all the possible values of k, the search can be speeded up by considering that algebraic connectivity is a concave function of k.

This approximation leads to better results with rounding methods that have good regularity. For large networks, the rounding methods with lower regularity can be used. Instead of computing the value for a given k, a local average value is computed based on three values

$$\forall k \in \mathbb{N}, \quad \tilde{f}(k) = \frac{f(k-1) + f(k) + f(k+1)}{3}.$$

As only the value of the connectivity for integer values of k can be computed, it is not possible to use continuous optimization principles. Thus, the golden section search [35] is adopted. It consists in creating a decreasing set of intervals containing the optimal value

$$\forall i \in \mathbb{N}, \quad [a_{i+1}, b_{i+1}] \subset [a_i, b_i]$$

and $k_{\rm opt} \in [a_i, b_i]$. Two test values $c_i < d_i$ in $[a_i, b_i]$ facilitate the search. The rules used to update the interval are: 1) $f(c_i) < f(d_i) \Rightarrow [a_{i+1}, b_{i+1}] = [c_i, b_i]$ and 2) $f(c_i) > f(d_i) \Rightarrow [a_{i+1}, b_{i+1}] = [a_i, d_i]$.

At each step, only one new value of f needs to be computed. This new value is usually chosen so that the test values are at the golden ratio $\phi = (1 + \sqrt{5})/2$. Here, the value is rounded to get an integer. This method allows, on average, to divide the length of the interval by ϕ at each step.

2) Relaxed SDP With Rounding Algorithm: All the steps of the algorithm are summed up. The SDP is solved using the SeDuMi solver [34] and the golden section search. This algorithm that leads to the approximation of the optimum is listed in Algorithm 1.

Algorithm 1 Relaxed SDP with step-by-step rounding

Initialize a, b and d
while $b - a > 2$ **do**
  Choose c in $\{(a + d/2), (d + b/2)\}$
  Solve the relaxed SDP with $k = c$
  for $p = 1$ to k **do**
    $j \leftarrow \arg\max_i \{x_i | x_i < 1\}$
    Impose $x_j = 1$
    Solve the SDP
  end for
  if $f(c) < f(d)$ **then**
    $a \leftarrow c$
  else
    $b \leftarrow d, d \leftarrow c$
  end if
end while
return λ_2

The complexity of this algorithm can be analyzed, which depends on several parameters of the problem. The algorithm uses the step-by-step rounding technique and requires to solve $k + 1$ SDPs for each value of k selected. Each step has a different value for k, and most of them are close to $k_{\rm opt}$.

In addition, there are U such steps. U is defined by $k_{\rm lim}\phi^{-U} = 1$ since at each step the length of the interval is divided by ϕ. It is obtained that

$$U = \frac{\log(1/k_{\rm lim})}{\log(1/\phi)}.$$

Complexity T also needs to be considered to solve the SDP. T is a polynomial in the size of the entry, which is equivalent to n^2. Therefor, T is a polynomial function of n.

Therefor, the complexity of the whole algorithm can be approximated by $O(k_{\rm opt}UT)$.

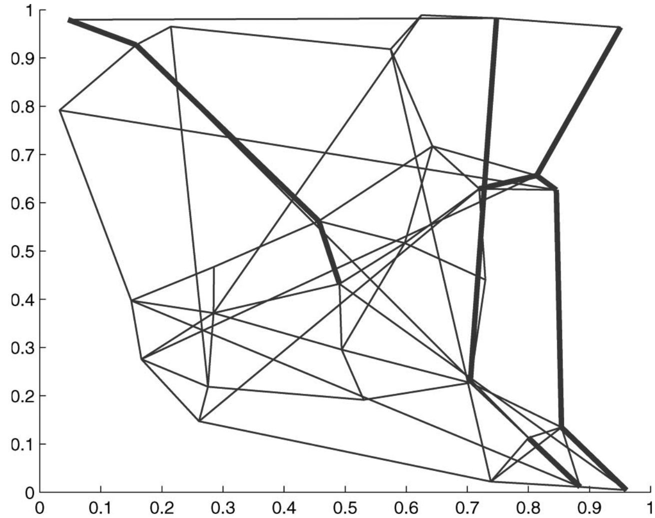


Fig. 9. Optimal result for a 30-node graph.

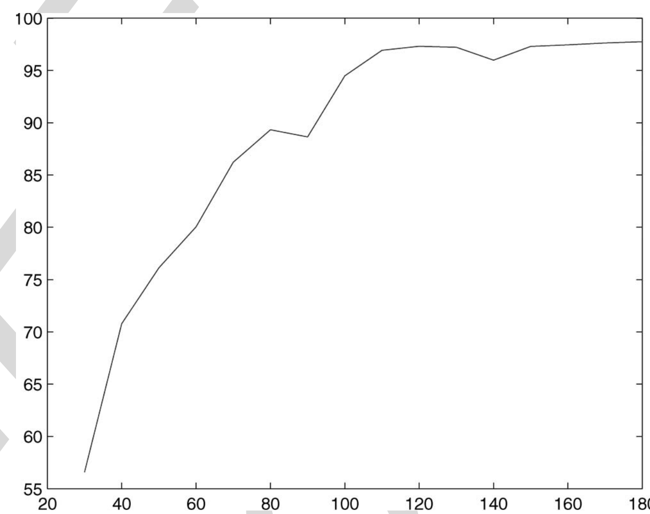


Fig. 10. Values of r for different values of C.

E. Numerical Results

1) Optimal Network: In order to test the full algorithm, a set of random nodes is generated in a square. An example of the optimal network for 30 nodes is shown in Fig. 9.

The edges that have a weight greater than the lower bound α are represented with a thicker line. In the example in Fig. 9, there are ten edges with a larger weight value than α.

2) Efficiency: Now, the efficiency of the result is going to be evaluated by comparing the optimal algebraic connectivity to the upper bound. For a given set of nodes, the problem is solved for different values of the total operating cost budget C and the percentage of the solution is computed compared to the bound

$$r = 100 \times \frac{val(P)}{val(R)}\%.$$

The result, as illustrated in Fig. 10, shows that for small values of C, the best result found is very far from the upper

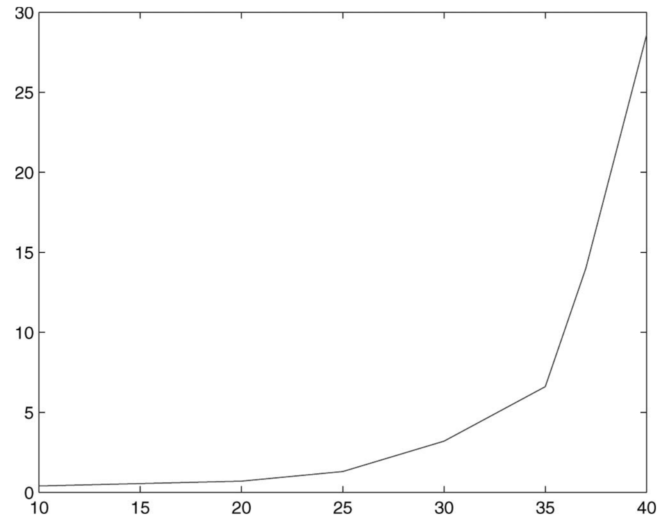


Fig. 11. Time (in seconds) to solve the SDP formulation of the problem for a given number of n nodes.

bound. However, when increasing C, the objective value of the problem P quickly increases to reach the value of its relaxed problem R.

IV. Large-Scale Air Transportation Networks

A. Necessity

The method in the previous section is going to be applied to large networks. The most time-consuming computation in the process is solving the SDP. Fig. 11 shows the computational time of solving one SDP for n nodes. It is observed that the running time increases very rapidly. In fact, for n nodes, there are $n(n-1)+2 \sim n^2$ variables in the SDP. As several SDPs need to be computed in order to solve the problem, it becomes impossible for $n \geq 35$ on the authors' workstation.

However, it is necessary to get some results for large values of n because real networks are usually large. For example, the air transportation network contains several hundred nodes when considering the entire USA.

B. Cluster Decomposition

Since the key factor for operating cost for each link is the route distance (a longer distance route consumes more fuel), the idea in this section is to divide the airports into $g \in \mathbb{N}$ clusters based on the distance between the nodes. These clusters can be solved independently with the relaxed SDP method (Algorithm 1) developed in the previous section and can be connected afterward.

To connect the cluster, choose k major nodes in each cluster that will be connected to each other. Then, the problem P has to be solved for these $g \times k$ nodes, except that links between two nodes from the same cluster are not allowed; hence, the graph is not complete.

At the end, $g+1$ problems of type (P) need to be solved to get the final result. Fig. 12 shows the idea of the decomposition into several clusters and the selection of major nodes.

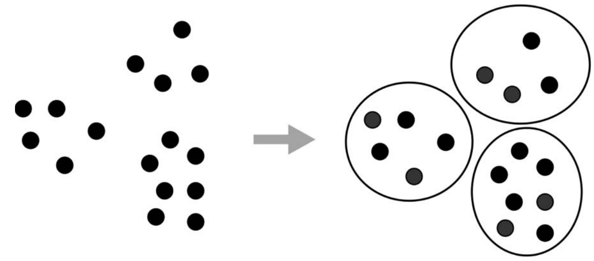


Fig. 12. Set of 16 nodes separated in three clusters with two major nodes in each cluster (in red).

There are several parameters whose values have to be chosen to apply this idea. First, choose the number of clusters and how many major nodes are used in each cluster to connect to other clusters. Second, choose which nodes are kept as major nodes among each cluster. Naturally, it is decided here to take the airports that have the largest traffic demand.

In addition, to solve the problem for each small problem $1 \leq i \leq g+1$, the value of the maximum operating cost budget C_i in each cluster has to be chosen. A natural option is to choose C_i proportional to the sum s_i of all the costs of the edges in the cluster i and such that $\sum_{i=1}^{g+1} C_i = C$

$$s_i = \sum_{(x,y)\in E} c_{xy},$$

$$C_i = \frac{s_i}{\sum_{j=1}^{g+1} s_j} C.$$

The separation of the nodes into several clusters is made by k-means algorithm [36]. This algorithm has the advantages of being fast, easy to implement, and generally giving good results.

To sum up the method described above, the full cluster decomposition algorithm is listed in Algorithm 2.

Algorithm 2 Large-scale cluster decomposition

1: Initialize g, C_i
2: k-means algorithm gives g clusters
3: **for** $p = 1$ to g **do**
4:    Solve the cluster problem (with Algorithm 1)
5: **end for**
6: Solve the major node problem (Algorithm 1)
7: Build the resulting network
8: **return** $\lambda_2(L)$

C. Evaluation of Efficiency

The goal here is to show that if all $g+1$ clusters are well connected, the resulting graph is well connected too. This depends on the values of some parameters that characterize how each cluster is linked to the others.

g clusters are considered. Each cluster has n nodes and k of its nodes are used to connect to other clusters. Let G be the matrix of the graph and F be the vector defined by the

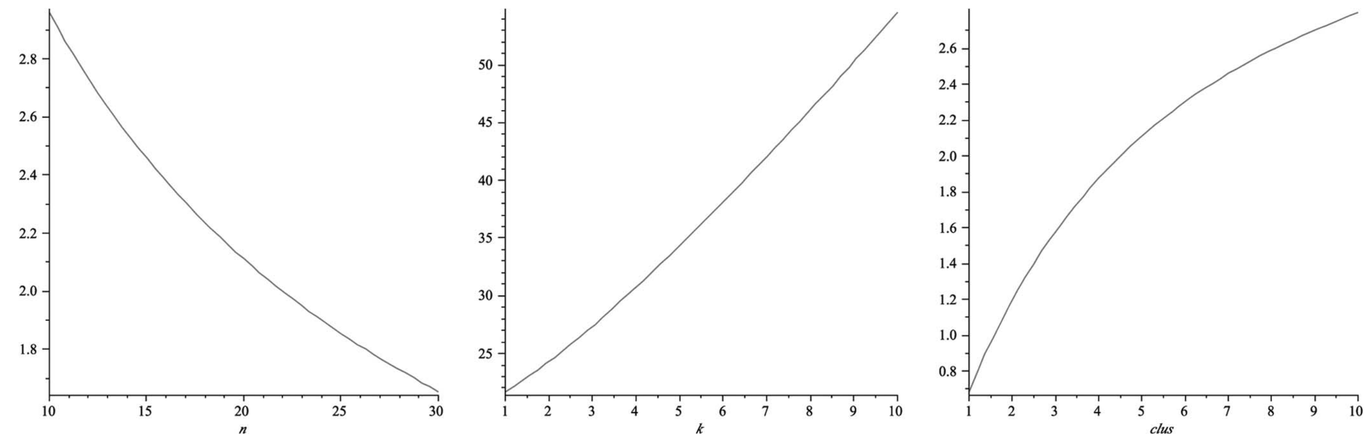


Fig. 13. (a) $\lambda_2 = f(n)$; (b) $\lambda_2 = f(k)$; (c) $\lambda_2 = f(g)$.

expression below. If, for instance, $g = 3$, matrix G can be put into the following form:

$$G = \left(\begin{array}{ccc|ccc|ccc} & & & E & 0 & 0 & E & 0 & 0 \\ & A_1 & & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline E & 0 & 0 & & & & E & 0 & 0 \\ 0 & 0 & 0 & & A_2 & & 0 & 0 & 0 \\ 0 & 0 & 0 & & & & 0 & 0 & 0 \\ \hline E & 0 & 0 & E & 0 & 0 & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & & A_3 & \\ 0 & 0 & 0 & 0 & 0 & 0 & & & \end{array}\right)$$

$$F = \left(\begin{array}{c} \alpha e \\ \beta e \\ \beta e \\ \hline 0 \\ 0 \\ 0 \\ \hline -\alpha e \\ -\beta e \\ -\beta e \end{array}\right)$$

with the following notation. e is the all-one vector. α and β are constants that will be computed in the next paragraph. A_1, A_2, and A_3 represent the adjacency matrices of the three clusters. E is a $k \times k$ matrix with all elements equal to 1.

1) Fiedler Vector: The Fiedler vector is the vector solution of the minimization problem

$$\min_{x \in \mathbb{R}^n} \left\{ x^T L x \| \|x\| = 1, \quad xe = 0 \right\}.$$

It is known to be an indicator on how to split a graph into two smaller graphs. In fact, the nodes that have the same sign in this vector form a cut of the graph (see [37]).

Here, the optimal cut will naturally be found between two clusters. Since some of the nodes play the same role, the Fiedler vector has a shape close to F where α and β are constants that need to be determined.

This assumption has been verified by numerical experiments and it seems to be a very good approximation of the real Fiedler vector.

2) Computing the Connectivity: Consider that the Fiedler vector has the form of F and matrices are full, which means all nondiagonal elements are equal to 1. The matrix products give

$$\lambda_2 = F^T L F,$$
$$\lambda_2 = 2k\alpha X + 2(n-k)\beta Y$$

with

$$X = \alpha\,(n - 1 + k(g-1)) - (k-1)\alpha - (n-k)\beta + k\alpha,$$
$$Y = \beta(n-1) - k\alpha - (n-k-1)\beta.$$

It is also known that $\|F\| = 1$; thus

$$2k\alpha^2 + (2n-2k)\beta^2 = 1,$$
$$\alpha = \sqrt{\frac{1-(2n-2k)\beta}{2k}}.$$

Substitute α with this expression and β is given by the

$$\frac{d\,\lambda_2}{d\,\beta} = 0. \tag{7}$$

With a computation software package like Maple, this gives us the expression of function f such that

$$\lambda_2 = f(n,k,g).$$

3) Resulted Curves: By solving (7), the values of α, β, and λ_2 are obtained. The following figures have been obtained with Maple. Among the three parameters k, g, and n, fix two of them and let the third one vary to see its influence on connectivity.

Fig. 13 provides a clearer idea on how to choose the value of each parameter. For example, connectivity is almost linear regarding k but has a concave shape when represented as a function of the number of clusters g.

There exists a tradeoff: If g is too large, kg will be too large to be solved. On the contrary, if g is too small, each of the cluster will have too many nodes to be solved.

4) Numerical Results: The data used are the 100 largest cities in the United States. Fig. 14 shows the 100 biggest cities without any link. The cluster decomposition is used to divide these 100 cities into $g = 5$ groups. In each group, we selected $k = 5$. The lower bound $\alpha = 2$ and the upper bound

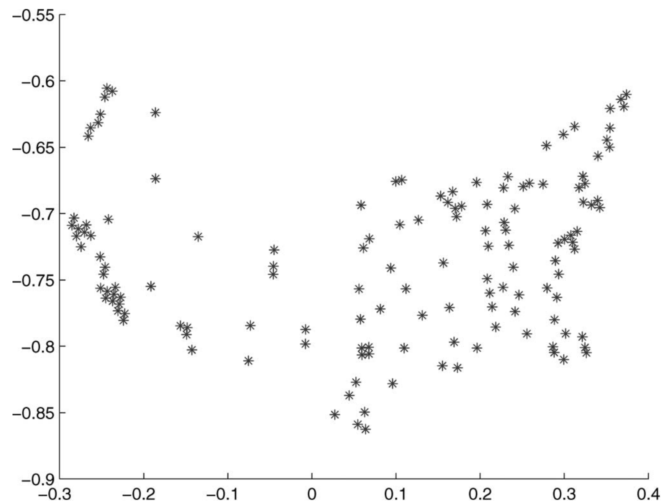


Fig. 14. Shown are the 100 largest cities in the USA.

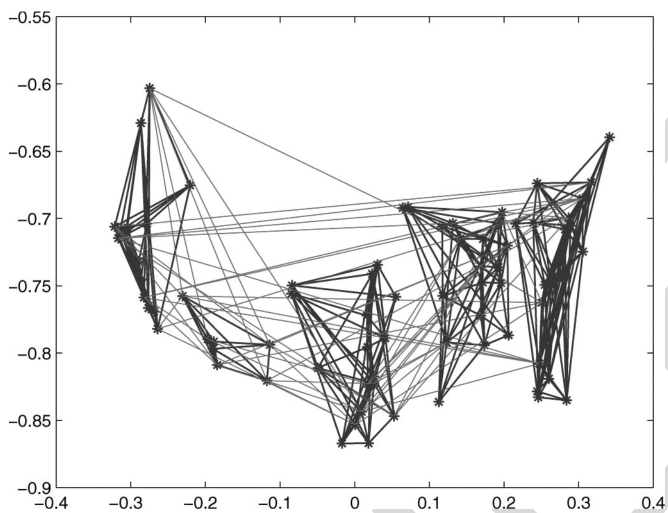


Fig. 15. Result for the 100 largest cities in the USA.

$\beta = 10$. The total running time is 317 s with MATLAB on our workstation. The algebraic connectivity that we have achieved is $\lambda_2 = 2.6$.

The optimal network found is illustrated in Fig. 15. The blue lines represent the edges inside each cluster and the red lines represent the edges that connect nodes from different clusters.

In a real network, most airlines use spoke-hub planning, in which the regional airports are clustered and connected to their regional hub airport. Fig. 15 shows the same behavior that regional airports are clustered together. In a real network in the United States, the number of the major nodes k for each cluster is usually 1, which means that the hub airport is the only major node for its cluster. In a real network in Europe, the number of the major nodes k is usually bigger than 1. In fact, the number k in a European network is those hub airports in each country. For example, Air France has hubs at Paris and Lyon $(k = 2)$ and Air Berlin has its hubs at Berlin, Düsseldorf, Hamburg, and Munich $(k = 4)$. In summary, the result in Fig. 15 is a practical design, and more importantly, the computation is very efficient for large-scale network planning.

V. Directed Air Transportation Network

A. From Undirected Graph to Directed Graph

In this section, the methods are extended in directed graphs. In order to be consistent with the undirected case, the graphs need to be balanced, which means that the number of aircraft that comes in is the same as the number of aircraft that leaves the airport

$$\forall i \in \{1, \ldots, n\}, \quad \sum_{j=1}^{n} w_{ij} = \sum_{k=1}^{n} w_{ki}.$$

It is clear that the set of undirected graphs is included in the set of directed balanced graphs. Therefor, the results should be at least as good as in the previous section.

Definition: According to [38], if $\Omega = \{x \in \mathbb{R}^n, xe = 0, \|x\| = 1\}$, the definition of the algebraic connectivity can be extended for directed balanced graphs with

$$\min_{x \in \Omega} x^T L x = \lambda_2 \left(\frac{1}{2}(L + L^T) \right).$$

Property 4: In the directed case and with this definition of the algebraic connectivity, the upper bound given by the continuous relaxation is the same as in the undirected case.

Proof: Given the optimum directed balanced graph in the relaxed problem and its incidence matrix G, H can be created

$$H = \frac{G + G^T}{2}$$

where H is symmetric and satisfies all the constraints of the problem since they are linear. In addition with the definition of the connectivity for directed graphs, the connectivity of H is clearly the same as the connectivity of G.

Since it is also known that undirected graphs are a subset of directed balanced graphs, the bounds are equal in both cases. This will allow us to easily evaluate the improvement brought by directed graphs. ∎

B. Results

The same method is used as in Section III. The results are impressively better with directed graphs, as shown in Fig. 16. The best value for the directed balanced case almost reached the upper bound. It is also noticed that the optimal result has less edges than the optimal network in the undirected case (an edge in the undirected case is counted in both ways).

It is shown in Fig. 17 that most of the edges in the optimal solution are oriented in only one way, which shows that the solution is very different from the undirected case.

However, there is an important drawback. Indeed, two times as many variables are needed for directed networks. Thus, the problem takes a much longer time to be solved and is only applicable on smaller networks.

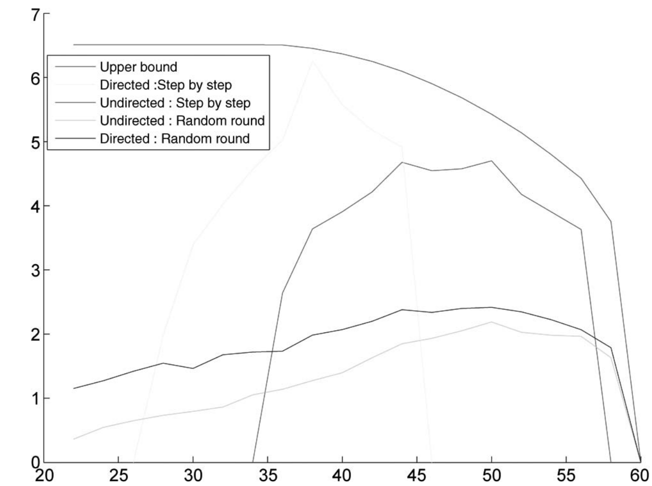


Fig. 16. $\lambda = f(k)$ for the same graph with different approaches.

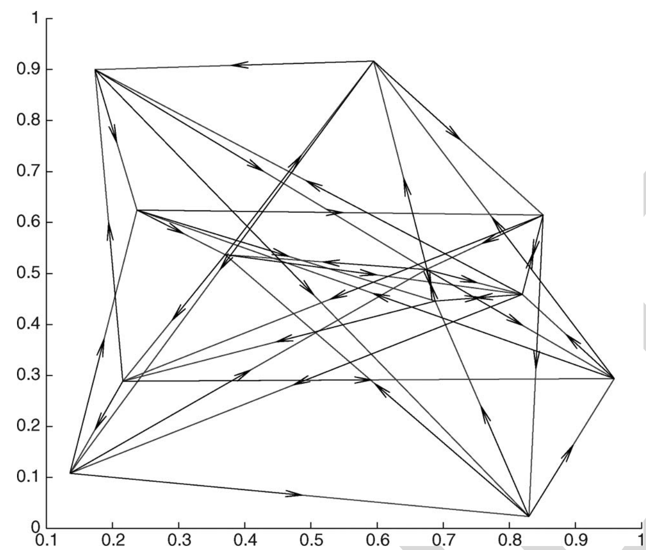


Fig. 17. Optimal directed graph.

C. Failure Case

If an edge or a node is removed, the graph is not balanced anymore. This can cause important problems in practice; hence, the remaining weights in the graph need to be changed to handle this problem.

This operation can be done by using a flow algorithm. The first step is to link the nodes with positive aircraft balance to a virtual source and those with negative balance to a sink. The capacities of these links are equal to the absolute value of the difference in the balance flow for the node. The capacity of the other links of the graph is β.

Then, consider the problem of the maximization of the flow from the source to the sink. This problem can be solved by using Edmonds–Karp algorithm [39], which has efficient complexity, i.e., $O(nm^2)$. This algorithm maintains the balance of the flow at each node.

At the end, the graph is the graph solution of the flow problem when the source and the sink are removed. Figs. 18 and 19 show an example of a graph in which a node is removed,

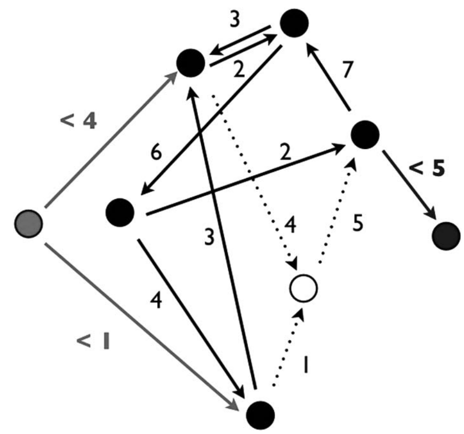


Fig. 18. Example of a graph with a node failure.

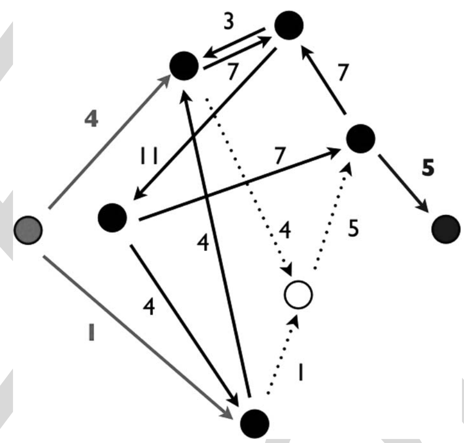


Fig. 19. Example of a graph after maximization of the flow.

before and after maximization of the flow. The graph in Fig. 19 is now balanced by Edmonds–Karp algorithm after removing the source (green node) and the sink (blue node).

VI. Conclusion

In this paper, a new problem concerning the maximization of the algebraic connectivity of a network has been presented and studied. This problem consists of finding both the edges of the graph and their weight assignment under several constraints. First, the problem in small networks is exactly solved, and it is shown that the problem cannot be separated into two already studied independent problems. Then, a relaxed SDP method with step-by-step rounding is presented to solve the problem approximately for better computational efficiency. With the method developed for small-scale network, the cluster decomposition method is proposed to solve the large-scale network problem and successfully find the near-optimal solution for a network of the 100 largest cities in the United States. Finally, the study is extended to directed graphs. Numerical experiments and analysis are performed for all the proposed algorithms. The developed methods are able to model and maximize robustness in air transportation networks for airlines and for the NAS. They also provide an option to improve current ways of the generic network design.

REFERENCES

[1] R. Guimera and L. Amaral, “Modeling the world-wide airport network,” *Eur. Phys. J. B, Condens. Matter Complex Syst.*, vol. 38, no. 2, pp. 381–385, Mar. 2004.

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

[3] E. Vargo, R. Kincaid, and N. Alexandrov, “Towards optimal transport networks,” *J. Syst., Cybern. Inf.*, vol. 8, no. 4, pp. 59–64, Jul. 2010.

[4] International Civil Aviation Organization (ICAO), “Procedures for air navigation services—Rules of the air and air traffic services,” Montréal, QC, Canada, doc 4444-RAC/501, 2010.

[5] P. Kostiuk, D. Lee, and D. Long, “Closed loop forecasting of air traffic demand and delay,” presented at the 3rd USA/Europe Air Traffic Management R&D Seminar, Napoli, Italy, Jun. 2010, Paper 80.

[6] K. Nam and T. Schaefer, “Forecasting international airline passenger traffic using neural networks,” *Logist. Transp. Rev.*, vol. 31, no. 3, pp. 239–251, Sep. 1995.

[7] F. Alamdari and S. Fagan, “Impact of the adherence to the original low cost model on the profitability of low cost airlines,” *Transp. Rev.*, vol. 25, no. 3, pp. 377–392, May 2005.

[8] M. Dresner, J.-S. C. Lin, and R. Windle, “The impact of low-cost carriers on airport and route competition,” *J. Transp. Econ. Policy*, vol. 30, no. 3, pp. 309–328, Sep. 1996.

[9] K. Button and S. Lall, “The economics of being an airport hub city,” *Res. Transp. Econ.*, vol. 5, pp. 75–105, 1999.

[10] T. Oum, A. Zhang, and Y. Zhang, “A note on optimal airport pricing in a hub-and-spoke system,” *Transp. Res. B, Methodol.*, vol. 30, no. 1, pp. 11–18, Feb. 1996.

[11] X.-B. Hu, W.-H. Chen, and E. Di Paolo, “Multiairport capacity management: Genetic algorithm with receding horizon,” *IEEE Trans. Intell. Transp. Syst.*, vol. 8, no. 2, pp. 254–263, Jun. 2007.

[12] X.-B. Hu and E. Di Paolo, “Binary-representation-based genetic algorithm for aircraft arrival sequencing and scheduling,” *IEEE Trans. Intell. Transp. Syst.*, vol. 9, no. 2, pp. 301–310, Jun. 2008.

[13] Y. Eun, I. Hwang, and H. Bang, “Optimal arrival flight sequencing and scheduling using discrete airborne delays,” *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 2, pp. 359–373, Jun. 2010.

[14] Z.-H. Zhan, J. Zhang, Y. Li, O. Liu, S. Kwok, W. Ip, and O. Kaynak, “An efficient ant colony system based on receding horizon control for the aircraft arrival sequencing and scheduling problem,” *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 2, pp. 399–412, Jun. 2010.

[15] *FlightStats-Global Flight Tracker, Status Tracking and Airport Information*, FlightStats, Inc., Portland, OR, USA, 2013, retrieved on March 16, 2013. [Online]. Available: http://www.flightstats.com

[16] Federal Aviation Administration (FAA), “FAA Aerospace Forecasts FY 2008-2025,” Washington, DC, USA, Dec. 2010.

[17] S. Conway, “Scale-free networks and commercial air carrier transportation in the United States,” in *Proc. 24th Int. Congr. Aeron. Sci.*, Yokohama, Japan, 2004, pp. 1–12.

[18] P. A. Bonnefoy, “Scalability of the air transportation system and development of multi-airport systems: A worldwide perspective,” Ph.D. dissertation, MIT, Cambridge, MA, USA, Jun. 2008.

[19] N. Alexandrov, “Transportation network topologies,” NASA Langley Res. Center, Hampton, VA, USA, Tech. Rep., 2004.

[20] T. Kotegawa, D. DeLaurentis, K. Noonan, and J. Post, “Impact of commercial airline network evolution on the US air transportation system,” in *Proc. 9th USA/Europe ATM Res. Develop. Semin.*, Berlin, Germany, 2011.

[21] A. Bigdeli, A. Tizghadam, and A. Leon-Garcia, “Comparison of network criticality, algebraic connectivity, and other graph metrics,” presented at the 1st Annu. Workshop Simplifying Complex Network Practitioners, Venice, Italy, 2009, Paper 4.

[22] A. Jamakovic and S. Uhlig, “On the relationship between the algebraic connectivity and graph’s robustness to node and link failures,” in *Proc. 3rd EuroNGI Netw. Conf.*, May 2007, pp. 96–102.

[23] A. Jamakovic and P. V. Mieghem, “On the robustness of complex networks by using the algebraic connectivity,” in *Proc. Netw. Ad Hoc Sensor Netw., Wireless Netw., Next Gen. Internet*, A. Das, H. K. Pung, F. Bu Sung Lee, and L. Wong Wai Choong, Eds., 2008, pp. 183–194.

[24] R. H. Byrne, J. T. Feddema, and C. T. Abdallah, “Algebraic connectivity and graph robustness,” Sandia Nat. Lab., Albuquerque, NM, USA, Tech. Rep., Jul. 2009.

[25] P. Wei and D. Sun, “Weighted algebraic connectivity: An application to airport transportation network,” in *Proc. 18th IFAC World Congress*, Milan, Italy, Aug. 2011, pp. 13 864–13 869.

[26] A. Ghosh and S. Boyd, “Growing well-connected graphs,” in *Proc. 45th IEEE Conf. Decis. Control*, Dec. 2006, pp. 6605–6611.

[27] J. Sun, S. Boyd, L. Xiao, and P. Diaconis, “The fastest mixing Markov process on a graph and a connection to a maximum variance unfolding problem,” *SIAM Rev.*, vol. 48, no. 4, pp. 681–699, Nov. 2006.

[28] F. Goring, C. Helmberg, and M. Wappler, “Embedded in the shadow of the separator,” *SIAM J. Optim.*, vol. 19, no. 1, pp. 472–501, Feb. 2008.

[29] S. Boyd, “Convex optimization of graph Laplacian eigenvalues,” in *Proc. Int. Congr. Math.*, 2006, vol. 3, pp. 1311–1319.

[30] S. Boyd, P. Diaconis, and L. Xiao, “Fastest mixing Markov chain on a graph,” *SIAM Rev.*, vol. 46, no. 4, pp. 667–689, Dec. 2004.

[31] M. Fiedler, “Algebraic connectivity of graphs,” *Czechoslovak Math. J.*, vol. 23, no. 98, pp. 298–305, 1973.

[32] B. Mohar, “The Laplacian spectrum of graphs,” in *Graph Theory, Combinatorics, and Applications*. New York, NY, USA: Wiley, 1991, pp. 871–898.

[33] M. Siotani, “Some applications of Loewner’s ordering on symmetric matrices,” *Ann. Inst. Stat. Math.*, vol. 19, no. 1, pp. 245–259, Dec. 1967.

[34] J. F. Sturm, “Using SeDuMi 1.02, A Matlab toolbox for optimization over symmetric cones,” *Optim. Methods Softw.*, vol. 11, no. 1–4, pp. 625–653, Jan. 1999.

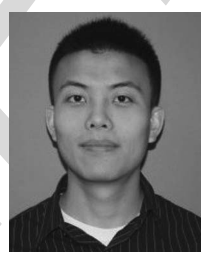
[35] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *The Art of Scientific Computing,* 3rd ed. Cambridge, U.K.: Cambridge Univ. Press, 2007. ch. Section 10.2. Golden Section Search in One Dimension, pp. 492–496.

[36] J. Kleinberg and E. Tardos, *Algorithm Design*. Reading, MA, USA: Addison-Wesley, 2005, ch. 4.7 Clustering, pp. 157–161.

[37] S. Martinez, G. Chatterji, D. Sun, and A. Bayen, “A weighted graph approach for dynamic airspace configuration,” presented at the AIAA Conf. Guidance, Control Dynamics, Hilton Head, SC, USA, Aug. 2007, Paper AIAA 2007-6448.

[38] C. Wu, “Algebraic connectivity of directed graphs,” *Linear Multilinear Algebra*, vol. 53, no. 3, pp. 203–223, Jun. 2005.

[39] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, MA, USA: MIT Press, 2009, ch. 26.2 The Ford-Fulkerson method, pp. 727–730.



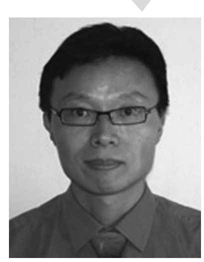
Peng Wei (M’12) received the Bachelor’s degree in automation from Tsinghua University, Beijing, China; the Master’s degree in electrical engineering from Stony Brook University, Stony Brook, NY, USA; and the Ph.D. degree in aerospace engineering from Purdue University, West Lafayette, IN, USA.

He is an Operations Research Consultant with American Airlines, Fort Worth, TX, USA. His research interests include next-generation air transportation system, environmental impact of aviation, and unmanned aerial vehicle integration in the National Airspace System.

Dr. Wei received the Purdue College of Engineering Outstanding Research Award in 2013.

Gregoire Spiers received the Bachelor’s degree in applied mathematics from École Polytechnique, Palaiseau, France, in 2012.

He was a Visiting Student with Purdue University, West Lafayette, IN, USA, in 2011. He is currently a Research Engineer with Amadeus S.A.S., Sophia Antipolis, France.



Dengfeng Sun (M’08) received the Bachelor’s degree in precision instruments and mechanology from Tsinghua University, Beijing, China; the Master’s degree in industrial and systems engineering from The Ohio State University, Columbus, OH, USA; and the Ph.D. degree in civil engineering from University of California, Berkeley, CA, USA.

He is an Assistant Professor with the School of Aeronautics and Astronautics, Purdue University, West Lafayette, IN, USA. His research interests include control and optimization, with an emphasis on applications in air traffic flow management, dynamic airspace configuration, and studies for the next-generation air transportation system.

AUTHOR QUERIES

AUTHOR PLEASE ANSWER ALL QUERIES

AQ1 = Please provide keywords.
AQ2 = Please check if changes made in the affiliation note of author Gregoire Spiers are appropriate, with reference to the given biography. Please also check accuracy of postal code and location.
AQ3 = Please provide page range in Ref. [20].
AQ4 = Please check accuracy of provided location.

END OF ALL QUERIES