Algebraic Connectivity Optimization of the Air Transportation Network

Gregoire Spiers[1], Peng Wei[2], Dengfeng Sun[2]

***Abstract*— In transportation networks the robustness of a network regarding nodes and links failures is a key factor for its design. The goal of this work is to design the most robust network given only the location of each node. A common way to measure the robustness of a network is to evaluate the algebraic connectivity of the graph, which was introduced by Fiedler. Several works solve the maximization of the algebraic connectivity by choosing the weights for the edges in the graph. Other works focus on the best way to add edges in a network in order to optimize the connectivity. In this work we present a practical way to find both the edges and their weights in order to optimize the network robustness. We show that this wider problem which is not currently mentioned in the literature is interesting because the two sub-problems of adding edges and choosing edge weights can not be treated separately. The new combined problem inspired by the air transportation network is formulated and exactly solved in small size network case. For larger size networks, we propose our approximation algorithm and the simulated numerical results are analysed.**

I. Introduction

The goal of this work is to design a robust air transportation network. More precisely, we want to build a network where if there is a failure of an edge or a node, the network remains connected. The network is made of nodes that represent the airports and edges that represent the flight routes which directly link two airports. In some circumstances, a failure in the network can happen. Either a node or a link of the graph may breakdown. For example the failures can be caused by severe weather conditions. Therefore to build a robust or well connected network is a practical problem that has very important economic impact. Although there exist other tools to measure the connectivity of a network, we measure the connectivity by computing the algebraic connectivity, which is usually considered as one of the most reasonable and neat evaluation methods [1], [2].

We consider a graph G with n nodes and m edges. Let $A = (a_{ij})$ be the adjacency matrix of G. The Laplacian matrix $L = (l_{ij})$ of G is defined by:

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

We name the eigenvalues of L: $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. L is a semi-definite positive matrix so for all i, $\lambda_i \geq 0$. We also know that $\lambda_1 = 0$ since $Le = 0$ with $e = (1, \cdots, 1)$.

[1] G. Spiers is with the Department of Applied Mathematics, Ecole Polytechnique, 91120, Palaiseau, France `gregoire.spiers at polytechnique.org`

[2] P. Wei and D. Sun are with the School of Aeronautics and Astronautics, Purdue University, West Lafayette, IN 47907, USA `{weip,dsun} at purdue.edu`

Definition: $\lambda_2(L)$ is the algebraic connectivity of G.

We now recall the three well known properties of the algebraic connectivity that will be used in this work.

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

$$\Omega = \{x \in \mathbb{R}^n \mid \|x\| = 1,\ e^T x = 0\}$$

The Courant Fischer principle states that [3]:

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

Property 2: The algebraic connectivity is a lower bound for both the node connectivity and the edge connectivity of a graph (see [4]).

This property is the main reason why the algebraic connectivity is used to measure the robustness of a graph.

Property 3: The function $w \to \lambda_2(w)$ is concave. This can be proven by seeing that $\lambda_2(w)$ is the pointwise infimum of a family of linear functions of w (see [5]):

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

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

The related work in algebraic connectivity has been studied for a long time and there are some results about it. Concerning the optimization of the algebraic connectivity on a graph, the problems studied in the literature can be divided into two groups:

a) The edge addition problem: The goal is to add (or remove) a given number of edges on a graph in order to get the best algebraic connectivity:

$$\max_{\Delta E} \ \lambda_2(G(E_0 + \Delta E))$$
$$\text{s.t.} \begin{cases} |\Delta E| = k \\ \Delta E \subset P,\ P \cap E_0 = \emptyset \end{cases}$$

The algorithms that have been developed to solve the problem include tabu search [2], greedy algorithms [2], [6], and rounded SDP [6].

b) The variable weights problem: The edges of the graph are fixed and the goal is to choose the weights of the edges in order to maximize the algebraic connectivity:

$$\max_{w \in \mathbb{R}^m} \lambda_2(G)$$
$$\text{s.t.} \begin{cases} \sum_{i=1}^{m} w_i d_i \leq D \\ \forall i,\ w_i \geq 0 \end{cases}$$

where D and $(d_1, \cdots, d_m)$ are the given data of the problem. This is a convex optimization problem and it is often solved by using a SDP formulation [5], [7], [8] or a sub-gradient algorithm [9].

The main contribution of this paper compared with what have been studied is that we show that in order to find the optimal solution, the two problems above can not be separated. Therefore we propose our algorithm to solve both problems at the same time: the edges of the graph are free, as well as their weights.

The rest of this paper is organized as follows. We show in Section II why this problem naturally arise in transport networks and how it can be formulated. In Section III we solve exactly the problem for small networks and highlight the fact that these two problems are not independent. As a result, we write an algorithm that gives better results. To deal with the larger size networks, we present the SDP formulation in Section IV. And we discuss the SDP solution rounding techniques in Section V. Section VI gives the full approximation algorithm for larger size networks. Section VII presents the simulation results. Finally we conclude the paper in Section VIII.

II. Problem formulation

We consider a set of airports a_i with given locations in the plane. The goal is to connect them so that we maximize the algebraic connectivity of the network under several constraints.

There are $m = \frac{n(n-1)}{2}$ edges in the complete symmetric graph. Each of them has a distance d_{ij} and a weight w_{ij} representing the amount of traffic on the link. We consider the following constraints:

- For safety reasons and because the airports have a limit in the traffic they can handle, the edges have a maximum capacity β:
$$\forall (i,j) \in E,\ w_{ij} \leq \beta$$
- The total distance represents the cost of the fuel used which is one of the main cost. So the total distance travelled is limited by:
$$\sum_{ij} w_{ij} d_{ij} \leq D$$
- The edges also need a minimum amount of traffic α:
$$\forall (i,j) \in E, \quad w_{ij} \geq \alpha \quad \text{or} \quad w_{ij} = 0$$
Indeed, opening a new route when there is considerable traffic demand. For example there are no flights from Paris to Indianapolis because the demand is not large enough.
- The airports need a minimum number of passengers which corresponds to the amount of travellers that actually want to go to this particular airport.
$$\forall i \in \{1, \cdots, n\},\ \sum_{j=1}^{n} w_{ij} \geq p_i$$
The constant p_i can be set proportional to the population of the city. p_i may be ignored at first but will be important to get realistic results at the end.

In summary, the complete problem we aim at solving is:

$$\max_{w} \ \lambda_2(L(w)) \quad \text{s.t.} \ \begin{cases} \sum_{ij} w_{ij} d_{ij} \leq D \\ w_{ij} \in \{0, [\alpha, \beta]\} \\ \sum_j w_{ij} \geq p_i \end{cases} \qquad \text{(P)}$$

A. The alternative formulation

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

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

This is useful since we can now express the domain for w as:

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

Then we add a variable k that determines the number of edges in the graph. The final formulation of the problem is:

$$\max_{x,w,k} \ \lambda_2(L(w)) \quad \text{s.t.:} \ \begin{cases} \sum_{i,j} x_{ij} = k \\ x_{ij} \in \{0,1\} \\ \sum_{i,j} w_{ij} d_{ij} \leq D \\ \alpha x_{ij} \leq w_{ij} \leq \beta x_{ij} \\ \sum_j w_{ij} \geq p_i \end{cases} \qquad (2)$$

B. Difficulty

The problem (P) is an extension of the flight routes addition problem in [2], which is proven to be NP-hard, therefore the extended version problem (P) is also NP-hard.

An important remark is that the problem can not really be split into two steps of first deciding if $w = 0$ or not and followed by choosing the appropriate weights. This is due to the fact that there are minimum and maximum constraints on w. However, assuming the two steps are independent, we can try a decoupled approach. The first step is to choose edges for the empty graph which corresponds to the edge addition problem:

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

and then to choose the weights on them:

$$\max_{w} \ \lambda_2(L(w))$$
$$\sum_i w_i d_i \leq D, \quad \alpha y_i \leq w_i \leq \beta y_i, \quad y = x_{opt}$$

We will see in Section III that if we use this approach the results will not be optimal.

C. Relaxation

The relaxation (R) of the problem is obtained by allowing non-integer values for x:

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

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

III. 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 a 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 we can optimize independently each configuration and find the one that leads to the best result.

We consider n nodes chosen randomly. There are $m = \frac{n(n-1)}{2}$ edges and 2^m configurations to test. For each configuration, if Y is the set of the edges that are actually in the graph, we solve the problem:

$$\max_w \ \lambda_2(L(w)) \quad \text{s.t.} \quad \begin{cases} \sum_{ij} w_{ij} d_{ij} \leq D \\ \alpha \leq w_{ij} \leq \beta, \quad \forall (i,j) \in Y \\ w_{ij} = 0, \qquad \forall (i,j) \notin Y \\ \sum_j w_{ij} \geq p_i \end{cases}$$

This can be done by solving the SDP corresponding to the weight optimization problem (see [5] for details)

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

It becomes impossible to exactly solve the problem when n is large. So we assume n to be small in this section.

A. Results

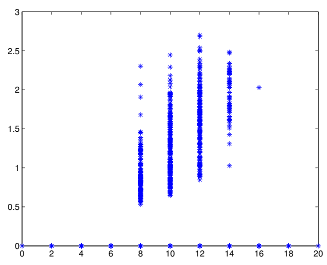

Fig. 1: Results of (k, λ_2) for all the configurations for $n = 5$, $D = 6.5$.

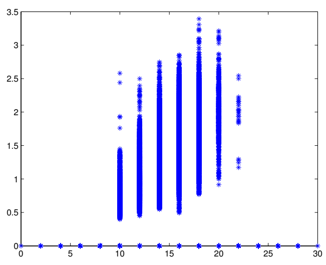

Fig. 2: Results of (k, λ_2) for all the configurations for $n = 6$, $D = 8$.

We compute for each configuration the number of edges k in the graph. We then plot (k, λ_2) and we get the results of Figure 1 and 2 for two different networks.

We notice that the best connectivity is not reached for the maximum number of edges and therefore the choice of the edges and the choice of the weights are not independent.

As it is impossible to exactly solve this NP-hard problem for networks with a reasonable larger size, we are going to design an algorithm that solves it approximately. The main idea is to use the quasi-concave shape of the function $f(k)$.

For the practical problem with a larger size, the first step is to choose a value for k. We are able to solve the relaxed version (R) of the problem where x and w are non-integer variables. We can then round the result to get a feasible solution of the original problem (P).

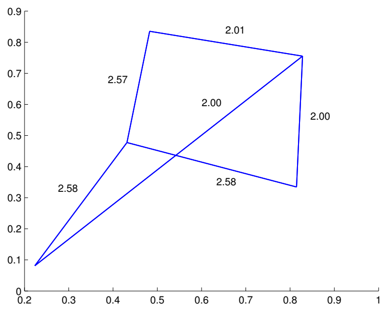

Fig. 3: An optimal network for $n = 5$ with the weights. $p = 0$, $\alpha = 2$ and $D = 6.5$

B. 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. For randomly chosen nodes in a square, we find k_{lim}, which is the maximal number of edges k, is the root of the following equation:

$$a k_{lim}^2 + b k_{lim} + c = \frac{D}{\alpha} \tag{3}$$

where a, b and c are constant parameters.

The quadratic fitting result obtained by Formula (3) is a good approximation of our experiment.

IV. SDP formulation for larger size relaxation problems

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

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

We add a variable μ that allows any $v \in \mathbb{R}^n$:

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

It can be written using Loewner's order [10]:

$$\begin{cases} \lambda_2 = \max_{\lambda,\mu} \lambda \\ \mu e e^T + L - \lambda I \succeq 0 \end{cases} \tag{4}$$

The relaxation of the problem we want to solve is:

$$\max_{x,w,k} \ \lambda_2(L(w)) \quad \text{s.t.} \quad \begin{cases} \sum_i x_i = k \\ x_i \in [0,1] \\ \sum_i w_i d_i \leq D \\ \alpha x_i \leq w_i \leq \beta x_i \\ \sum_j w_{ij} \geq p_i \end{cases}$$

which now becomes with Eqn. (4):

$$\max_{x,w,k,\lambda,\mu} \ \lambda \quad \text{s.t.} \quad \begin{cases} \sum_i x_i = k \\ x_i \in [0,1] \\ \sum_i w_i d_i \leq D \\ \alpha x_i \leq w_i \leq \beta x_i \\ \sum_j w_{ij} \geq p_i \\ \mu e e^T + L - \lambda I \succeq 0 \end{cases} \tag{5}$$

This problem is a SDP since there is a semi-definite constraint and all the other constraints are linear. It can be solved very efficiently by a SDP solver. We use SeDuMi [11] for this paper.

A. *Optimality conditions*

For simplicity, we consider the case in which $p = 0$. The primal SDP is:

$$\max_{x,w,k,\lambda,\mu} \quad \lambda \quad \text{s.t.} \left\{ \begin{array}{l} \sum_i x_i = k \\ x_i \in [0,1] \\ \sum_i w_i d_i \leq D \\ \alpha x_i \leq w_i \leq \beta x_i \\ \mu ee^T + L - \lambda I \succeq 0 \end{array} \right.$$

The KKT optimality conditions are:

$$\left\{ \begin{array}{l} SX = XS = 0 \\ S \succeq 0, \quad X \succeq 0 \\ \langle E_{ij}, X \rangle = d_{ij} \\ L - I + \frac{1}{n} J = S \end{array} \right.$$

- If we have $w_i = w$ for all i, we get:
$$S = (nw - 1)(I - \frac{1}{n} J)$$
if we choose, $w = \frac{1}{n}$, we get $S = 0$ and the conditions are satisfied.
- Reciprocally, if the optimality conditions are satisfied, we have
$$\left(L - (I - \frac{1}{n} J) \right) X = 0$$
X is the matrix of the distances and has rank n so
$$L = I - \frac{1}{n} J$$
and thus w is constant for all i and $w = \frac{1}{n}$.

We finally get the optimality condition:

$$\left\{ \begin{array}{l} \forall (i,j), \ w_i = w_j \\ \sum_i w_i d_i = D \end{array} \right.$$

B. *Upper bound*

With the SDP formulation, we can easily solve the continuous relaxation. When computing the value of the optimal connectivity for different values of k, we get the upper bound in Figure. 4.

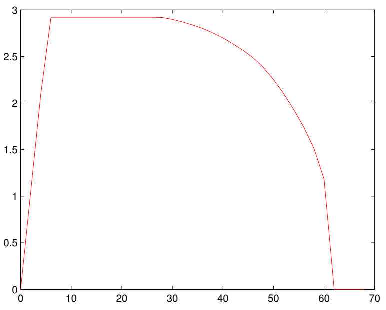

Fig. 4: Upper bound: $\lambda_2 = f(k)$.

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

$$\left\{ \begin{array}{l} \sum_{i=1}^m w_i d_i = D \\ \forall (i,j), \ w_i = w_j \end{array} \right.$$

All the weights are equal and their value is: $\forall i, \ w_i = \frac{D}{\sum_j d_j} = \Omega$. If we have $w = \beta x$ all the elements of x are equal and their value is: $\forall i, \ x_i = \frac{k}{n^2 - n}$, which leads to

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

By doing the same computation we prove the optimal value is also reached with

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

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

In addition, when we round the solution, we may get no feasible value. For example, if $k = k_{min}$, there is often no solution since we can have:

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

if $\frac{\Omega}{\beta} << 1$ which is often the case. And as we need at least $n - 1$ edges to connect a n node graph there is no positive solution. The upper bound is not a very good bound for small values of k.

V. The rounding techniques for SDP solution

A. *Description of the methods*

In this section, we suppose we have found the value of the relaxed optimal solution s_0. We want to select k edges from s_0 which means $x_i = 1$ for k values and $x_i = 0$ for the others. There are several ways to do so. We present here the methods that have been studied and implemented.

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

2) Random fast: We randomly choose the rounding. For each $i \in \{1, \cdots, m\}$, $x_i = 1$ with probability $s_0(x_i)$ and $x_i = 0$ otherwise. Then we affect the weights 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 $s_0(x_i)$ and therefore

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

and, for the same reason:

$$\overline{\sum_i w_i d_i} = \sum_i \frac{s_0(w_i)}{s_0(x_i)} \overline{x_i} d_i = \sum_i s_0(w_i) d_i \leq D$$

Thus on average the solution satisfies the constraints. We keep at the end the best solution that satisfies all the constraints.

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

4) Step by step: We select the biggest element $s_0(x_i) < 1$ and affect its minimum value to 1 in the SDP formulation. Then we solve again the SDP and repeat k times the two steps.

5) Log step by step: This is the same idea as the "step by step" except that, at each step, we choose the best half of the remaining elements. So their are only $\log(k)$ SDP that has to be solved.

B. Numerical results

We set up the simulation with 20 nodes generated randomly in a square. The results are presented in Figure 5 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 might fail to find a solution. In that case we remove the corresponding values from the figure.

We see that we can do better than 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.

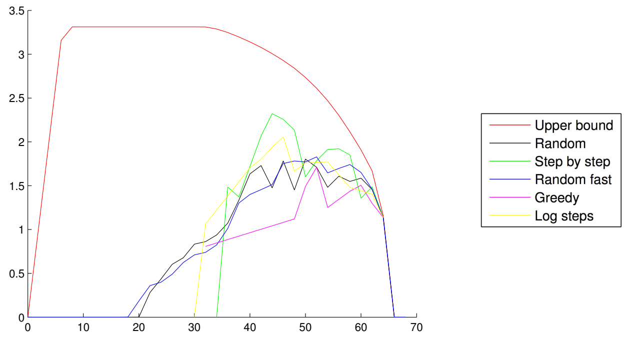

Fig. 5: λ_2 as a function of k. The results are represented for a 20 nodes graph and several rounding methods.

C. 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*. And the method that gives the best compromise between speed and value is the *log step by step*.

VI. Full algorithm

A. Golden section search

We are now able to find, for a given k, a well connected network with k edges. But instead of testing all the possible values of k, we can speed up the search by considering that the algebraic connectivity is a concave function of k.

This approximation leads to better results with rounding methods that have a good regularity. For large networks we can use rounding methods with lower regularity but instead of computing the value for a given k, use a local average value by computing three values :

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

As we can only compute the value of the connectivity for integer values of k, we cannot use continuous optimization principles. Thus we adopt the golden section search [12]. It consists in creating a decreasing set of intervals containing the optimal value :

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

and $k_{opt} \in [a_i, b_i]$. We need two test values $c_i < d_i$ in $[a_i, b_i]$. The rule used to update the interval is the following :

- $f(c_i) < f(d_i) \Rightarrow [a_{i+1}, b_{i+1}] = [c_i, b_i]$
- $f(c_i) > f(d_i) \Rightarrow [a_{i+1}, b_{i+1}] = [a_i, d_i]$

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

B. Algorithm

We are now able to sum up all the steps of the algorithm. The SDP are solved using the SeDuMi solver [11]. This algorithm that leads to the approximation of the optimum is listed in Algorithm 1.

Algorithm 1 Full algorithm

```
1: Initialize $a$, $b$ and $d$
2: while $b - a > 2$ do
3:     Choose c in by golden section search
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:        $b \leftarrow d$
14:        $d \leftarrow c$
15:    end if
16: end while
17: return $\lambda_2$
```

We can analyse the complexity of this algorithm which depends on several parameters of the problem. The algorithm requires to solve $k+1$ SDP 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 $k_{lim}\phi^{-U} = 1$ since at each step the length of the interval is divided by ϕ. We have:

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

We also need to consider the complexity T to solve the SDP. T is polynomial in the size of the entry which is equivalent to n^2. So T is a polynomial function of n.

Therefore complexity C of the whole algorithm can be approximated by $C = O(k_{opt}UT)$.

VII. Numerical results

A. The optimal network

In order to test the full algorithm, we generate a set of random nodes in a square. The parameters p_i's are also chosen randomly. An example of the optimal network is represented in Figure 6 for 30 nodes.

The edges that have a weight greater than the minimum weight value α are represented with a thicker line. In the example of Figure 6 there are 10 edges with a larger weight value than α.

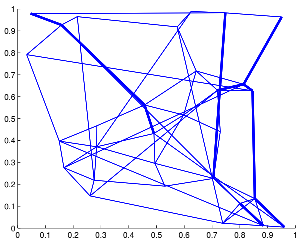

Fig. 6: The optimal result for a 30 nodes graph.

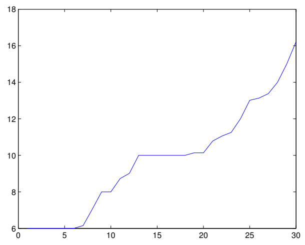

Fig. 7: The sorted degrees of the nodes for the optimal network.

After obtaining the optimal network, we can compute the degree d_i for each node by $d_i = \sum_j w_{ij}$. The law of the network is the curve representing the sorted degree of the nodes. For example, for the air transportation network, the law of the network is known to be a power law [13], [14]. However, the network which is the solution of the problem has a much more progressive law which is represented in Figure 7 since $n = 30$ is not a number large enough to make a precise comparison.

B. Efficiency

We now want to evaluate the efficiency of the result by comparing the optimal algebraic connectivity to the upper bound. For a given set of nodes, we solve the problem for different values of the parameter D and we compute the percentage of the solution to the bound:

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

The result, illustrated in Figure 8, shows that for small values of D the best result found is very far from the upper bound. However, when increasing D, the objective value of the problem (P) quickly increases to reach the value of its relaxed version (R).

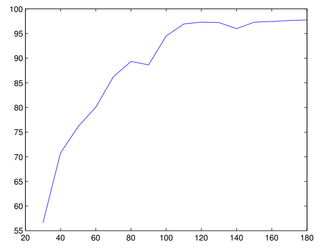

Fig. 8: Values of r for different values of D.

VIII. Conclusion

In this work we have presented and studied a new problem concerning the optimization of the connectivity of a network. This problem consists of finding both the edges of the graph and their weights under several practical constraints. We first exactly solve the problem on small size networks and show that the problem can not be separated into two already studied independent problems. Then we presented an algorithm to solve approximately the problem for larger size instances. Finally we perform analysis on the proposed algorithm and the numerical results.

The problem studied in this work is able to model real problems such as transportation networks robustness. It also provides an option to improve current ways of designing networks.

References

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

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

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

[4] N. M. M. de Abreu, "Old and new results on algebraic connectivity of graphs," *Linear Algebra and its applications*, vol. 423, pp. 53–73, 2007.

[5] 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 Review*, vol. 48(4), pp. 681–699, November 2006.

[6] A. Ghosh and S. Boyd, "Growing well-connected graphs," in *the 45th IEEE Conference on Decision and Control*, December 2006, p. 6605C6611.

[7] F. Goring, C. Helmberg, and M. Wappler, "Embedded in the shadow of the separator," *SIAM Journal On Optimization*, vol. 19(1), pp. 472–501, 2008.

[8] S. Boyd, "Convex optimization of graph laplacian eigenvalues," in *Proceedings International Congress of Mathematicians*, vol. 3, 2006, pp. 1311–1319.

[9] S. Boyd, P. Diaconis, and L. Xiao, "Fastest mixing markov chain on a graph," *SIAM Review*, vol. 46(4), pp. 667–689, December 2004.

[10] M. Siotani, "Some applications of loewner's ordering on symmetric matrices," *Annals of the Institute of Statistical Mathematics*, vol. 19(1), pp. 245–259, 1967.

[11] I. Polik, T. Terlaky, and Y. Zinchenko, *SeDuMi: a package for conic optimization*. McMaster University, Advanced Optimization Laboratory, 2007.

[12] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *The Art of Scientific Computing*, 3rd ed. Cambridge University Press, 2007.

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

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