Algebraic Connectivity Optimization of Large Scale and Directed Air Transportation Network

Gregoire Spiers * **Peng Wei** † **Dengfeng Sun** †

* *Department of Applied Mathematics, Ecole Polytechnique, 91120, Palaiseau, France (e-mail: gregoire.spiers@polytechnique.org)*
† *School of Aeronautics and Astronautics, Purdue University, West Lafayette, IN, USA, 47907 (e-mail: {weip, dsun}@purdue.edu)*

Abstract:
In transportation networks the robustness of a network regarding nodes and links failures is a key factor for its design. A common way to measure the robustness of a network is to evaluate its algebraic connectivity. In our previous work we formulated a new air transportation network model and showed the algebraic connectivity optimization problem is important because the two sub-problems of adding edges and choosing edge weights can not be treated separately. We presented a method to find both the edges and the corresponding weights in order to optimize the small scale network algebraic connectivity, which is a measure for robustness. In this paper we propose the cluster decomposition method to solve the larger size network problem. Moreover, the algebraic connectivity optimization for directed air transportation network is discussed. Simulations are performed for both large scale problem and directed network problem.

Keywords: Air transportation network modeling, algebraic connectivity, large scale network, directed network

1. INTRODUCTION

The air transportation network is made of nodes that represent the airports and edges that represent the flight routes which directly link two airports (Vargo et al., 2010; Wei and Sun, 2011). In air transportation network both node or link failures can happen due to airline budget cut, weather hazard, economic policy, etc. Under these planned or unpredictable node and link failures, how to build a robust or well connected network is a practical problem that has significant economic impact. We measure the robustness of air transportation network by computing the algebraic connectivity, which is usually considered as one of the most reasonable and neat evaluation methods (Wei and Sun, 2011; Jamakovic and Uhlig, 2007; Jamakovic and Mieghem, 2008).

We consider an air transportation network, which has its graph representation 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 $e = (1, \cdots, 1)$.

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 (Mohar, 1991) states that:

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

Property 2 The algebraic connectivity is a lower bound for both the node connectivity and the edge connectivity of a graph (see de Abreu (2007)).

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 Sun et al. (2006)):

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

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

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.

1.1 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 (Wei and Sun, 2011), greedy algorithms (Wei and Sun, 2011; Ghosh and Boyd, 2006), and rounded semidefinite programming (SDP) (Ghosh and Boyd, 2006).

1.2 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 semidefinite programming (SDP) formulation (Sun et al., 2006; Goring et al., 2008; Boyd, 2006) or a sub-gradient algorithm (Boyd et al., 2004).

In Spiers et al. (2012) we showed that in order to find the optimal solution, the two problems above can not be separated. For small scale problems, we proposed our algorithm to solve both problems at the same time: the edges of the graph are free, as well as their weights.

The major contribution of this paper is to present the cluster decomposition method for the same problem from Spiers et al. (2012) in large scale networks. In addition, the algebraic connectivity optimization for directed air transportation network is also discussed. Numerical results are shown to study the performance of our methods in large scale scenario and directed network scenario.

The rest of this paper is structed as follows. We revisit our problem formulation in Section 2. In Section 3 we solve the problem for large scale networks, the computational efficiency is analysed and the numerical results are provided. We present the algebraic connectivity optimization for directed air transportation network in Section 4. The algorithm performance, solution result and failure case examples in directed network scenario are also discussed in Section 4. Section 5 concludes the paper.

2. 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:
$$\forall (i,j) \in E, \quad w_{ij} \geq \alpha \ \text{ or } \ w_{ij} = 0$$
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:

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

2.1 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 exist or not 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 :

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

The problem now becomes:

$$\max_{x,w}\ \lambda_2(L(w)) \quad \text{st} : \begin{cases} x_{ij} \in \{0,1\} \\ \sum_{ij} w_{ij} d_{ij} \leq D \\ \alpha x_{ij} \leq w_{ij} \leq \beta x_{ij} \\ \sum_j w_{ij} \geq p_i \end{cases}$$

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

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

These additional variables will be necessary to approach the solution of the problem by progressively eliminating bad assignments.

2.2 Difficulty

The problem (P) is an extension of the flight routes addition problem (Wei and Sun, 2011) in which the weights were fixed and the goal was to choose which edges had to be added in order to optimize the connectivity. This

problem is proven to be NP-Hard. The extended version we are dealing with is therefore also a difficult problem.

An important remark is that the problem can not really be splitted into two steps where the first one consists in choosing if $w = 0$ or not and the second one in choosing the value of the weights. This is due to the fact that their is a minimum and a maximum constraint on w (node and distance).

When we forget the minimum node condition we can however try a decomposition. The first step is to choose edges for the empty graph which corresponds to the edge addition problem:

$$\max_x \ \lambda_2(L(x))$$

$$\text{subject to : } \sum_i x_i = k, \quad x_i \in \{0,1\}, \quad \sum_i x_i d_i \leq \frac{D}{\alpha}$$

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 saw in Spiers et al. (2012) that if we use this approximation in two steps the results can be really bad in comparison with well adapted methods.

2.3 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. Several kinds of rounding techniques were implemented and compared in Spiers et al. (2012) too.

3. LARGE SCALE AIR TRANSPORTATION NETWORK

3.1 Necessity

We now want to apply the previous work to large networks. The most time consuming operation in the process is the resolution of the SDP. Figure 1 represents the computational time required to solve the SDP of the problem for n nodes. We see that the time to solve it increases very rapidly. In fact for n nodes there are $n(n-1)+2 \sim n^2$ variables in the SDP. As we need to solve several SDPs in order to solve the problem, it becomes almost impossible for $n \geq 35$.

However we would like 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 even when considering only the USA.

3.2 Cluster decomposition

As the main constraint is a limit on the maximum distance traveled, the optimal networks tend to contain short links. Thus, the idea in this section is to divide the airports

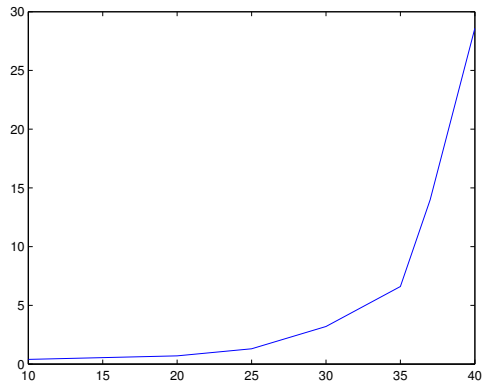


Fig. 1. Time (in seconds) to solve the SDP formulation of the problem for a given number of nodes n.

into $c \in \mathbb{N}$ clusters based on the distance between the nodes. These clusters can be solved independently with the method used in Spiers et al. (2012) and be connected afterwards.

To connect the cluster, we choose k major nodes in each cluster that will be connected each other. Then we simply have to solve the problem (P) for these $c \times k$ nodes, except that links between two nodes from the same cluster are not allowed, so the graph we are working on is not complete.

At the end, we need to solve $c+1$ problems of type (P) to get the final result. Figure 2 shows the idea of the decomposition into several clusters and the selection of major nodes.

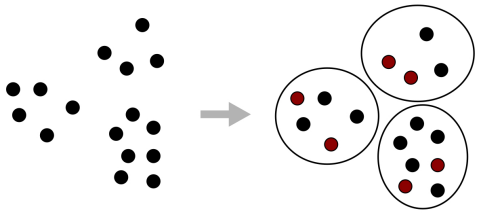


Fig. 2. A set of 16 nodes separated in 3 clusters with 2 major nodes in each (in red).

There are several parameters whose values have to be chosen to apply this idea. First we have to choose the number of clusters and how many major nodes are used in each cluster to connect to other clusters. We also need to choose which nodes are kept as major nodes among each cluster. Naturally we decide here to take the biggest nodes which are the nodes with the largest value of p_i.

In addition, to solve the problem for each small problem $1 \leq i \leq c+1$ we have to choose the value of the maximum distance D_i. A natural option is to choose D_i proportional to the sum s_i of all the distances of the edges in the cluster i and such that $\sum_{i=1}^{c+1} D_i = D$:

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

$$D_i = \frac{s_i}{\sum_{j=1}^{c+1} s_j} D.$$

The separation of the nodes into several clusters is made by k-means algorithm (Kleinberg and Tardos, 2005). This algorithm has the advantages of being fast, easy to implement and generally gives good results.

To sum up the method described above, we write the full cluster decomposition algorithm as:

3.3 Evaluation of the efficiency

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

Algorithm 1 Large scale algorithm

1: Initialize c, D_i
2: k-means algorithm gives c clusters
3: **for** $p = 1$ to c **do**
4: Solve the cluster problem (with Algorithm 1 in Spiers et al. (2012))
5: **end for**
6: Solve the major node problem (Algorithm 1 in Spiers et al. (2012))
7: Build the resulting network
8: **return** $\lambda_2(L)$

We consider c clusters. Each cluster have n nodes and k of its nodes are used to connect to other clusters. Let G be the matrix of the graph and F the vector defined by the expression below. If, for instance $c = 3$, the matrix G can be put into the following form :

$$G = \left(\begin{array}{ccc|ccc|ccc} & & & E & 0 & 0 & E & 0 & 0 \\ & A_1 & & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline E & 0 & 0 & & & & E & 0 & 0 \\ 0 & 0 & 0 & & A_2 & & 0 & 0 & 0 \\ 0 & 0 & 0 & & & & 0 & 0 & 0 \\ \hline E & 0 & 0 & E & 0 & 0 & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & & A_3 & \\ 0 & 0 & 0 & 0 & 0 & 0 & & & \end{array}\right) \quad F = \left(\begin{array}{c} \alpha \\ \beta \\ \beta \\ \hline 0 \\ 0 \\ 0 \\ \hline -\alpha \\ -\beta \\ -\beta \end{array}\right)$$

with the following notation :

- α and β are constants that will be computed in the next paragraph,
- A_1, A_2 and A_3 represent the matrices of the 3 clusters,
- E is a $k \times k$ matrix with all elements equal to 1.

The Fiedler vector The Fiedler vector is the vector solution of the minimization problem :

$$\min_{x \in \mathbb{R}^n} \{x^T L x \mid \|x\| = 1,\ xe = 0\}$$

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 Martinez et al. (2007)).

Here, the optimal cut will naturally be 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 on numerical tests and therefore seems to be a very good approximation of the real Fiedler vector.

Computing the connectivity We consider that the Fiedler vector has the form of F and we also consider that matrices are full: all non diagonal elements are equal to 1. We matrix products give:

$$\lambda_2 = F^T L F$$

$$\boxed{\lambda_2 = 2k\alpha X + 2(n-k)\beta Y}$$

with

$$X = \alpha(n - 1 + k(c-1)) - (k-1)\alpha - (n-k)\beta + k\alpha$$
$$Y = \beta(n-1) - k\alpha - (n-k-1)\beta$$

We also know that $\|F\| = 1$, thus :

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

We substitute α with this expression and β is given by the equation :

$$\boxed{\frac{d\lambda_2}{d\beta} = 0} \tag{3}$$

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

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

In practice, the matrices have a density of ρ instead of being full, we simply get instead:

$$\lambda_2 = \rho f(n, k, c).$$

Results with Maple By solving equation (3), we get the value of α, β and then λ_2. The following figures have been obtained with Maple. Among the three parameters k, c and n we fix two of them and let the third one vary to see its influence on the connectivity.

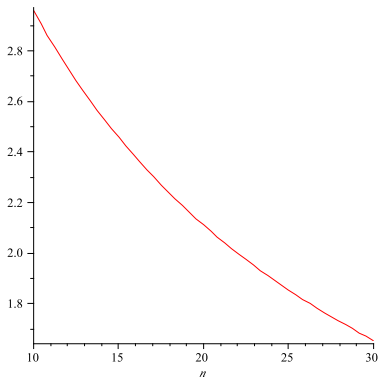


Fig. 3. k and c are fixed. $\lambda_2 = f(n)$.

Figures 3, 4, 5 provide us a more clear idea on how to choose the value of each parameter. For example, the connectivity is almost linear regarding k but has a concave shape when represented as a function of the number of clusters c.

We are looking for a trade-off: if c is too large, kc will be too large to be solved. On the contrary, if c is too small, each of the cluster will have too many nodes to be solved.

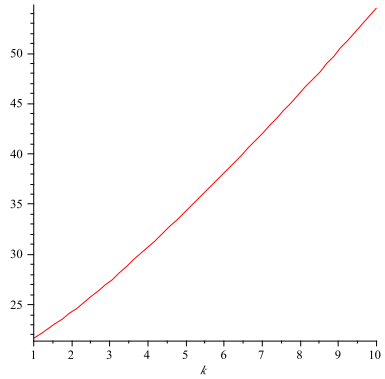


Fig. 4. c and n are fixed. $\lambda_2 = f(k)$.

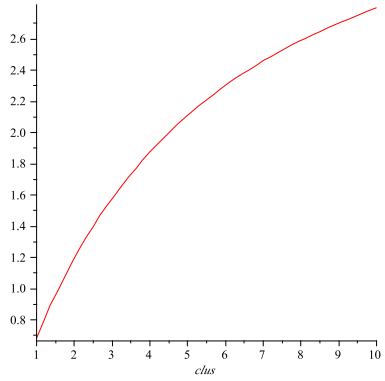


Fig. 5. k and n are fixed. $\lambda_2 = f(c)$.

3.4 Numerical results

The data used are the 100 largest cities in the USA. The values of p_i proportional to the population. Figure 6 represents the 200 biggest cities without any link.

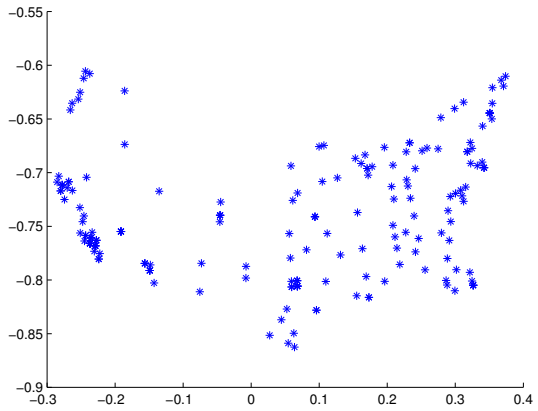


Fig. 6. The 200 biggest cities of the USA.

The optimal network found is represented on Figure 7. The blue lines represent the edges inside each cluster and the red lines represent the edges that connect nodes from different clusters.

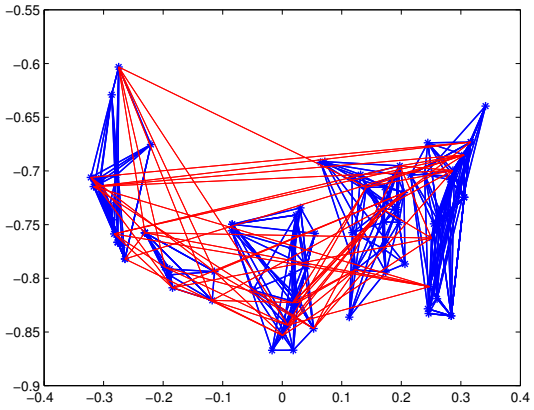


Fig. 7. Result for the 100 biggest cities of the USA.

The connectivity we have reached is:

$$\boxed{\lambda_2 = 2.6}$$

We plot the sorted degrees of the nodes in the network in Figure 8. The curve is closer to cubic law rather than exponential law as in the actual air transportation network (Guimera and Amaral, 2004).

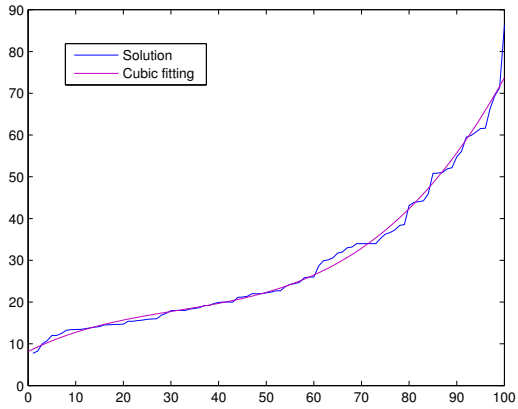


Fig. 8. The sorted degrees of the optimal solution and its corresponding cubic fitting for the USA air transportation network.

4. DIRECTED AIR TRANSPORTATION NETWORK

4.1 From undirected graph to directed graph

In this section, we would like to improve the results by using directed graphs. In order to be consistent with the undirected case, the graphs need to be balanced: the number of aircrafts that come in is the same as the number of aircrafts that leave the airport:

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

It is clear that the set of undirected graphs is included in the set of directed balanced graphs. As a result, we know

we can get results at least as good as our previous work (Spiers et al., 2012).

Definition

According to Wu (2005), if $\Omega = \{x \in \mathbb{R}^n, xe = 0, \|x\| = 1\}$ we can extend the definition of the algebraic connectivity for directed balanced graphs with:

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

Property

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 G its incidence matrix, we can create

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

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 we also know that undirected graphs is a subset of directed balanced graphs, the bound are equals in both cases. This will allow us to evaluate easily the improvement brought by directed graphs.

4.2 Results

We use the same method as in Spiers et al. (2012). The results are impressively better with directed graphs as we can see on Figure 9. The best value for the directed balanced case almost reach the upper bound. We also notice 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).

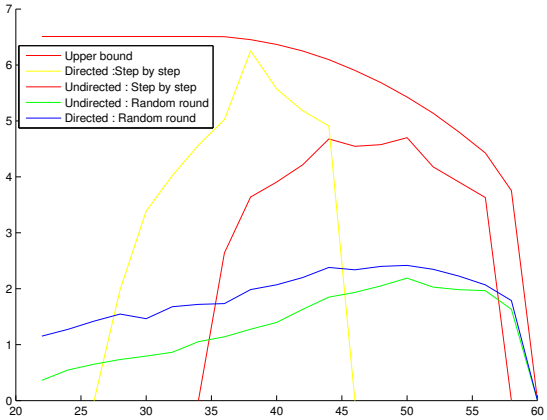


Fig. 9. $\lambda_2 = f(k)$ for the same graph with different approaches. y-axis is the value of λ_2 and x-axis is the number k of edges to be added.

We can see on Figure 10 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.

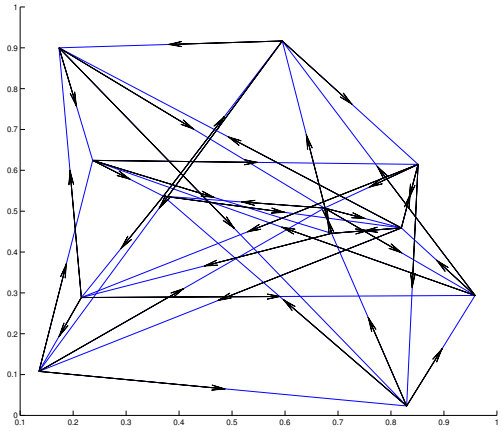


Fig. 10. Optimal directed Graph.

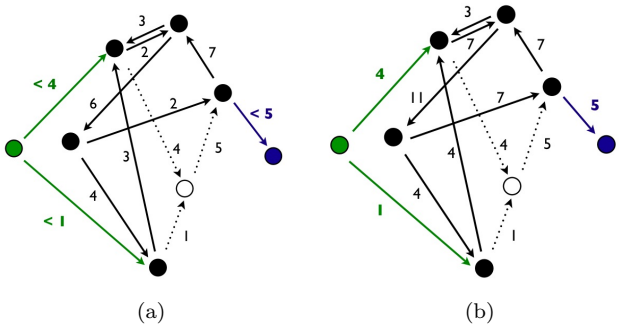
There is however an important drawback. Indeed, we now need two times as many variables. Thus the problem takes a much longer time to be solved and is only applicable on smaller networks for now.

4.3 Failure case

If an edge or a node is removed, the graph is not balanced anymore. This can cause important problems in reality, so we need to change the remaining weights in the graph to face 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 we consider the problem of the maximization of the flow from the source to the sink. This problem can be solved by using Edmond Karp algorithm (Cormen et al., 2009) which has a efficient compexity: $O(nm^2)$. This algorithm maintains the balance of the flow at each node.



(a) (b)

Fig. 11. (a) Example of a graph with a node failure; (b) Example of a graph after maximization of the flow.

At the end the graph we are looking for is the graph solution of the flow problem when we remove the source and the sink. Figures 11a and 11b show an example of a graph in which a node has been removed, before and after

maximization of the flow. The graph in Figure 11b is now balanced when we remove the source (green node) and the sink (blue node).

5. CONCLUSION

In this work we revisit our problem formulation concerning the optimization of the connectivity of a network. This problem consists in finding both the edges of the graph and their weights under physical distance constraint. With the exact solution developed for small scale network, we propose the new cluster decomposition method to solve the large scale network problem and successfully find a reasonable solution for the 100 largest cities in the USA. Finally, we extend our study on directed graphs and emphasize the fact that the algorithm's performance is improved in directed network scenario.

The problem studied is able to model real problems such as transportation networks robustness. It may therefore have practical applications and can lead to improvements with current way of designing networks. Many improvements of this work can be considered, in particular for the large scale problem in which the great complexity of the problem let us think that other methods might be able to outperform our results.

REFERENCES

Boyd, S. (2006). Convex optimization of graph laplacian eigenvalues. In *Proceedings International Congress of Mathematicians*, volume 3, 1311–1319.

Boyd, S., Diaconis, P., and Xiao, L. (2004). Fastest mixing markov chain on a graph. *SIAM Review*, 46(4), 667–689.

Cormen, T.H., Leiserson, C.E., Rivest, R.L., and Stein, C. (2009). *Introduction to Algorithms*, chapter 26.2 The Ford-Fulkerson method, 727–730. The MIT Press.

de Abreu, N.M.M. (2007). Old and new results on algebraic connectivity of graphs. *Linear Algebra and its applications*, 423, 53–73.

Ghosh, A. and Boyd, S. (2006). Growing well-connected graphs. In *the 45th IEEE Conference on Decision and Control*, 6605–6611.

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

Guimera, R. and Amaral, L. (2004). Modeling the world-wide airport network. *European Physical Journal B*, 381–385.

Jamakovic, A. and Mieghem, P.V. (2008). On the robustness of complex networks by using the algebraic connectivity. In A.D. et al (ed.), *Networking 2008 Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet*, 183–194.

Jamakovic, A. and Uhlig, S. (2007). On the relationship between the algebraic connectivity and graph's robustness to node and link failures. In *3rd EuroNGI Conference on Next Generation Internet Networks*.

Kleinberg, J. and Tardos, E. (2005). *Algorithm Design*, chapter 4.7 Clustering, 157–161. Addison-Wesley.

Martinez, S., Chatterji, G., Sun, D., and Bayen, A. (2007). A weighted graph approach for dynamic airspace configuration. In *AIAA Conference on Guidance, Control and Dynamics*. Hilton Head, SC.

Mohar, B. (1991). The laplacian spectrum of graphs. *Graph Theory, Combinatorics, and Applications*, 2, 871–898.

Spiers, G., Wei, P., and Sun, D. (2012). Algebraic connectivity optimization of the air transportation network. In *American Control Conference*. Montreal, Canada.

Sun, J., Boyd, S., Xiao, L., and Diaconis, P. (2006). The fastest mixing markov process on a graph and a connection to a maximum variance unfolding problem. *SIAM Review*, 48(4), 681–699.

Vargo, E., Kincaid, R., and Alexandrov, N. (2010). Towards optimal transport networks. *Systemics, Cybernetics and Informatics*, 8(4), 59–64.

Wei, P. and Sun, D. (2011). Weighted algebraic connectivity: An application to airport transportation network. In *the 18th IFAC World Congress*. Milan, Italy.

Wu, C. (2005). Algebraic connectivity of directed graphs. *Linear and Multilinear Algebra*, 53, 203223.