Preprints of the 18th IFAC World Congress
Milano (Italy) August 28 - September 2, 2011

Weighted Algebraic Connectivity: An Application to Airport Transportation Network

Peng Wei * **Dengfeng Sun** *

* *Purdue University, West Lafayette, IN 47907 USA*
(e-mail: {weip, dsun}@purdue.edu)

Abstract: The airport transportation network structure and its performance are key factors to benefit the local area or national transportation capability and economic development. This paper presents a method based on the weighted algebraic connectivity to optimize and analyze the network. The indispensability of the weighted adjacent matrix and weighted Laplacian matrix in airport transportation network performance study is emphasized in this paper. According to practical operational viability, we formulate the weighted algebraic connectivity application as the flight routes addition/deletion problems. The greedy perturbation heuristic by Ghosh and Boyd is analyzed and extended into the *Modified Greedy Perturbation Algorithm* (MGP) to provide a local optimum in a fast iterative manner. The unweighted tabu search applying algebraic connectivity to airport transportation network by Vargo and Kincaid is re-designed as the *Weighted Tabu Search* (WTS) to fit the new weighted flight routes addition/deletion problems, potentially offering a global optimum with much longer running time. The simulation results indicate the trade-off between the Modified Greedy Perturbation and the Weighted Tabu Search, with which we can decide the appropriate algorithm to be used for enhancing the performance of an airport transportation network.

1. INTRODUCTION

In this paper we consider robustness as the major performance metric of airport networks. The methods developed are expected to be implemented as the performance measurement and the guidance to enhance the robustness on both local small scale and nationwide large scale airport networks. The authors present a metric to evaluate the robustness of the airport network by compute the algebraic connectivity. How to maintain and improve the network robustness is also investigated.

1.1 An Airport Transportation Network

An airport transportation network consists of distinct airports (cities) and non-stop flight routes between airport pairs (Guimera[2004]). Usually a graph $G(V, E)$ is used to describe an airport transportation network (Kincaid[2008], Vargo[2010]), where the node set V represents all the n airports and the edge set E represents all the m non-stop flight routes between airports. With the assumption that if a non-stop flight route from airport a to airport b exists, the non-stop return route from airport b to airport a also exists, $G(V, E)$ is constructed as an undirected simple graph for ease of analysis. The airports are indexed as $\{v_i | i = 1, 2, \ldots, n\}$ and the non-stop flight routes are named as $\{e_{ij} | \text{if there is a non-stop route between airports } i \text{ and } j\}$.

As a simple graph, it is required that no more than one edge exists between any two vertices. However, it is common that the multiple flight routes which are operated by different airlines or even by one airline company exist between two airports. As a result, we introduce non-negative integral edge weights to record multiple fight routes information while maintaining G as a simple graph, i.e., the edge weight w_{ij} shows that there are w_{ij} distinct non-stop flight routes between airports i and j, in this case all w_{ij}'s should be integers.

Moreover, the non-negative integral weighted graph description of the airport transportation network is demanded by the robustness and synchronization evaluations of the airport transportation network, which are demonstrated in following two subsections.

1.2 Connectivity and $\lambda_2(G)$

Connectivity is the metric which is used to measure how well a graph is connected. Usually the more connected is a graph, the more robust it is. With the expected dramatic growth of air traffic in near future (NEXTGEN[2007]), there will be more and more flight delays and cancellations. Thus a robust well-connected airport transportation network design scheme is necessary. Following the scheme, the designer not only can construct a new network, but also can maintain or modify the structure of an existed network and build the backup strategies for the future development of the airport transportation network.

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

In order to show the limitation of vertex (edge) connectivity evaluation, two different topologies are shown in Fig. 1, where Fig. 1a is a N-node line topology and Fig. 1b is a N-node star topology. Obviously, the vertex connectivity for both topology formations is 1 and so is the edge connectivity. However, the star topology should be more robust than the line topology. Since either vertex connectivity or edge connectivity can not observe the difference between these two topologies, we need to introduce a new metric.

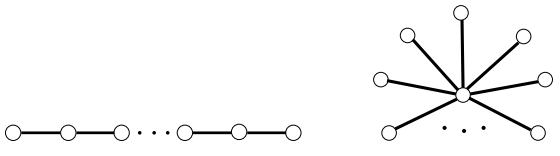


Fig. 1. Two connected N-node topologies.

The second smallest Laplacian eigenvalue λ_2 of a graph is related to many graph invariants. Fiedler calls the number $\lambda_2(G)$ the *algebraic connectivity* (Fiedler [1973]) of graph G.

According to the definition, when $N = 4$, the algebraic connectivities of Fig. 1a and Fig. 1b are 0.588 and 1 respectively, which show that the star topology is more robust than the line topology.

The algebraic connectivity owns higher resolution on how well a graph or network is connected and can be considered as a measurement of the robustness in complex network models which is proved in Jamakovic and Uhlig [2007]. So λ_2 is adopted in this paper to evaluate the robustness of the airport transportation network.

Most of the research on airport transportation network before this work study unweighted simple graphs. Nevertheless, only using 0/1 in graph adjacent matrix is far from enough to describe the status of the flight routes. A weighted simple graph contains more information about the flight route statuses, i.e., a "3/2/1/0" four-level weighting can be adopted to describe a route under different level weather conditions such as "normal/warning/severe/extremely severe." It is important to provide a multi-level flight route status to all the aircraft which plan to take this route.

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

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

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

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

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

1.3 Synchronization and $\lambda_2(G)$

Besides the connectivity, $\lambda_2(G)$ is also relevant to the graph's synchronization (Kincaid[2008]), which can be interpreted as the network behaving in a regular and predictable manner while maintaining networking stability (Barahona[2002]).

1.4 Introduction Summary

In summary, in order to maintain the graphic description of the airport transportation network as a simple graph and obtain better accuracy and higher resolution of the network as well as to provide more information of the route status, it is necessary to introduce the weighted Laplacian matrix. Huang and Dang [2009] first introduced the weighted air transportation network and they studied the correlations of betweenness-degree, degree-degree and clustering-degree of the network. In this paper, we build a different weighted network and focus on the algebraic connectivity application in airport transportation network.

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

Mohar [1991] shows that λ_2 of the weighted Laplacian matrix is also the performance metric for the weighted graph. Therefore, if we want to improve the connectivity, robustness and synchronization of the airport transportation network, we only need to increase λ_2 of its corresponding graph. Therefore the aim of this paper is to maximize the performances of the airport transportation network with the constraints of a fixed number of non-stop flight routes addition or deletion inside a given candidate routes set.

The rest of the paper is organized as follows. In Section 2 we formulate the problem and show the weighted problem is NP-hard. In Section 3 the heuristic algorithm from Ghosh and Boyd for unweighted graph is analyzed and extended to solve the weighted problem. The unweighted tabu search algorithm by Vargo and Kincaid is re-designed into weighted scenario in Section 4. In Section 5 we evaluate the performance of our algorithms via simulations. Section 6 concludes this paper.

2. PROBLEM FORMULATION AND NP-HARDNESS

In application there are few chances to construct a totally new airport transportation network either in a local area or for a whole country. Instead, to maintain or to improve the performance of an existing network, restricted by adding or deleting a fixed number of routes due to airline budgets, weather conditions, economic policy, etc., is the major motivation of this work.

2.1 The flight routes addition/deletion problem

Given the graphic description $G(V, E_0)$ of an existed integral weighted airport transportation network, where the node set V is the collection of all the airports in this network and the edge set E_0 contains the existed routes between the airport pairs. The size of set V is n and the

size of E_0 is m. The objective of this paper is to maximize λ_2 with a fixed number k of edge additions or deletions based on E_0 while the edges to be added or deleted are given in a pre-determined set P or Q. All the weights of the edges in sets E_0, P, Q are non-negative integral and bounded by W. We denote the routes to be added or deleted as a set of ΔE. Thus the *flight routes addition problem* is:

$$\begin{aligned} &\max \lambda_2(G(E_0 + \Delta E)) \\ \text{s.t. } &|\Delta E| = k, \\ &\Delta E \subseteq P, P \cap E_0 = \emptyset, \\ &w_{ij} \in \mathbb{I}, w_{ij} < W. \end{aligned} \tag{3}$$

The *flight routes deletion problem* is:

$$\begin{aligned} &\max \lambda_2(G(E_o - \Delta E)) \\ \text{s.t. } &|\Delta E| = k, \\ &\Delta E \subseteq Q, Q \subseteq E_0, \\ &w_{ij} \in \mathbb{I}, w_{ij} < W. \end{aligned} \tag{4}$$

For simplicity of demonstration, we only focus on the flight routes addition problem. The algorithms for solving flight routes deletion problem can be derived similarly.

2.2 Alternative problem formulation based on edge vector

Since $\lambda_2(G)$ is computed based on the weighted Laplacian matrix of G, we can also denote $\lambda_2(G)$ as $\lambda_2(L)$, in which L is the weighted Laplacian matrix of graph G. According to Ghosh and Boyd [2006], the weighted Laplacian matrix L can be represented by the dot product summation of *edge vectors*. For an edge e connecting two nodes i and j, we define the edge vector $h_e \in \mathbb{R}^n$ as $h_{e_i} = 1$, $h_{e_j} = -1$, and all other entries 0. w_e is the non-negative integral weight on e. Suppose there are m edges in graph G. The weighted Laplacian matrix L of G is the $n \times n$ matrix:

$$L = \sum_{e=1}^{m} w_e h_e h_e^T. \tag{5}$$

In fact, the weighted Laplacian matrix expressions in Eq. (5) and in Eq. (2) are equivalent.

According to the edge vector description of L, the flight routes addition problem (3) can be written as:

$$\begin{aligned} &\max \lambda_2(L_0 + \textstyle\sum_{e=1}^{|P|} x_e w_e h_e h_e^T) \\ \text{s.t. } &\mathbf{1}^T x = k, \\ &x \in \{0,1\}^{|P|}, \\ &w_e \in \mathbb{I}, w_e < W, \end{aligned} \tag{6}$$

where L_0 is the weighted Laplacian matrix of the existed network $G(V, E_0)$. A fixed number of k edges are to be added. P is the pre-determined set with size $|P|$ which contains the candidate edges to be added. e is the index for candidate edges in P. x_e is a boolean variable, in which 1 means that edge e in P is in the set of ΔE in (3) and 0 means that edge e in P is not in the set of ΔE. x is a vector consisting of x_e whose length is $|P|$, illustrating which candidate edges are to be added and which are not. We observe that $\lambda_2(L)$ is a function of x, which can be denoted as $\lambda_2(L(x))$.

2.3 The flight routes addition problem is NP-hard

Theorem 1. The non-negative integral weighted flight routes addition problem is NP-hard.

Proof 1. The proof is a process of two steps reduction.

Step 1: As we defined in the problem, the weights are non-negative integral and bounded by W. Let's set W as 2, now all the weights can only be 0 or 1. The problem becomes a regular unweighted problem.

Step 2: Another reduction happens in set P. Based on the same node set V, we construct a complete graph G_c and denote the edges of the complete graph as set E_c. Now if we reduct P to all the other edges which do not exist in E_0, i.e. the set $(E_c - E_0)$, our problem is transformed to the maximum algebraic connectivity augmentation problem (Aoyama[2008]), which is proved to be NP-hard.

Since the flight routes addition problem can be reducted into a proved NP-hard problem, it is also NP-hard.

As a result, we seek heuristic algorithms to solve the flight routes addition problem.

3. MODIFIED GREEDY PERTURBATION

The second smallest eigenvalue $\lambda_2(L)$ is called the *algebraic connectivity*, and the corresponding normalized eigenvector is called the *Fiedler vector* (Fiedler[1973]). In Ghosh and Boyd [2006] present a greedy local heuristic, they add the k edges one at a time based on the calculation of the Fiedler vector. In this section their approach is analyzed and extended to the weighted problem.

The heuristic by Ghosh and Boyd [2006] is in an iterative manner. In each round the method adds the k edges one at a time, each time selecting the edge e_{ij} with the largest value of $(v_i - v_j)^2$, where v is the Fiedler vector of the current round Laplacian. In this section we extend their heuristic into the Modified Greedy Perturbation Algorithm (MGP).

3.1 The bond between λ_2 and L

According to Mohar [1991], no matter L is weighted or not, the algebraic connectivity can be computed by:

$$\lambda_2(L(x)) = \min\{\frac{y^T L(x) y}{y^T y} \,|\, y \neq 0, \mathbf{1}^T y = 0\}, \tag{7}$$

where y is a $n \times 1$ non-zero vector and it is orthogonal with all-one vector $\mathbf{1}$.

Furthermore, Eq. (7) can be transformed into:

$$\lambda_2(L(x)) = \min\{\frac{y^T L(x) y}{\|y\|^2} \,|\, y \neq 0, \mathbf{1}^T y = 0\}, \tag{8}$$

in which we substitute vector y with normalized vector $v = y/\|y\|$ and we have Eq. (9):

$$\lambda_2(L(x)) = \min\{v^T L(x) v \,|\, \|v\| = 1, \mathbf{1}^T v = 0\}, \tag{9}$$

When the normalized vector v in Eq. (9) is also a Fiedler vector, since

$$\lambda_2(L(x))v = L(x)v, \tag{10}$$

v^T is multiplied to the left of the both sides of Eq. (10):

$$v^T \lambda_2(L(x))v = v^T L(x) v. \tag{11}$$

Because vector v is normalized,

$$v^T \lambda_2(L(x))v = \lambda_2(L(x))(v^T v) = \lambda_2(L(x)) = v^T L(x) v. \tag{12}$$

Therefore if v is a Fiedler vector, the minimum in Eq. (9) can be achieved.

$$\lambda_2(L(x)) = v^T L(x) v, \tag{13}$$

Eq. (13) shows that the Fiedler vector v is the bond between the algebraic connectivity λ_2 and the Laplacian matrix L, both in unweighted and weighted cases.

3.2 Maximize $\lambda_2(L(x))$ in the weighted problem

Based on Formulation (6), the weighted Laplacian matrix after flight routes addition is:

$$L(x) = L_0 + \sum_{l=1}^{|P|} x_e w_e h_e h_e^T \tag{14}$$

The partial derivative of $\lambda_2(L)$ with respect to x_e gives the first order approximation of the increase for $\lambda_2(L)$, if the edge e is added to graph G. According to Eq. (13),

$$\frac{\partial}{\partial x_e}\lambda_2(L(x)) = v^T \frac{\partial L(x)}{\partial x_e} v. \tag{15}$$

Plug Eq. (14) into Eq. (15). Since the Laplacian L_0 before flight routes addition is not a function of x_e, we obtain:

$$\begin{aligned}
&\frac{\partial}{\partial x_e}\lambda_2(L(x)) \\
&= v^T \frac{\partial L(x)}{\partial x_e} v \\
&= v^T \frac{\partial (L_0 + \sum_{l=1}^{|P|} x_e w_e h_e h_e^T)}{\partial x_e} v \\
&= v^T (w_e h_e h_e^T) v = w_e (v^T h_e)(h_e^T v) \\
&= w_e (v_i - v_j)^2.
\end{aligned} \tag{16}$$

Thus the unweighted approach is extended into the Modified Greedy Perturbation Algorithm, which picks one edge from remaining candidates with maximal $w_e(v_i - v_j)^2$ at each round, where v_i and v_j are the ith and jth items of the Fiedler vector v of the current Laplacian L. The complete algorithm is listed in Algorithm 1.

Algorithm 1 Modified Greedy Perturbation

1: given graph $G(V, E_0)$, candidate edge set P and all the edge weights in E_0 and P
2: let $E = E_0$
3: **for** 1 to k **do**
4: calculate $\lambda_2(G(V, E))$ and its Fiedler vector v
5: $e_{ij} = \arg\max_{e_{ij} \in P} w_{ij}(v_i - v_j)^2$
6: $E = E + e_{ij}$
7: $P = P - e_{ij}$
8: **end for**
9: output $G(V, E)$

The computation complexity of Algorithm 1 heavily depends how fast we can compute the second smallest eigenvalue of the Laplacian matrix L. Line 4 takes polynomial $O(n^Z)$ arithmetic operations if Lanczos (Cullum[1985]) or EIGIFP (Money[2005]) algorithm is applied. Line 5 takes $|P|$ operations. Line 6 and Line 7 both only need 1 operation. Thus the total complexity is $k \cdot O(n^Z + |P| + 2) = O(kn^Z + k|P|)$. Because $|P|$ is usually smaller than n^2 and Z's of all the currently eigenvalue computation methods are bigger than 2, we have total complexity $O(kn^Z)$.

4. WEIGHTED TABU SEARCH

Besides local greedy search like the MGP, which usually results in a local optimum, a global search can perform better in solving the maximum. Given the graph $G(V, E_0)$, we are interested in finding k edges from set P to add to G, which together maximally improve $\lambda_2(G(V, E_0))$ to $\lambda_2(G(V, E_0 + \Delta E))$. The global exhaustive search gives out $\binom{|P|}{k}$ different λ_2 and the biggest λ_2 is the final solution. Generally the number $\binom{|P|}{k}$ is so huge that the running time of the global exhaustive search is very long.

4.1 Tabu Search Preliminary

As a solid alternative of the exhaustive search, *tabu search* (Glover[1989, 1990]) improves the efficiency of the search process by keeping track of the searching trajectory and operating flexible evaluation criteria. An appropriate implementation of memory is the key feature of tabu search. Tabu search records the the best solution and the searching trajectories to the recent found solutions. The idea of the tabu search is to permit the method to go beyond local optimum while still running into a better solution value at each step. The tabu restriction can not be violated except when the searching meets *aspiration criteria* (Glover [1989]).

4.2 First attempt in Airport Transportation Network Connectivity with Tabu Search

To the best of our knowledge, Kincaid [2008] and Vargo [2010] are the first and only research work that introduces algebraic connectivity into airport transportation network and the tabu search is implemented to solve their optimization problem. However, their consideration on algebraic connectivity is restricted on the classical unweighted Laplacian matrix.

Although they also pursuit the maximum of $\lambda_2(G)$, their problem is to design an airport transportation network which requires preserving the related graph's degree distribution and keeping the graph connected. The problem is different from ours and the comparison with their algorithm would not be appropriate, but Kincaid [2008] and Vargo [2010] are the only research work studying algebraic connectivity in airport transportation network so far, so in this paper we re-design our own tabu search algorithm as the Weighted Tabu Search (WTS) to fit in the flight routes addition problem and then compare it to the MGP.

4.3 Weighted Tabu Search in Flight Routes Addition Problem

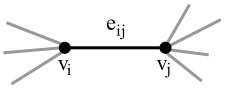


Fig. 2. The neighbor of edge e_{ij} in current solution s.

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

Dynamic Neighbor The WTS looks for the next round solution s' inside the neighbor $N(s)$ of the current solution s. After s' is chosen, the following round solution will be selected from its neighbor $N(s')$.

Instead of the swapping operation in Kincaid [2008] and Vargo [2010], we define $N(s)$ for our problem. Inside a given initial solution s, there are k edges to be added. The pth $(1 \leq p \leq k)$ edge e_{ij} in solution s connects two nodes v_i and v_j, which is shown in Fig. 2. Thus all the edges incident to v_i or v_j and in set P constitute the sub-neighbor $N(s,p)$ of solution s at the pth edge. The edges which already exist in G or are not in candidate set P are not displayed in Fig. 2. To prevent $N(s,p)$ from being empty, a random jump inside P is also included in $N(s,p)$, which jumps to any edge in P but not current edges 1 to k in solution s. If the random jump gives out an existing edge in $N(s,p)$, then we execute another random jump. The neighbor of current solution s is $N(s)$, which is the union of the sub-neighbors of every edge in s:

$$N(s) = \bigcup_{p=1}^{k} N(s,p). \tag{17}$$

Notice that each of the k new edges will be picked from its own sub-neighbor $N(s,p)$ and form s' together.

Algorithm 2 Weighted Tabu Search

1: given $G(V,E_0)$, P and the edge weights in E_0 and P
2: random pick k edges from P to construct s_0
3: $s = s_0$, $\lambda_2^* = 0$, $s^* = s_0$, T is set to a empty queue with the pre-fixed size $|T|$
4: **for** *iteration* $= 1$ to Φ **do**
5: **for** $p = 1$ to k **do**
6: construct $N(s,p)$ of the pth edge in s
7: **end for**
8: **while** 1 **do**
9: pick one edge p' from each $N(s,p)$ to construct s'
10: **if** $\lambda_2(s') > \lambda_2^*$ **then**
11: $s = s'$, update T
12: $\lambda_2^* = \lambda_2(s)$, $s^* = s$
13: break
14: **end if**
15: **if** s' is not in T **then**
16: $s = s'$, update T
17: break
18: **end if**
19: **end while**
20: **end for**
21: output λ_2^* and s^*

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

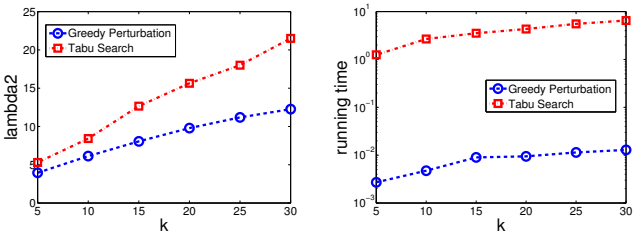
Aspiration Criteria The WTS offers an opportunity where the search can violate tabu. When a searching move is improving the solution value and leading to the best observed value of λ_2, this move can be performed. So the best observed value λ_2^* needs to be updated throughout the searching process.

The complete Weighted Tabu Search (WTS) is shown in Algorithm 2. In Algorithm 2, Line 2 sets an initial solution s_0. Line 3 initializes the parameters, where s is the solution in the current round, λ_2^* and s^* record the best λ_2 and its corresponding s respectively. Line 4 shows that the algorithm terminates after Φ rounds. Line 5 to 7 constructs the sub-neighbors of the current solution. Line 9 forms s' from $N(s)$. Line 10 to 14 checks the aspiration criteria. Line 15 to 18 checks whether the move from s to s' is in the tabu list T.

5. SIMULATION

We use simulations to compare the performance of two weighted algorithms. By default 20 nodes are generated one by one at random locations in a 2D plane. Each new node is ensured to get connected with the distributed ones. Thus these 20 airports (nodes) form a connected network with random topology. We denote this topology as $G(V,E_0)$. Once $G(V,E_0)$ is fixed, the existed edges in E_0 and the edges in candidate edge set P are assigned with weights with $\{w_{ij}|w_{ij} = 1, 2 \text{ or } 3\}$. The number of edges k to be added is set as input. The total round Φ in WTS is set to be 1000.

5.1 The impact of k



(a) (b)

Fig. 3. Impact of k.

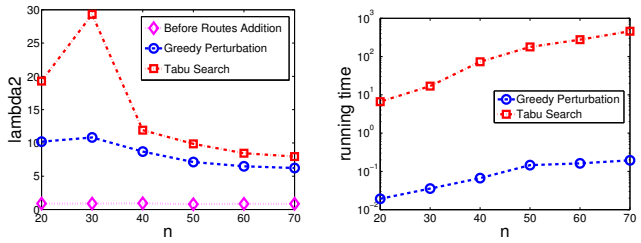
k is the number of routes added to graph $G(V,E_0)$ by two algorithms. This simulation is operated based on the default settings with k varying. From Fig. 3a we observe that when more routes are added to G, the algebraic connectivity increases. In other words, the more routes added to the network, the better connection the graph gains. The WTS offers better $\lambda_2(G)$ than the MGP does. However, the running time of the WTS is longer than the MGP, which is shown in Fig. 3b. Note that the running time axis is in logarithmic scale.

5.2 The impact of network size n

In this simulation we fix $k = 30$ and vary the network size n. When generating different sized networks, their topologies can not be kept same. For a better evaluation, the different sized networks with initial $\lambda_2(G)$ between 0.8 and 1 are selected. Fig. 4a illustrates that both MGP and WTS improve the value of λ_2 by at least 5 and the WTS performs better than MGP. Fig. 4b shows the running times of both algorithms increase with n increasing. The running time axis is still in logarithmic scale.

The trade-off is that when the number k of routes to be added or the network size n increases, the WTS always finds a better solution than the MGP does, however, the computation complexity of the WTS is much higher than the MGP. According to the simulations, if the problem

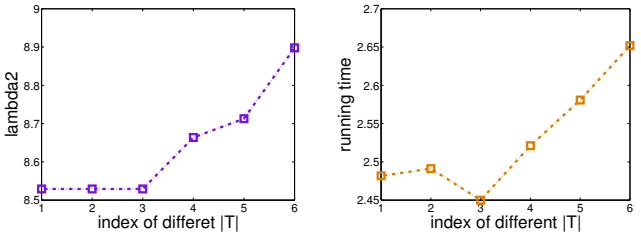
is about a local small area with fewer airports, the WTS should be selected for improving the network performance drastically; if it is a global nationwide problem, the MGP should be adopted to provide an efficient computation speed with an acceptable performance enhancement. The next two sets of simulations both focus on the parameter settings of the WTS.



(a) (b)

Fig. 4. Impact of network size n.

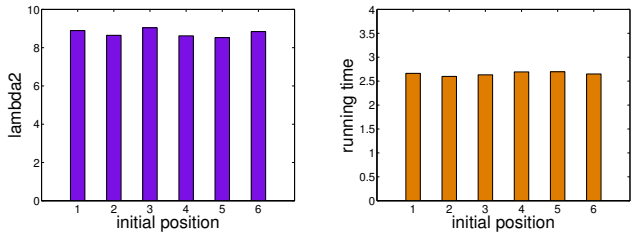
5.3 The impact of tabu list size $|T|$



(a) (b)

Fig. 5. Impact of $|T|$.

In this simulation we maintain the default network size and set $k = 10$. We hold the same initial positions of the k routes to be added. In fact, the indices $|T| = 1, 2, 3, 4, 5, 6$ represent the tabu list sizes $2, 5, 10, 20, 40, 80$. Fig. 5a shows that with bigger $|T|$, the WTS finds better λ_2. The reason is that the bigger tabu list contains more information which helps the algorithm search more other neighbors and get out of the local optimum. However, bigger tabu list leads to slower computation, which is shown in Fig. 5b.



(a) (b)

Fig. 6. Impact of initial position.

5.4 The impact of different initial positions

In this simulation $n = 20$, $k = 10$, $|T| = 40$. Fig. 6a and Fig. 6b illustrate that the WTS is pretty sensitive to initial condition. The x-axis is the index of different initial conditions. Six different initial positions of k added routes produce different λ_2 and each convergence time of them also varies slightly.

6. CONCLUSION

Based on practical operational viability, we investigate the airport transportation network application of the weighted algebraic connectivity by formulating the weighted flight routes addition/deletion problems. We derive the Modified Greedy Perturbation Algorithm to compute the locally optimal weighted λ_2 facilitated by the work of Ghosh and Bosh and develop a new Weighted Tabu Search suitable for the weighted flight routes addition/deletion problems. We compare the MGP and WTS through the simulations and provide our suggestions on how to pick the appropriate algorithm in airport transportation network performance enhancement of different sizes.

REFERENCES

M. Barahona and L. Pecora. Synchronization in small-world networks. *Physical Review Letters*, 89(5), 2002.

J. Cullum and R. Willoughby. *Lanczos Algorithms for Large Symmetric Eigenvalue Computations: Theory*. Birkhuser, 1985.

M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematics Journal*, 23:298–305, 1973.

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

F. Glover. Tabu search-part i. *ORSA Journal on Computing 1*, pages 190–206, 1989.

F. Glover. Tabu search-part ii. *ORSA Journal on Computing 2*, pages 4–32, 1990.

R. Guimera and L.A.N. Amaral. Modeling the world-wide airport network. *European Physical Journal B*, 38:381–385, 2004.

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

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

Concept of Operations for the Next Generation Air Transportation System. Joint Planning and Development Office, Version 2.0, June 2007.

R. Kincaid, N. Alexandrov, and M. Holroyd. An investigation of synchrony in transport networks. *Complexity*, 14(4):34–43, 2008.

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

J. Money and Q. Ye. Algorithm 845: Eigifp: A matlab program for solving large symmetric generalized eigenvalue problems. *ACM Transactions on Mathematical Software*, 31(2):270–279, June 2005.

D. Mosk-Aoyama. Maximum algebraic connectivity augmentation is np-hard. *Operations Research Letters*, 36 (6):677–679, 2008.

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