Research on Route Search of Connecting Flights in International Airline Network

Ruixin Ma1*, Xian Chen1, Jian Liu1, Jinfu Zhu1

1Civil Aviation College, Nanjing University of Aeronautics and Astronautics, Nanjing, Jiangsu, 211106, China
*Corresponding author’s e-mail: 18222712693@163.com

Abstract. With the development of global aviation industry, a huge international airline network has been formed. In order to solve the problem that the connecting path search method takes long time in global airline network, the objective of this paper is to find out all the target connecting flights paths fast between an OD pair. According to the features of the connecting path search in the schedule-based network, the solving method is divided into three stages. In the first stage, we build a time-space graph. In the next stage, three-step cutting is used to reduce the scale of graph. After that, we apply an algorithm which is combined with depth-first and breadth-first search algorithm to enumerate the paths. It can be seen that our algorithm is suitable for multi-constraints, big data and searching all paths.

1. Introduction
With the rapid development of global aviation industry and the increasing complexity of international airline network, the scale of international airline network has become larger. Passengers from one country's non-portal airport to another country's portal airport or non-portal airport may not have direct flights. Therefore, if we can make use of the airline network in other regions or countries, passengers can get everywhere or travel worldwide. However, because of the psychological complexity of passengers' choice, only one or a few shortest paths often fail to meet the passengers' needs. Meanwhile, it may take a lot of time to find all the target connecting flights between an OD pair in the huge global airline network. All these are the urgent problems to be solved in this paper.

Finding the target paths in a schedule-based network is similar to finding the K shortest paths in a network. In recent years, finding paths based on schedule has become a hot research topic, generally based on urban public transport system. Hamdouch et al. [1] proposed a new schedule-based equilibrium transit assignment model and adopted the method of successive averages by solving a dynamic programming assumption. Xu, T et al. [2] proposed a constrained Yen* algorithm, which adopted the heuristic strategy of A* algorithm to solve the K-multiple constrained shortest paths problem. Xu, W.T. et al. [3] considered a schedule-based transit network and proposed a new algorithm to enumerate these paths and compared it with other algorithms. Yusof et al. [5] developed a flight connections searching system that was able to give suggestion to users of broader destination network by applying the concept of graph theory and partial constraint satisfaction problem.

We consider a schedule-based global airline network, in which each node has a scheduled departure and arrival time of flights. Based on this, our paper aims to propose a fast search algorithm to enumerate all the connecting flights between any OD pairs on a global scale. The difficulty lies in how to select small scale networks for search in large scale networks and then find the feasible path combination. Our
solving method is to come up with the method to cut the scope according to time and space, and use the combination search algorithm to find the paths.

2. Algorithm
At present, the international airline network contains thousands of airports and millions of flight schedule data. How to reduce the search scope continuously and find all the target flights from the large schedule-based network is the problem to be solved in this part. Meanwhile, with the change of OD city pairs, the small-scale networks that need to be used are different. Based on it, our solving approach can be divided into three stages.

2.1. Stage I. Build time-space graph
In order to solve the problem, we convert a physical network to a time–space network. Abcissa is time and ordinate is space. Each node means an arrival or departure airport. Each link means a different flight arriving from an airport to the other. We simulate a local graph of a small-scale time-space graph, as shown in figure 1. Each link and node has many attributes, including arrival time, departure time, duration time, aircraft type, district, etc. In the international airline network, there are more than 3,000 airports and more than 2 million flights in a month. If we convert it to a time-space network, it must be a huge time-space graph with too many nodes and links.

2.2. Stage II. Cut graph
Considering the graph size, it must be cut to improve the speed of computing. In this global time-space graph, some links and nodes are useless for searching the target path between an OD pair. Therefore, a three-step cutting process is used in order to reduce the running time to enumerate paths.

Step 1: Nodes that exceed 72 hours compared with the origin node will be deleted. Links related with those nodes will also be cut. Usually, it is seldom that passengers spend more than three days on the transportation.

Step 2: Delete the link which is the arrival flight of the origin node and the link which is departure flight of the destination node. There is no need to keep the links that arrive at origin airport and leave from destination airport.

Step 3: Nodes without link connections are cut. The cutting process is shown in figure 2.
2.3. Stage III. Search Paths of connecting flight

This stage proposes a search algorithm combining depth-first algorithm with breadth-first algorithm, called BFS_D algorithm, to get all the available connecting flight paths of any OD pairs.

One of the ways to improve search efficiency of BFS_D is to create flight connection tree before searching. Firstly, the breadth-first search algorithm is used to search the flight connection tree’s nodes, and then the depth-first search algorithm is used to search the follow-up flights, which satisfies the transit connection time to form a new flight connection tree. Then, we go through the generated flight tree to find the final possible results. Depth-first search algorithm finds the path by restricting the maximum number of nodes on a path, that is, specifying the maximum depth of the tree. If the maximum times of transit is not more than 2, the maximum depth of searching is 3, regardless of whether the path endpoint is the destination or not. Small-scale flight connection tree is shown in figure 3. Compared with the traditional search method, the search scope is greatly reduced and the search speed is improved by BFS_D.

Also, to improve search efficiency, we use hash table data structure, which is accessed directly based on key-value. Hash table is different from arrays, linked lists and trees, whose time level are O (N). But for a hash table, only O (1) time level is needed when searching. Key is an airport’s three-character code, value is a record of all flights departing from the airport, including attributes about the time of departure and arrival, flight number, aircraft type, flight duration, etc. After the flight tree is searched by BFS, hash table is used to store the results of the flight connection tree, and depth search is used to get the
path satisfying the conditions in the hash table. Because both depth-first and breadth-first algorithms need to use hash table, only one hash table is needed so that we can improve programming efficiency.

Implementation steps:
Step 1: From the origin airport node, the breadth-first algorithm is used to generate the flight tree node, that is, to search the flight information related to the origin airport node, and form a subset A of arrival airports. Store the results by hash table;
Step 2: Repeat the steps mentioned above to find the follow-up flights of all flights in set A and form a subset B1 of flights;
Step 3: Calculate the difference between arrival flights in B1 and departure flights in set A. If the difference meets the condition of transit time, store this flight and form a new flight set B. Otherwise, abandon this record and stop searching down;
Step 4: Repeat the above steps to find the follow-up flights of the airports in set B. Form flight subset C1, abandon the value of flights that do not meet the transit connection time, and form set C. Store the results by hash table;
Step 5: After the search tree is generated, the flight connection path is searched by depth first. If the arrival airport is equal to the destination airport in A set, the non-stop routines are completed;
Step 6: Find out the flights arriving at the airport equal to the destination airport in set B, and calculate the total time spent on this routine. If they do not meet the conditions, including detour rate and total flight duration, we will abandon this record and no longer downward. In this way, we can decrease the search scope and complete one-stop routines search;
Step 7: Repeat the Step 6 search method in set C to complete the two-stops connecting flights routines.

3. Computational Experiments

3.1. Results
In the test experiment, the used test data is from OAG Travel Planner global data for December 2018, a total of nearly 2.2 million flight data, with 3859 airports. All the codes were written in Python 3.6 and the computational experiments were carried out on a computer, which has 128 Mbytes of RAM and a 1.4 GHz Intel Core i5 processor running MAC OS. The data is stored in Oracle database, and data preprocessing is carried out before use.

While searching the path, it should also meet the rules of connecting flights when passengers travel. The real flight connecting is very complex, thus we make some rules as follows:
- The times of transit is no more than twice. The times of transit can be 0, 1 and 2, which corresponds to non-stop, one-stop and two-stop respectively. Too many times of transits can cause discomfort.
- The departure time of follow-up flights should be later than the arrival time of pre-order flights.
- The departure airport of follow-up flights should be the same with the arrival airport of pre-order flights.
- Transit connection time: the minimum connecting time is 90 minutes, when it is domestic transit; the minimum connecting time for domestic-to-international transfer is 120 minutes; when it is international transfer, the minimum transfer connection time is 180 minutes.
- Three kinds of detour rates are available: low detour rate (135%), medium detour rate (175%) and high detour rate (200%).
- Our shortest path is defined as the shortest total time spent in the whole journey in our paper. For a two-transit connecting flight, the total time spent is the sum of three flights duration and two waiting time of transit.

The departure airport we choose is Kunming Changshui International Airport, China (KMG), and the arrival airport is Charles de Gaulle Airport, Paris (CDG). The departure time is December 22. The times of transits is no more than twice, the maximum detour rate is 150%, and the maximum waiting time of transit is 6 hours. The results include 1226 schemes. And the first three paths are as table 1 below.
| Plan | Routines | Departure time | Arrival time | Flight No. | Duration(H:M) |
|------|----------|----------------|--------------|------------|---------------|
| 1    | KMG-SVO  | 12-22 09:50    | 12-22 14:40  | 8L881      | 15:05         |
|      | SVO-CDG  | 12-22 15:55    | 12-22 17:55  | AF1045     |               |
| 2    | KMG-PEK  | 12-22 20:50    | 12-23 00:10  | MU5492     | 15:10         |
|      | PEK-CDG  | 12-23 01:00    | 12-23 05:00  | AF381      |               |
| 3    | KMG-CTU  | 12-22 23:50    | 12-23 01:05  | EU2224     | 15:30         |
|      | CTU-FRA  | 12-23 01:55    | 12-23 05:40  | CA431      |               |
|      | FRA-CDG  | 12-23 06:50    | 12-23 08:00  | LH1026     |               |

3.2. Analysis

In order to test the performance of the algorithm, three experiments were carried out.

Experiment 1 compares the influence of waiting time of transit and maximum rate of detour on running time. When comparing the effects of the waiting time of transit, the maximum detour rate is set to 150% and times of transit is no more than twice. From figure 4, we can see that the average running time of the algorithm is increasing when waiting time is increasing. It can be seen that the waiting time of transit is a key factor affecting the running time of the algorithm. In theory, the number of flights that can be transferred will be reduced when the waiting time for transfer is reduced. The effective network scale is reduced with small nodes to be expanded when searching. Therefore, the longer the waiting time for transit, the longer the running time of our algorithm.

Experiment 2 compares the effects of different maximum detour rates from figure 5, the times of transit is set to twice. Although the maximum detour rate is different, the average running time of the algorithm is around 30-40 milliseconds. It can be seen that the deviation rate has little influence on the algorithm. It is because the possible path is generated before the rate of deviation is judged.

Experiment 3 compares the running time between Yen algorithm and our algorithm. As can be seen from figure 6, when the number of paths is small, the running time difference between the two algorithms is not obvious. With the increase of the paths, the running time of Yen algorithm increases, and our algorithm runs faster. From the analysis of experimental results, when it is more than 100 connecting flights plans, the average running time of BFS_D algorithm is reduced greatly. It can be seen that our algorithm is more suitable for multi-constraints, big data and enumerating all the paths.
4. Conclusion
In this paper, an algorithm is proposed to search all connecting flights paths between any OD pairs in an international airline network. Our solving method is divided into three stages. First, we build a time-space graph. Second, we reduce the scale of graph by cutting useless nodes and links. Third, we apply a BFS_D algorithm to find the paths. The innovation of our method lies in: (1) before searching, we first find ways to cut global airline network to reduce scope of searching. (2) we generate flight tree according to the airport and apply BFS_D searching algorithm to find out all the paths quickly. Finally, the performance of the algorithm is proved by modifying the constraints of the waiting time of transit and the maximum detour rate and comparing with Yen algorithm. It can be seen that our algorithm has short running time and is suitable for solving large data and enumerating all the paths search.

References
[1] Hamdouch, Y., Ho, H.W., Sumalee, A., (2011) Schedule-based transit assignment model with vehicle capacity and seat availability, Transportation Research Part B: Methodological., 45(10): 1805-1830.
[2] Xu, T., Ding, X.L., Li, J.F. (2014) Research on KMCSP of International Route Network Route Search, Journal of Southwest Jiaotong University., 49 (01): 153-159.
[3] Xu, W.T., He, S.W., Song, R., (2012) Finding the K shortest paths in a schedule-based transit network, Computers & Operations Research., 39:1812-1826.
[4] Li, J.F., Li, T.J., (2014) Research on Fast KCSP Algorithms for Searching Connecting Paths in Airline Networks, Applied Mechanics and Materials., 505-506:1005-1013.
[5] Yusof, S.Z.M., Asirvadam, V.S., Hassan, M.F., (2010) Flight connections multi-leg searching by adopting partial constraint satisfaction, International Symposium in Information Technology., 1:1-4.
[6] David Canca, Alejandro Zarzo, PedroL. González-R, EvaBarrena and Encarnación Algaba. (2013) A methodology for schedule-based paths recommendation in Multimodal public transportation networks, Journal of Advanced Transportation., 47: 319-335.