How many passengers are planned to fly around the world?

The calculation of the total capacity between an origin and destination in the airline industry

Author:	Laura Wiersma
Supervisor:	prof. dr. R.D. van der Mei
Date:	January 2015

KLM
AMSRP
Amsterdamseweg 55
1182 GP Amstelveen
The Netherlands

VU University Amsterdam
Faculty of Sciences
De Boelelaan 1081a
1081 HV Amsterdam
The Netherlands

Preface

This research paper is part of the Dual Master's program Business Analytics at the Vrije Universiteit Amsterdam. The objective of this paper is to perform research using the business-related aspects of the program as well as the more fundamental aspects of mathematics and/or computer science.

Hereby, I would like to thank prof. dr. R.D. van der Mei for his help and support during the research. My thanks goes also to AMSRP (the decision support department for revenue management at KLM and Air France), in particular Gijs Hendrix and René Bloemers, for their help and ideas during the process.

Abstract

Revenue management is an important aspect in the airline industry. Creating more insight into the capacity development of competitors is a promising means to earning more revenue. When the competition increases the capacity on a route, it is probably because their is an opportunity on that route. KLM Royal Dutch Airlines (KLM) can react to this by increasing its own capacity and sell the extra tickets for a competitive price. Another aspect of competitive analysis is the determination of the bottleneck flights in an origin and destination (O&D). When there is a bottleneck flight detectable in an O&D of the competition, there is an opportunity for KLM on this leg. At the moment, KLM is not able to determine the bottleneck flights for an O&D with the current data. The research question answered in this paper is therefore:

How can the planned capacity and the bottleneck flights between an origin and destination, based on capacity and booking data, be determined?

In this paper, we answer this question using general network flow theory. We determine the maximum flow between an source (origin) and sink (destination) with the Ford-Fulkerson algorithm. The data sources used in this research are OAG and MIDT data. OAG data contain the scheduled flights with their configuration and route for each airline in the world. The data available in MIDT data are the actual bookings of a flightmonth and the future bookings for the coming months. To limit the processing time of the algorithm, the focus in this report is on only ten O&Ds.

Since OAG data is based on operating carrier information and MIDT data on marketing carrier information, the OAG data should be converted into marketing carrier based data. This is done by creating a set indexed on marketing carrier flight number. Next, the marketing flight numbers in the flightpath of the MIDT data are converted in operational flight numbers using the set created in the processing of the OAG data. All routes between an O&D are saved as a network flow in the form of an adjacency list. The nodes of the network are the flight numbers in the O&D. Finally, we implemented the Ford-Fulkerson algorithm to determine the maximum capacity of the O&D. This resulted in the total capacity together with the associated bottleneck flights for the O&Ds. The calculation time of the algorithm is less than three seconds for each of the ten O&Ds used in this paper.

The result of this research is a method to determine the planned capacity and the bottleneck flights between an O&D. The Ford-Fulkerson algorithm seems to be an accurate method to determine the planned capacity and the bottleneck flights. A challenging task for further research is the specification of the capacity for both marketing as operational carriers. Another promising line of research would be to involve the number of booking and the QSI by calculatin the capacity.

Contents

1 Introduction

1.1 Introduction KLM

This paper is written for KLM Royal Dutch Airlines (KLM), therefore this paper will begin with a short introduction of this company. KLM was founded in 1919, which makes it the oldest airine in the world. In 1920, KLM made its first scheduled flights between Amsterdam and London.

In May 2004 KLM merged with Air France, which made Air France-KLM the largest European airline group. Both airlines retained their individual identity, trade name and brand. Both airlines run their own operations from their respective hubs Paris-Charles de Gaulle and Amsterdam-Schiphol. Air France and KLM carry more than 77 million passengers per year. They operate 573 aircrafts enabling them to fly to 243 destinations in 103 countries.

Appendix 7.1 shows the organogram of KLM. This paper is written for the Pricing and Revenue Management (PRM) department of KLM, this department is part of the blue bordered section in the organogram. A short introduction to Revenue Management can be found in Section 2.2.

The tools used to evaluate the ticket prices and the competition are created by the decision support department. One of these tools is called Skyline, Skyline is a new Air France KLM tool to help (better) understand the competitive environment and learn about capacity developments of partners and competitors. It is feeded with OAG data (Section 3.1.1). The complete schedules of all airlines are shown in this tool together with the planned capacity. Appendix 7.2 shows a screenshot of Skyline, giving an overview of the planned capacity and the number of flights between the Netherlands and France for the summer of 2014.

MIDT-Studio is another tool used by the PRM department. MIDT-Studio gives an overview of flight information of bookings made through global distribution systems (GDS). The tool is feeded with MIDT data (Section 3.1.2). It is an application to analyze market share. The aim of the tool is to offer management information in a user-friendly environment. Appendix 7.3 shows a screenshot of MIDT-studio. This screenshot gives an overview of the bookings for the summer of 2014.

1.2 Problem and research question

To create more revenue in the future, KLM wants more insight into the capacity development of the competition. When the competition increases the capacity on a route, it is probably because there is a business opportunity on that route. KLM can react to this by increasing their own capacity and sell this extra tickets for a competitive price. Another aspect of competitive analysis is the determination of the bottleneck flights in an origin and destination (O&D) (Section 2.1). The bottleneck flight is the flight which restricts the total capacity between an O&D. When there is a bottleneck flight detectable in an O&D of the competition, there is an opportunity for KLM on this leg. It is also useful for the network department of KLM, because it can increase its capacity on the bottleneck flight to create more revenue. Skyline is currently based on legs (Section 2.1) and it is therefore not possible to determine the bottleneck flights for an O&D with the current data. The problem is that it is not possible to determine the total capacity and the bottleneck flights in an O&D with the data currently available in Skyline.

The research question answered in this paper is therefore:

How can the planned capacity and the bottleneck flights between an origin and destination, based on capacity and booking data, be determined?

The result of the research described in this paper is an algorithm which can determine the complete capacity between two stations. This algorithm is based on common flow network algorithms, since the flights in an O&D can be seen as a flow network (Section 2.3).

1.3 Outline

This paper is partitioned into four parts. The first part deals with the literature available for this subject. In the literature section we will first give commonly used terms in the airline industry, then a short introduction to revenue management and the section ends with important network flow theory. The second part deals with the methods and data description of this research. This section describes the available data and the approach we used to create an algorithm. In the following section the results are given. The last section states the conclusions and provides ideas for future work.

2 Literature background

In this section, we will give an overwiew of the literature used in this paper. First we will give commonly used terms in the airline industry, then a short introduction to revenue management and we will end with some network flow theory.

2.1 Terms in airline industry

To understand all terms used in this paper, we will give some definitions of common terms in the airline industry. The definitions are based on the International Air Transport Association (IATA) guidelines [4].

Aircraft configuration: Planned layout of the aircraft . i.e. the arrangement of the passenger cabin seating, usually in three categories: first, business and economy.

Capacity: The available seats in a given market and during a given time frame.

Code share: A commercial agreement between airline partners whereby a marketing partner publishes and manages marketing flight numbers for flights actually operated by a partner airline (operating carrier). Code Share is an agreement between two or more carriers to share each other's inventory for optimal utilisation of the same.

Destination: The ultimate stopping place of the journey as shown on the ticket.

Leg: The operation between a departure station and the next arrival station. In Figure 2.1 an example of a possible network is given. This network consists of the following three legs: AMS-LHR, LHR-JFK and AMS-JFK.

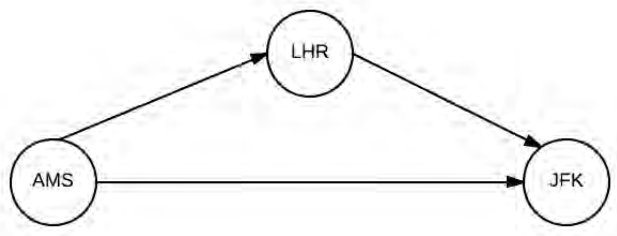


Figure 2.1: A possible network

Marketing carrier: The carrier that sells with its own code as part of a code-share agreement on a flight actually operated by another carrier.

Operating carrier: The carrier that actually operates the flight. It is the airline that holds the Air Operator's Certificate for the aircraft used for that flight.

Origin: The initial starting place of the journey as shown on the ticket.

Origin and Destination (O&D): When applied within a revenue management system, it is a term used to define controls and processes used to maximize revenue for a market. In Figure 2.1 the O&D AMS-LHR includes all three legs.

2.2 Revenue Management

According to Cross, Revenue Management "ensures that companies will sell the right product to the right customer at the right time for the right price" [2]. The most important question in the Revenue Management area is how much revenue can be generated from the marketplace with our existing resources.

Cross states in his book the following seven core concepts of Revenue Management [2]:

1. Focus on price rather than costs when balancing supply and demand.
2. Replace cost-based pricing with market-based pricing.
3. Sell to segmented micromarkets, not to mass markets.
4. Save your products for your most valuable customers.
5. Make decisions based on knowledge, not supposition.
6. Exploit each product's value cycle.
7. Continually reevaluate your revenue opportunities.

An important aspect is price sensitivity, the demand will probably depend on the price of the tickets. Number three of the seven core concepts is also very important, a business passenger will pay more than a leisure passenger, since the company will pay the ticket for the business passenger. To distinguish these different types, some conditions are added to the tickets. For example, a minimum length of stay or a weekend stay. This makes it possible to charge more for the business passengers than the leisure passengers [1].

2.3 Flow network

This section gives an introduction to the flow network theory used in this paper. The theorems described in this section are based on Roberts et al. [5] and Goodrich et al. [3].

A flow network N consists of a connected directed graph G with nonnegative weights on the edge (i, j), where the weight of an edge e is called the capacity $c_{ij} = w(i, j)$. This can be interpreted as the maximum amount of some unit that can flow through the edge. Flows are permitted only in the direction of the edge. The network has two distinguished vertices, s (source) and t (sink), of G, such that s has no incoming edges and t no outgoing edges. Figure 2.2 gives an example of a flow network with the capacity of each edge.

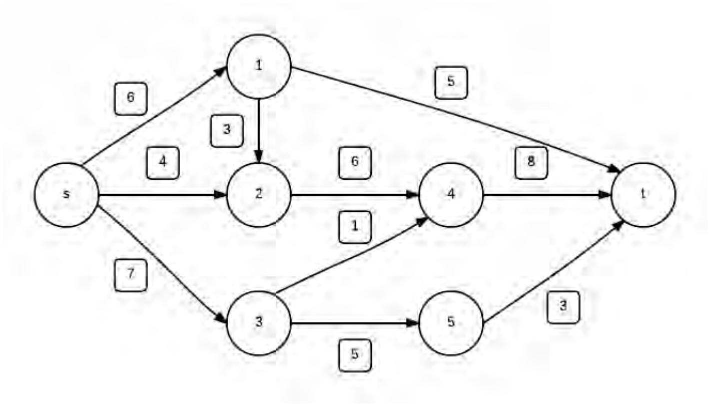


Figure 2.2: A flow network N

Let x_{ij} be the flow through the edge (i, j). The flow has to satisfy the following requirements:

- $0 \leq x_{ij} \leq c_{ij}$ (The capacity rule, the flow of an edge must not exceed the capacity of the edge)
- $\sum x_{ij} = \sum x_{ji},\ i \neq s, t$ (The conservation rule, which says that the same amount of flow needs to leave the vertex as enters it)

The value of the flow v is equal to the flow coming out from the source and also equal to the amount of flow going into the sink. In other words,

$$v = \sum_j x_{sj} - \sum_j x_{js} = \sum_j x_{jt} - \sum_j x_{tj} \tag{2.1}$$

A feasible flow x for the flow network of Figure 2.2 is illustrated in Figure 2.3. For each edge the capacity is bigger than the flow through that edge and the amount of flow going into each vertex is equal to the amount going out of the vertex. The value of the flow in Figure 2.3 is according to Equation 2.1 $v = 5 + 3 + 4 = 3 + 6 + 3 = 12$.

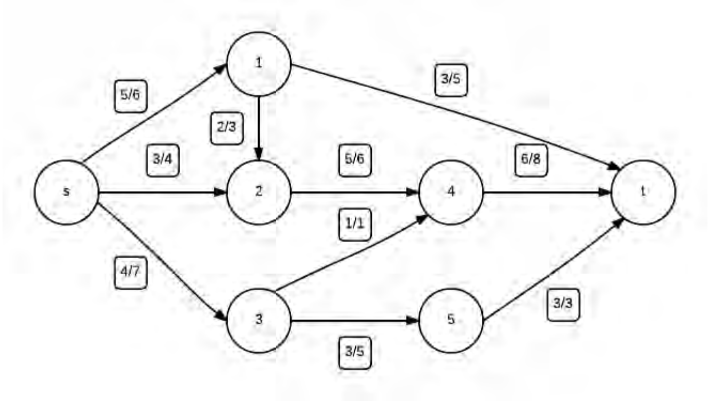


Figure 2.3: A flow x for N of Figure 2.2

2.3.1 Maximum flow

A maximum flow x^* is a flow with maximum value over all flows for N. An important problem to solve is how to find the flow, since a maximum flow is using the capacity most efficiently. We will give an example of a maximum flow algorithm in Section 2.4. The maximum flow x^* of Figure 2.2 is illustrated in Figure 2.4. According to Equation 2.1 the value of the maximum flow $v^* = 6 + 4 + 4 = 5 + 6 + 3 = 14$.

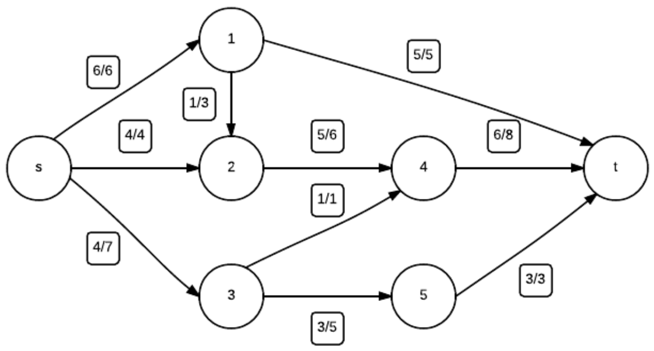


Figure 2.4: A maximum flow x^* for N of Figure2.2

2.3.2 Cuts

A cut is a division of the vertices of a flow network N into two sets, with s in the first set and t in the other. Let S (includes s) and T (includes t) be these sets with $V(N) = S \bigcup T$ and $S \bigcap T = \emptyset$. The set C is the set of edges leaving a vertex in S and entering a vertex in T. When the edges in C are removed, the path between S and T is also eliminated. Figure 2.5 shows a possible cut for the flow network of Figure 2.2. In this example $S = \{s, 1, 2, 3\}$, $T = \{4, 5, t\}$ and $C = \{(1,t), (2,4), (3,4)\,(3,5)\}$.

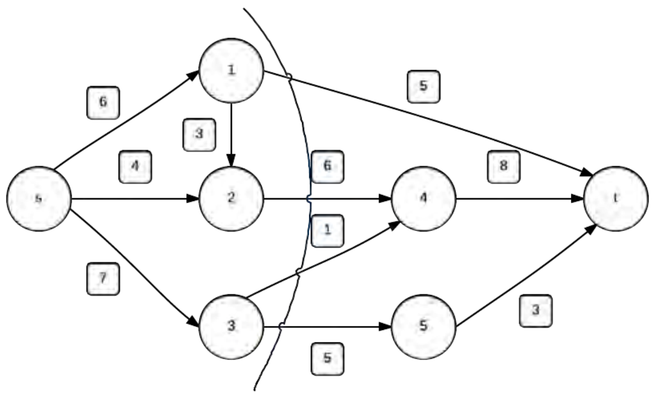


Figure 2.5: Flow network N of Figure 2.2 with cut

The capacity of a cut(S,T), $c(S,\, T)$, is $\sum_{i\epsilon S}\sum_{j\epsilon T} c_{ij}$. In the example in Figure 2.5 the capacity of the cut is $c(S,\, T) = c_{1t} + c_{24} + c_{34} + c_{35} = 5 + 6 + 1 + 5 = 17$. The capacity of a cut is an upper bound on any flow in the network. This holds also for the example in Figure 2.5, since the maximum flow is 14 and the capacity of the cut is 17. The next theorem states that this is true in any case (this theorem is proven in Roberts et al. [5]) :

Theorem 1. *The value of any (s,t)-flow is* $\leq$ *the capacity of any (s,t)-cut in a flow network.*

2.3.3 Residual capacity

The residual capacity for edge (i, j) is denoted by $\triangle x(i, j)$. The residual capacity is defined as

$$\triangle x(i,j) = c_{ij} - x_{ij}\ and\ \triangle x(j,i) = x_{ij}$$

The residual capacity of an edge going away from the source is the unused capacity of that edge with the current flow x. The residual capacity of an edge in the other direction is the used capacity of that edge.

2.3.4 Augmenting paths

An augmenting path for a flow x is a path π from the source to the sink with nonzero residual capacity, in other words for each edge (i, j) of π

- $x_{ij} < c_{ij}$ if (i, j) is a forward edge
- $x_{ij} > 0$ if (i, j) is a backward edge

2.3.5 Max-Flow Min-Cut Theorem

By Theorem 1 the following theorem holds (this theorem is proven in Roberts et al. [5]) :

Theorem 2. *In a directed network, the maximum value of an (s,t)-flow equals the minimum capacity of an (s,t)-cut.*

2.3.6 Adjacency list

There are three popular data structures to store a graph: the edge list, the adjacency list and the adjacency matrix. We decided to use the adjacency list structure in this report, we will therefore shortly discuss this data structure.

The adjacency list contains a vector of all vertices in the graph. Each vertex in this vector has its own list with edges. An edge consists of the end vertex of the edge and the corresponding weight of that edge. Figure 2.6 shows the adjacency list belonging to the network flow in Figure 2.2.

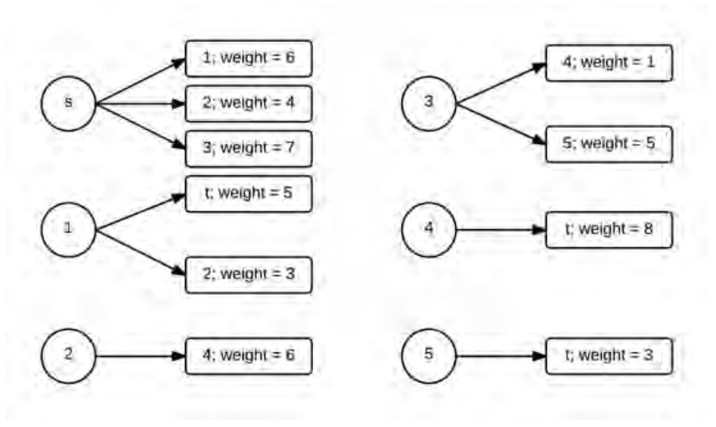


Figure 2.6: Adjacency list of Figure 2.2

2.4 Ford-Fulkerson Algorithm

The Ford-Fulkerson algorithm is an algorithm to compute a maximum flow in a network. This algorithm computes a maximum flow by applying the greedy method to the augmenting-path approach. The main idea of this algorithm is to incrementally increase the value of a flow in stages, where at each stage some amount of flow is pushed along an augmenting path from the source to the sink. The algorithm terminates when the current flow f does not admit an augmenting path. The pseudo-code for this algorithm is provided in Algorithm 3 in Appendix 7.4.

In Figure 2.7 the Ford-Fulkerson algorithm is applied to the network of Figure 2.2. The augmenting path chosen for each step is blue colored and the backward edges are red. In this case the Ford-Fulkseron algorithm terminates after six steps. The associated maximum flow is the sum of all red edges coming in the source: $6 + 4 + 4 = 14$.

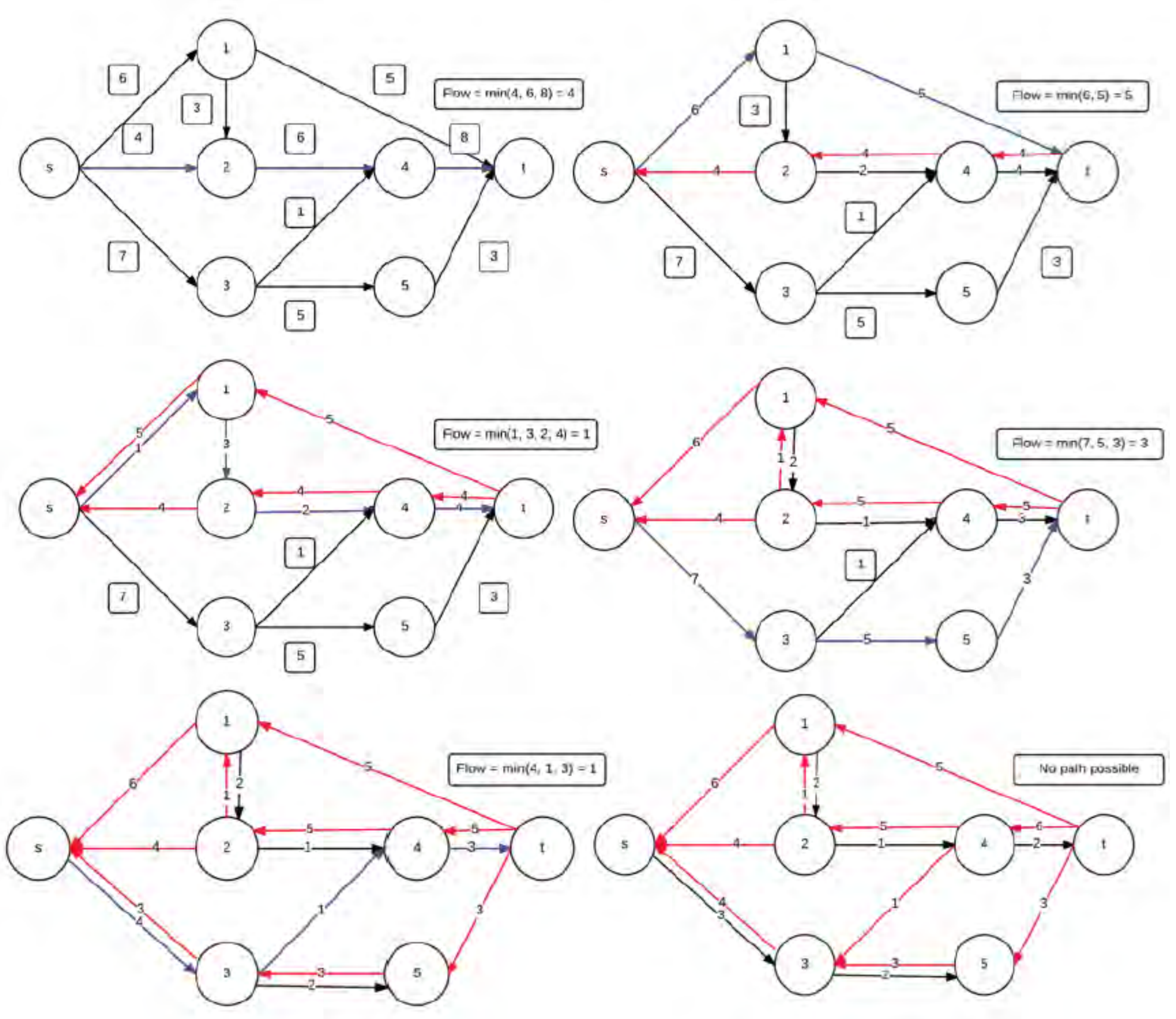


Figure 2.7: Ford-Fulkerson algorithm applied to the network of Figure 2.2

3 Methods and data description

This section begins with the description of the two data sources used for this research. Next, the data processing and the calculation of the capacity are provided.

3.1 Data description

3.1.1 OAG data

The data currently used in Skyline is OAG data. A complete description of this data-source can be found in Appendix 7.7. This dataset contains the scheduled flights with their configuration and route for each airline in the world. Every airline should file their schedule several months in advance. However, not every airline files its schedule correctly, KLM for example files the wrong aircraft configuration. Therefore, the data is not completely correct.

Currently, the OAG data is processed once a month and it contains information for the coming twelve months. The OAG data of May 2014 from the April snapshot are used in this paper. The number of OAG records used in this paper is 1,216,535.

3.1.2 MIDT data

The data used in MIDT-Studio is MIDT data. The description of all data fields can be found in Appendix 7.8. Currently, the MIDT data is processed monthly. It contains information from ten Global Distribution Systems (GDS). Limitations of the data are the absence of low cost airline bookings, since they are not distributed in GDS.

MIDT data contains the actual bookings of a flight month and the future bookings for the coming months. In this research the data of the actual bookings of May 2014 are used.

To limit the processing time of the algorithm, the focus in this report is on the following ten O&Ds. We have chosen these O&Ds, because multiple carriers fly with their own routes between these O&Ds:

- AMS - JFK
- AMS - LAX
- AMS - LAS
- AMS - NBO
- AMS - DLL
- AMS - BOG

- OSL - BCN
- AAL - LHR
- FRA - LAX
- JFK - LAS.

The total number of MIDT records processed in this paper is approximately 23,861 records.

3.1.3 Differences between data

These datasets cannot be used together, since the MIDT data is based on origin and destination and the OAG data is leg-based. Before both datas sources can be used together, the OAG data have to be translated in origin and destination based data.

Another difference between the two datasets is that MIDT data is based on marketing carrier flightnumbers and OAG data on operating carrier flightnumbers. Therefore, the OAG data should be converted into marketing carrier based data.

3.2 Approach

3.2.1 OAG data processing

First, the OAG data was processed. In this report, only the records from May 2014 are selected. Since the MIDT data consists of marketing carrier based data, we have to save the OAG records based on marketing carrier. First, the records were put in a set consisting of so called fltdatrecords. Table 1 shows the structure of a fltdatrecord.

Label	**Description**
long key_marketing_carflightnr	The marketing carrier flightnumber
short key_from	The departure station of the leg
short key_to	The arrival station of the leg
set<opcarrecord> dat_opcarset	A set containing all associated operation flightnumbers

Table 1: Structure of a flightpathrecord

After the first datacheck, it seemed that one marketing flight number can be flown with different operational flight numbers. Therefore, all operational flight records for one marketing flight number were put in a set consisting of opcarrecords. The structure of an opcarrecord is shown in Table 2.

Label	**Description**
long key_opcarfltnr	The operational carrier flightnumber
long dat_cap_array[7]	The capacity of all seats per day
long dat_f_cap_array[7]	The capacity of first seats per day
long dat_c_cap_array[7]	The capacity of business seats per day
long dat_ym_cap_array[7]	The capacity of ecnomy seats per day
long dat_day_indicator	Indicator to show if arrival day is different than departure day

Table 2: Structure of a opcarrecord

The capacity is recorded as the total capacity for the whole month per weekday, because a flight can have different configurations on different weekdays. On Monday for example the capacity is 142 and it flies 4 times in a month, the total capacity for this flight on Monday will then be 4 * 142 = 568. Besides storing the capacity per weekday, the capacity is also stored per seating class (economy, business and first).

The set of fltdatrecords is written to a .txt file and used for processing the MIDT data.

3.2.2 MIDT data processing

The next step is the processing of the MIDT data. First, the flight path of the MIDT record is stored in a vector of longs. The flight path consists of a carrier code and a flight number. Next, the via points of the record are stored in a vector of shorts. As told in Section 3.1.3, MIDT data contain only the marketing flight number, while OAG data contain the marketing and the operational flight numbers. Therefore, the MIDT data need to be transformed in operational carrier flight numbers.

To connect the marketing flight numbers in the flight path, we convert the marketing flight numbers using the set of opcarrecords created during the data processing of the OAG data (Section 3.2.1) in operational flight numbers. This is done with Algorithm 4 in Appendix 7.5. This Algorithm returns a vector of all possible flight paths for the OAG record. Table 3 gives the structure of a flightpathrecord. As told in Section 3.2.1 a marketing flight number can be operated with different operational flight numbers, therefore it is possible to have more than one possible flight path.

Label	**Description**
long key_opcarfltnr	The operational carrier flightnumber
long key_mcrfltnr	The marketing carrier flightnumber
short key_from	The departure station of the leg
short key_to	The arrival station of the leg
short key_opcarrier	The operational carrier of the flight
long dat_cap_array[7]	The capacity of all seats per day
long dat_f_cap_array[7]	The capacity of first seats per day
long dat_c_cap_array[7]	The capacity of business seats per day
long dat_ym_cap_array[7]	The capacity of ecnomy seats per day
long dat_day_indicator	Indicator to show if arrival day is different than departure day

Table 3: Structure of a flightpathrecord

For each possible flight path we want to check whether the flight path is valid. A flight path is valid when the different legs in an O&D are flown by the same operational carrier. The legs in an O&D can be flown by different operational carriers. In this report the focus is on O&Ds where all legs are flown by the same operational carrier, this is called operational O&Ds. In Figure 3.1 the O&D CDG-JFK is shown. The direct flight is flown by Air France, this flight is therefore used in this paper. The route with a stop in Amsterdam consists of a leg operated by Air France and a leg operated by KLM. That route is therefore not used in this paper.

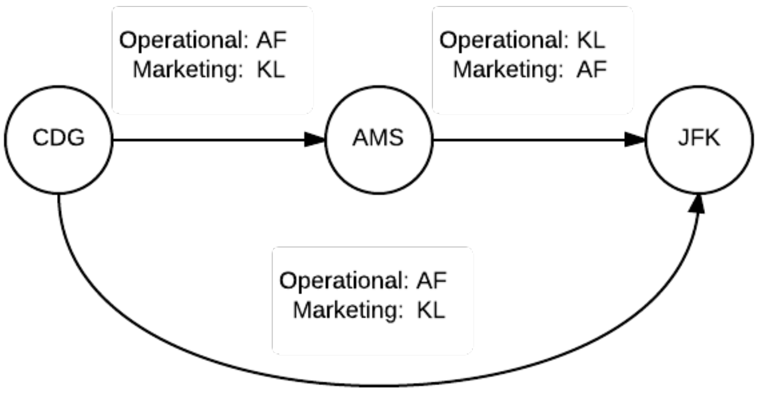


Figure 3.1: Example of marketing and operational O&D

Algorithm 5 in Appendix 7.6 determines whether a flight path is valid and stores the flight path in a set consisting of datrecords. The structure of a datrecord is shown in Table 4.

Label	**Description**
short key_origin	The origin of the O&D
short key_dest	The destination of the O&D
short key_opcarrier	The operational carrier
long key_fltpathidx	The index referring to the flightnumbers of the O&D
long key_viapointsidx	The index referring to the via points of the O&D
long dat_capidx	The index referring to the total capacity per flight
long dat_day_cap_idx	The index referring to the capacity per day per flight
long dat_f_cap_idx	The index referring to the first capacity per day per flight
long dat_c_cap_idx	The index referring to the business capacity per day per flight
long dat_ym_cap_idx	The index referring to the economy capacity per day per flight
long dat_day_indicator_idx	The index referring to the day indicators of the O&D

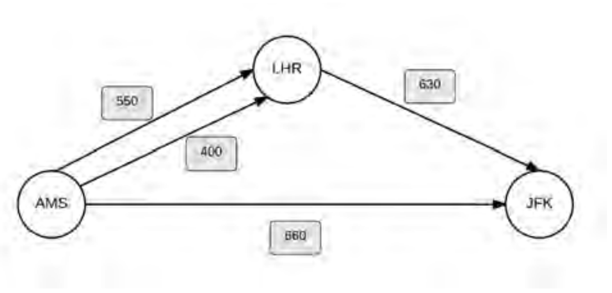
Table 4: Structure of a datrecord

Algorithms 4 and 5 are executed for each record in the MIDT file. This results in a set with all valid O&Ds for this paper with the associated capacities for each leg in the O&D.

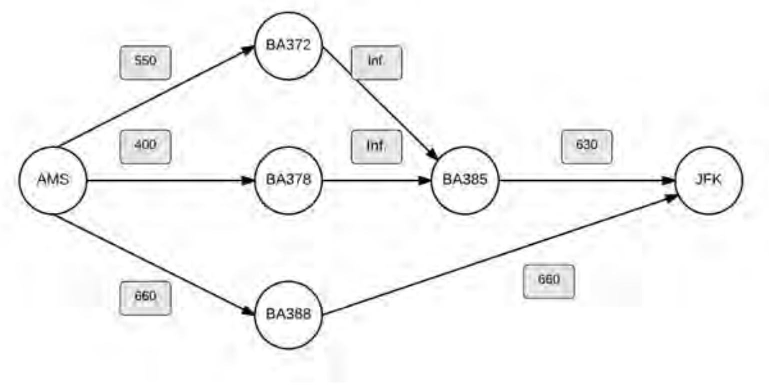
3.2.3 Calculation of O&D capacity

The goal of this paper is to determine the complete capacity between an O&D. Therefore, the next step is the calculation of the total capacity between an O&D. This is done with the Ford-Fulkerson Algorithm described in Section 2.4. First, we have to check on which days the O&D can be flown. We need to check this, since not all flights in an O&D are flown each day. Therefore, a vector with all possible weekdays of the O&D is needed. This is done by combining the capacity indices and the day indicator of the datrecord as determined in the previous section. We check for each day whether all flights in the O&D are flown or not. This check results in three vectors, one vector for each class.

Next, the network flows for all three classes per weekday are made. The network flows are saved in an adjacency list (see 2.3.6). All flights in the O&D are added to the adjacent list together with their capacity. Figure 3.2 shows how the network flow is being built in this research. The middle nodes are the flight numbers and the source and sink are the origin and destination. The original idea was to make a network flow with stations as nodes (Figure 3.2a). However, it seemed that the Ford-Fulkerson algorithm was not able to give an accurate result when there are multiple edges between two nodes. Therefore, we decided to use the flight numbers as nodes and add extra edges between the flight numbers with an infinite capacity (Figure 3.2b). The added edges between the flight numbers will not affect the outcome of the Ford Fulkerson algorithm, since the capacity of these edges is higher than the rest of the capacities in the graph.



(a) Stations as nodes



(b) Flight numbers as nodes

Figure 3.2: Network flow origin and destination in two different manners

Algorithm 1 is used to determine the maximum capacity for the adjacency list. The maximum capacity is found when there is no augmenting path possible from the source to the sink. To determine whether there is an augmenting path we used breadth-first search (Algorithm 2). For all 21 network flows (one for each class per weekday) the capacity is calculated with this algorithm. Finally, this resulted in the total capacity per carrier between all ten O&Ds.

Algorithm 1 Ford Fulkerson

```
function FordFulkerson(source, sink)
    u and v    ▷ Create a residual graph and fill it with the given capacities in the original graph
    $residualGraph[V][V]$    ▷ $residualGraph[i][j]$ indicates the residual capacity of the edge i,j. The residual capacity is 0 if these edge does not exist
    for u is 0 to size of list do
        for v is 0 to size of list do
            $residualGraph[u][v] \leftarrow 0$
            for all edges in the list do
                if $keyVertex$ is v then
                    $residualGraph[u][v] \leftarrow weight$
                end if
            end for
        end for
    end for
    parent[V]    ▷ This array is filled by BFS and stores the path
    $maxFlow \leftarrow 0$    ▷ There is no flow initially
    while $Bfs(residualGraph, source, sink, parent)$ do    ▷ While there is a path from the source to the sink, augment the flow
        $pathFlow \leftarrow$ MAXINT
            ▷ Find the maximum flow through the path
        for v is sink; v is not source; v ← parent[v] do
            $u \leftarrow parent[v]$
            pathFlow ← min($pathFlow, residualGraph[u][v]$)
        end for
        for v is sink; v is not source; v ← parent[v] do▷ Update the residual capacities of the edges and the reverse edges along the path
            $u \leftarrow parent[v]$
            $residualGraph[u][v] \leftarrow residualGraph - pathFlow$
            $residualGraph[v][u] \leftarrow residualGraph + pathFlow$
        end for
        $maxFlow \leftarrow pathFlow$    ▷ Add path flow to overall flow
    end while
    Return $maxFlow$
end function
```

Algorithm 2 Breadth-first search

```
function BFS(graph[V,V], source, sink, parent[])
    visited[V]                ▷ Create a visited array and mark all vertices as not visited
    for all v in V do
        $v \leftarrow false$
    end for
            ▷ Create a queue, enqueue source vertex and mark source vertex as visited
    queue q
    q.push(source)
    $visited[source] \leftarrow true$
    $parent[source] \leftarrow -1$
    while q is not empty do                                         ▷ Standard BFS loop
        $u \leftarrow q.front$
        $q.pop$
        for v is 0 to size of list do
            if visited[v] is false and $graph[u][v] > 0$ then
                $q.push(v)$
                $parent[v] \leftarrow u$
                $visited[v] \leftarrow true$
            end if
        end for
    end while
            ▷ If the sink is reached starting from the source, return true and else false
    if visited[sink] is true then
     return true
    else
     return false
    end if
end function
```

4 Results

The result of this research is a method to determine the planned capacity between an O&D. This method is described in Section 3. In this section, some results of the previous described algorithm are given. Besides the capacity between an O&D, we determined the bottleneck flights in an O&D. The bottleneck flight is the flight which restricts the total capacity between an O&D. In terms of the network flow theorem, the bottleneck flights are the arcs crossing the minimal cut of the network flow (Section 2.3.2).

4.1 O&D capacity

This section gives the calculated O&D capacity with the algorithms of Section 3 to show the correctness of the method. Since we cannot show all used O&Ds, we decided to show the outcomes of three O&Ds. One direct flight: AMS-JFK with KLM, one simple flight: AMS-JFK with Lufthansa (LH) and one complex flight: JFK-LAS operated by American Airlines (AA). The total capacity calculated with the algorithm for all ten O&Ds are given in the table of Appendix 7.9.

4.1.1 AMS-JFK operated by KLM

The O&D AMS-JFK flown by KLM is a direct flight and the capacity is therefore easy to calculate. Table 5 gives the possible flights between Amsterdam and New York (John F. Kennedy airport) with KLM.

Origin	**Destination**	**Flightpath**	**Business (m-t-w-t-f-s-s)**	**Economy (m-t-w-t-f-s-s)**
AMS	JFK	KL639	120-120-120-150-30-150-0	884-884-925-1105-221-1105-0
AMS	JFK	KL643	168-168-168-186-210-210-168	952-952-952-1600-1190-1190-1544
AMS	JFK	KL641	140-156-168-175-175-175-140	1168-1379-1544-1460-1950-1460-1168

Table 5: Routes between AMS-JFK with KLM

The network flow associated with this table is easy to create, since the routes are all direct flights. Figure 4.1 shows the network flow for this route on Monday. The total capacity in this case is the sum over the capacities of the three flights. Therefore, the outcome of the algorithm should be the sum of the values in the columns First, Business and Economy in Table 5. The total capacity is therefore 26,730. The outcome of the algorithm is also 26,730 (see Appendix 7.9).

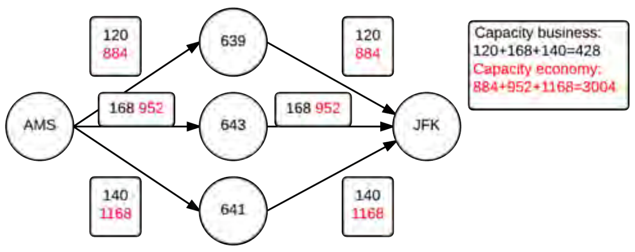

AMS
639
643
641
JFK
120 884
168 952
140 1168
120 884
168 952
140 1168
Capacity business: 120+168+140=428
Capacity economy: 884+952+1168=3004

Figure 4.1: AMS-JFK with KLM on Monday

4.1.2 AMS-JFK operated by LH

The O&D AMS-JFK operated by LH is an indirect flight with viapoints in Munich (MUC) and Frankfurt (FRA). All possible routes between Amsterdam and New York with Lufthansa are given in Table 6.

Origin	Destination	Flightpath	Day indicator	Viapoint
AMS	JFK	LH991 LH404	0 0	FRA
AMS	JFK	LH993 LH404	0 0	FRA
AMS	JFK	LH1003 LH400	0 0	FRA
AMS	JFK	LH2301 LH410	0 0	MUC
AMS	JFK	LH989 LH404	0 0	FRA

Table 6: Routes between AMS-JFK operated by LH

We made a flow network for each day to check whether the algorithm returns the correct capacity. Figure 4.2 gives the flow network for monday. The number of seats in the first class is zero for this O&D. It is possible to distinguish three different paths in the figure, we colored each path with a different color. The calculation of the capacity is given in the figure. We calculated the total capacity for each day, which resulted in a total number of seats for the business class of 3936 and a total number of seats for the economy class of 14,484. This is the same capacity as we calculated with the algorithm given in this paper (see Appendix 7.9).

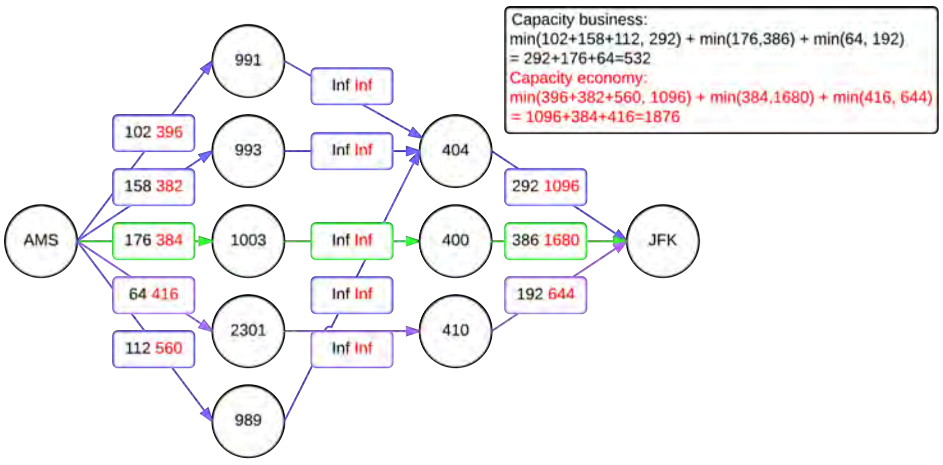


Figure 4.2: AMS-JFK with LH on Monday

4.1.3 JFK-LAS operated by AA

The O&Ds in the previous sections were simple examples where it is possible to calculate the total capacity by hand. However, there are a lot of O&Ds which are too complicated to calculate the capacity without the algorithm. An example of such an O&D is JFK-LAS operated by AA. All possible routes are given in the network flow of Figure 4.3. As one can see, there are a lot of possible routes between New York and Las Vegas. It is therefore (almost) impossible to calculate the total capacity by hand and it will cost a lot of calculation time to calculate it by hand. The outcome of the algorithm for this O&D is a total of 46,798 seats divided over 5,304 seats in the first class and 41,494 economy seats (see Appendix 7.9).

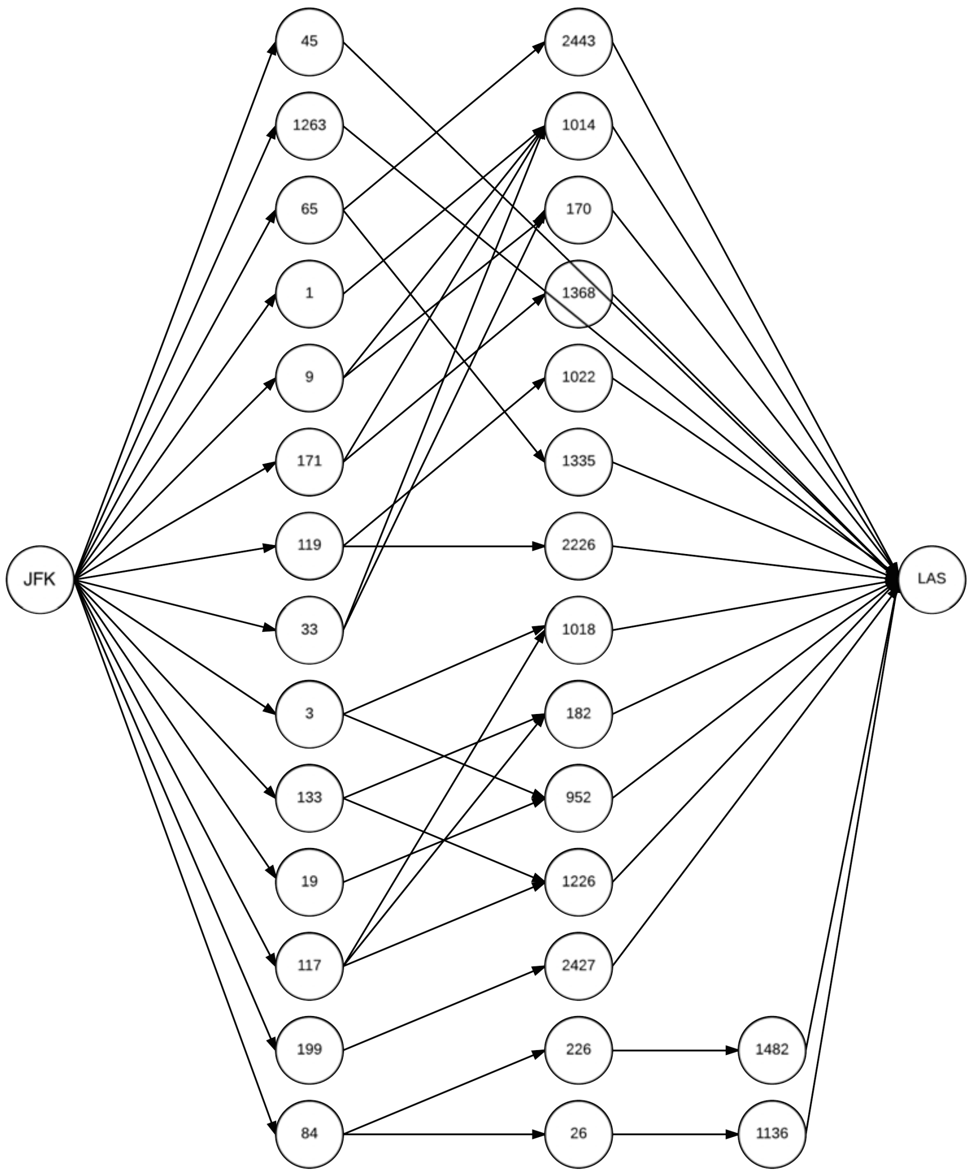


Figure 4.3: Flow network of JFK-LAS operated by AA

4.2 Bottleneck flights

Another outcome of the algorithm is the determination of the bottleneck flights in an O&D. The bottleneck flights in an O&D are the arcs crossing the minimal cut of the network flow. We will give the bottleneck flights of two simple O&D's. The first O&D is AMS-JFK operated by LH and the other simple O&D is AMS-JFK operated by OS. We have chosen to show only two simple O&Ds, since a complex O&D has a lot of bottleneck flights and it will get too complicated to show them all.

4.2.1 AMS-JFK operated by LH

Figure 4.2 shows the network flow of the O&D on Monday. We distinguished three paths in the network flow with three different colors. Each path has their own bottleneck flight, so the algorithm will give us three bottleneck flights for this specific O&D. The bottleneck flights in this example are the following three flights:

- LH1003
- LH2301
- LH404

When the capacity of these three flights increases, the total capacity of this O&D will also increase. Flight LH1003 should increase the capacity with 210 (386-176) first seats and 1296 (1680-384) economy seats to use all available capacity in this path. Flights LH2301 and LH404 should increase their business/economy capacity with 128/228 seats and 70/242 seats, respectively.

4.2.2 AMS-JFK operated by OS

Figure 4.4 gives the flow network of the O&D AMS-JFK operated by OS on Monday. The total capacity can be found in Appendix 7.9. The algorithm gave as bottleneck flights the flights OS372 and OS378. This outcome is easy to validate, since the capacity of these flights together is 550+400 = 950, while the capacity of the flight OS87 is 1040, which is more than the other flights together.

In terms of the network flow theorem, the bottleneck flights are the arcs crossing the minimal cut of the network flow (Section 2.3.2). The dotted line in Figure 4.4 is the minimul cut in the network flow and the arcs crossing them are therefore the bottleneck flights. These are indeed the flights OS372 and OS378.

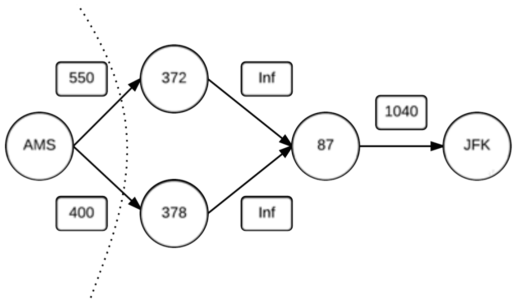


Figure 4.4: AMS-JFK operated by OS

4.3 Calculation time

For the usefulness of the algorithm it is important that the calculation time of the algorithm is low. We measured the calculation time of the algorithm for the ten O&Ds we used in this paper. The calculation time is the time it takes for the algorithm to calculate the capacities, so the processing time of the two datasources is excluded in the calculation time. In other words, it is the processing time of the algorithms described in Section 3.2.3. The calculation time is less than 3 seconds for each of the ten O&Ds. However, the calculation time will be higher when you process more O&Ds.

5 Discussion

5.1 Conclusion

The main question answered in this research was how the planned capacity between an O&D, based on capacity (OAG) and booking (MIDT) data, can be determined. Since OAG data is based on operating carrier information and MIDT data on marketing carrier information, the OAG data was converted into marketing carrier based data. This is done by creating a set indexed on marketing carrier flight number. Next, the marketing flight numbers in the flightpath of the MIDT data are converted in operational flight numbers using the set created in the processing of the OAG data. All routes between an O&D are saved as a network flow in the form of an adjacency list. The nodes of the network are the flight numbers in the O&D. Finally, we implemented the Ford-Fulkerson algorithm to determine the maximum capacity of the O&D. This resulted in the total capacity together with the associated bottleneck flights for the O&Ds. The calculation time of the algorithm is less than three seconds for each of the ten O&Ds used in this paper. In short, the result of this study supported the hypothesis that the Ford-Fulkerson algorithm is a good method to calculate the planned capacity between an O&D. This algorithm was tested on ten O&Ds for a test period of one month. The bottleneck flights were also detected in the algorithm.

This research makes it possible for KLM to convert the leg based OAG data into O&D based data. This conversion makes it possible to use Skyline together with other tools and to compare different tools with Skyline. The focus of this research was on calculating the planned capacity between an O&D. However, another output of this research is the bottleneck flights of the O&D. When the bottleneck flights are detected, KLM can react on this by increasing the number of seats for the bottleneck flight. This will increase the total capacity for the O&D.

5.2 Future work

The research described in this paper includes a small part of the possibilities to calculate the capacity between an O&D. This section will discuss some opportunities for future work to extend this research.

This study provides a method to calculate the total O&D capacity for operational carriers. A challenging task for further research is the specification of the capacity for both marketing as operational carriers. In this report only routes with flights operated by the same operational carrier are included. However, routes with flights operated by a different operational carrier, but the same marketing carrier are also interesting to investigate. Additional research is therefore needed to create a method to calculate the capacity also for marketing carriers. During this research it seemed that the available marketing data were incomplete. For example, one marketing flight number can be flown with different operational flight numbers at the same time in the OAG data. This is of course never possible. It is therefore necessary that there is more accurate marketing data available when the research continues.

Another promising line of research would be to involve the number of bookings and the QSI by calculating the capacity. The method provided in this report only uses the total capacity of the flights. However, it would give a more accurate view when the bookings and/or the QSI are included in the method.

Lastly, the MIDT data used in this research is monthly data, but the OAG data is at day level. The method would be more useful when the daily capacity can be calculated. Another thing to keep in mind is that information about low cost carriers is not included in the MIDT data. The complete capacity between an O&D is therefore in reality higher than the result of the algorithm provided in this report.

6 Bibliography

[1] Peter Belobaba, Amedeo Odoni, and Cynthia Barnhart. *The global airline industry*, volume 23. John Wiley & Sons, 2009.

[2] Robert Cross. *Revenue management*. Universities Press, 1997.

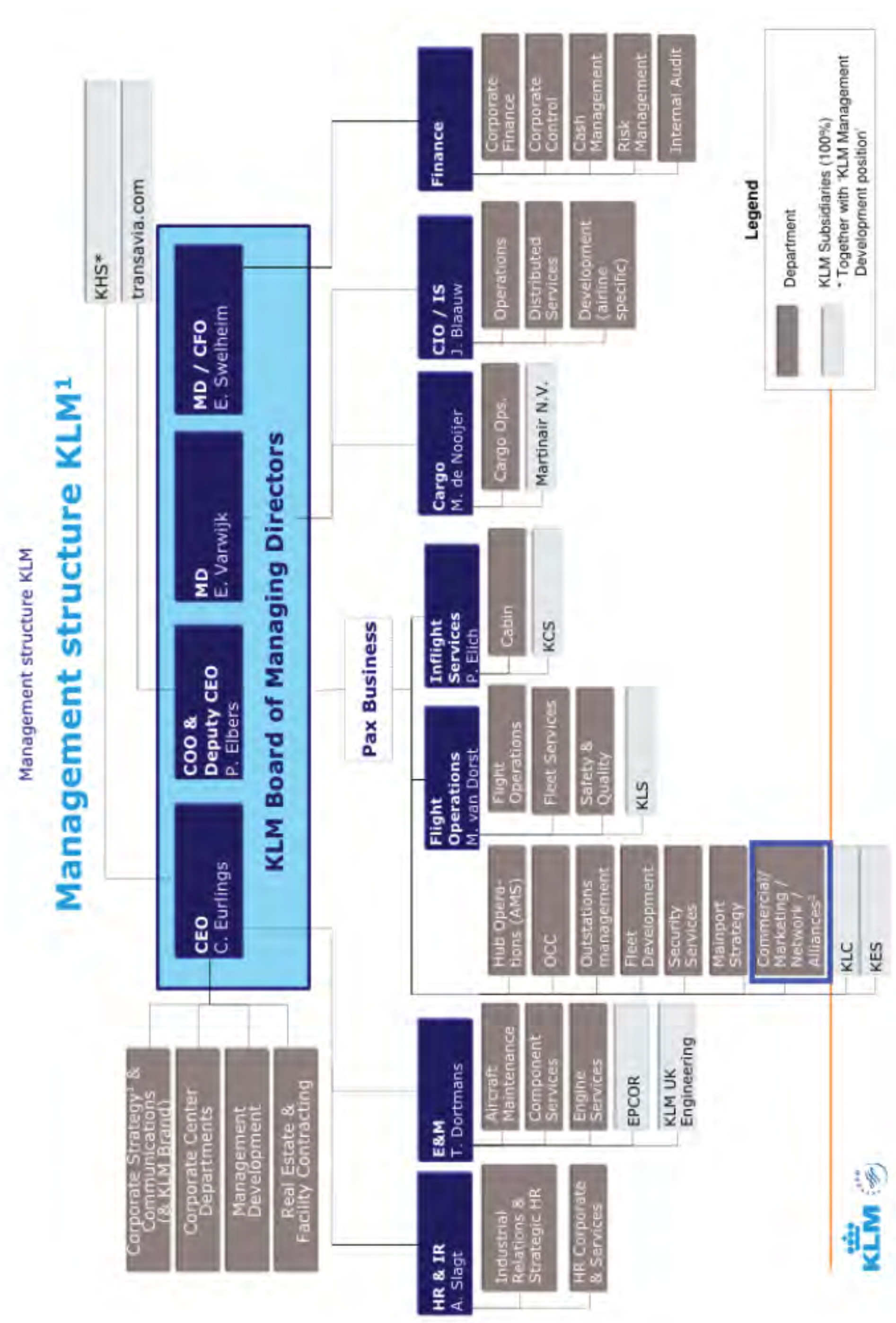
[3] Michael T Goodrich and Roberto Tamassia. *Algorithm Design: Foundation, Analysis and Internet Examples*. John Wiley & Sons, 2006.

[4] IATA. http://www.iata.org/whatwedo/passenger/pages/index.aspx.

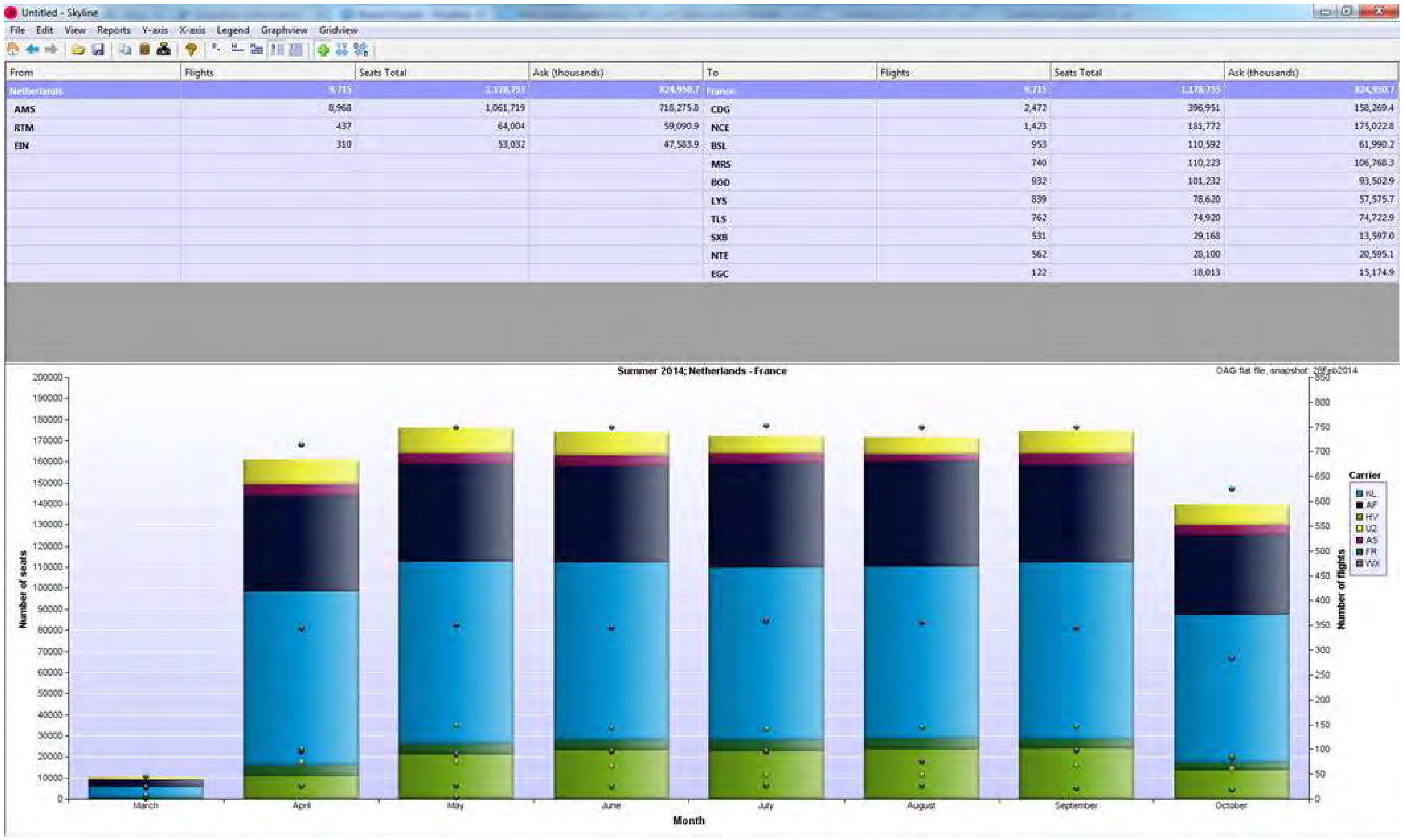
[5] Fred Roberts and Barry Tesman. *Applied combinatorics*. CRC Press, 2011.

7 Appendix

7.1 Organogram KLM



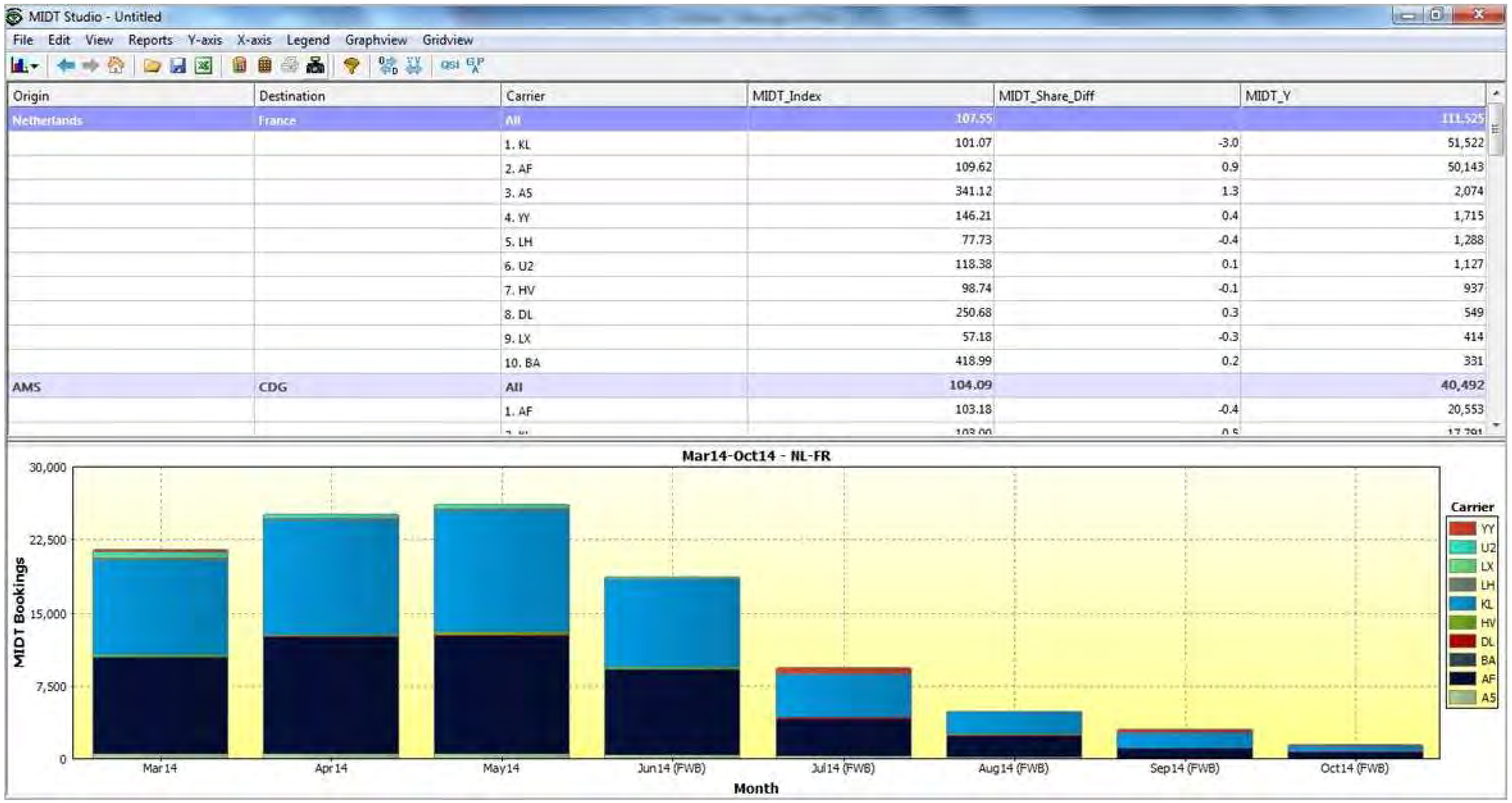
7.2 Screenshot Skyline



Untitled - Skyline
File Edit View Reports Y-axis X-axis Legend Graphview Gridview
Summer 2014; Netherlands - France
OAG flat file, snapshot: 28Feb2014
Number of seats
Number of flights
Month
March
April
May
June
July
August
September
October
Carrier
KL
AF
HV
U2
AS
FR
VX

From	Flights	Seats Total	Ask (thousands)	To	Flights	Seats Total	Ask (thousands)
Netherlands	9,715	1,178,755	824,950.7	France	9,715	1,178,755	824,950.7
AMS	8,968	1,061,719	718,275.8	CDG	2,472	396,951	158,269.4
RTM	437	64,004	59,090.9	NCE	1,423	181,772	175,022.8
EIN	310	53,032	47,583.9	BSL	953	110,592	61,990.2
				MRS	740	110,223	106,768.3
				BOD	932	101,232	93,502.9
				LYS	839	78,620	57,575.7
				TLS	762	74,920	74,722.9
				SXB	531	29,168	13,597.0
				NTE	562	28,100	20,595.1
				EGC	122	18,013	15,174.9

7.3 Screenshot MIDT studio



Origin	Destination	Carrier	MIDT_Index	MIDT_Share_Diff	MIDT_Y
Netherlands	France	All	107.55		111,525
		1. KL	101.07	-3.0	51,522
		2. AF	109.62	0.9	50,143
		3. A5	341.12	1.3	2,074
		4. YY	146.21	0.4	1,715
		5. LH	77.73	-0.4	1,288
		6. U2	118.38	0.1	1,127
		7. HV	98.74	-0.1	937
		8. DL	250.68	0.3	549
		9. LX	57.18	-0.3	414
		10. BA	418.99	0.2	331
AMS	CDG	All	104.09		40,492
		1. AF	103.18	-0.4	20,553

7.4 Pseudo-code description of the Ford-Fulkerson algorithm

Algorithm 3 The Ford-Fulkerson algorithm

```
function MaxFlowFordFulkerson(N)
    Input: Flow network $N = (G, c, s, t)$
    Output: A maximum flow $f$ for $N$
    for all Edges $e \epsilon N$ do
        $f(e) \leftarrow 0$
    end for
    $stop \leftarrow false$
    while $stop$ is false do
        Traverse G starting at $s$ to find an augmenting pathe for $f$
        if Augmenting path $\pi$ exists then
            Compute the residual capacity $\triangle_f(\pi)$ of $\pi$
            $\triangle \leftarrow +\infty$
            for all Edges $e \epsilon \pi$ do
                if $\triangle_f(e) < \triangle$ then
                    $\triangle \leftarrow \triangle_f(e)$
                end if
            end for
            Push $\triangle = \triangle_f(\pi)$ units of the flow along path $\pi$
            for all Edges $e \epsilon \pi$ do
                if $e$ is a forward edge then
                    $f(e) \leftarrow f(e) + \triangle$
                else
                    $f(e) \leftarrow f(e) - \triangle$
                                                            ▷ $e$ is a backward edge
                end if
            end for
        else
            $Stop \leftarrow true$                          ▷ $f$ is a maximum flow
        end if
    end while
end function
```

7.5 Pseudo-code to convert marketing flight numbers into operational flight numbers

Algorithm 4 Determine all possible flightpaths

```
Input: A vector $flightpath$ storing $n \geq 1$ longs, a vector $viapoints$ storing $m \geq 0$ shorts, an origin and destination
Output: A vector with all possible flightpaths for the oagrecord
$last_from \leftarrow origin$
for $i \leftarrow 0$ to $n$ do
    $fltdatrecord\ oagrec$
    $oagrec.key\_mcarflightnr \leftarrow flightpath[i]$
    $oagrec.key\_from \leftarrow last\_from$
    if $m \leq i$ then
        $oagrec.key\_to \leftarrow destination$
    else
        $oagrec.key\_to \leftarrow viapoints[i]$
    end if
    if $oagrec$ is in $oag\_data.fltdatset$ then
        $ocarfltnrs \leftarrow dat\_opcarset$
    else
        return
    end if
    Introduce a vector $tempvec$ storing $k \geq 0$ vectors of $flightpathrecords$
    Introduce a vector $possiblefltpaths$ storing $l \geq 0$ vectors of $flightpathrecords$
    for $f \leftarrow 0$ to size of $ocarfltnrs$ do
        Introduce a vector $vec$ storing $p \geq 0$ $flightpathrecords$
        Introduce a vector $oldvec$ storing $q \geq 0$ $flightpathrecords$
        $fltpathrec.key\_mcarfltnr \leftarrow oagrec.key\_mcarflightnr$
        $fltpathrec.key\_opcarfltnr \leftarrow oagrec.key\_opcarfltnr$
        $fltpathrec.key\_from \leftarrow oagrec.key\_from$
        $fltpathrec.key\_to \leftarrow oagrec.key\_to$
        $fltpathrec.dat\_cap\_first\_array \leftarrow ocarfltnrs[f].dat\_cap\_first\_array$
        $fltpathrec.dat\_cap\_business\_array \leftarrow ocarfltnrs[f].dat\_cap\_business\_array$
        $fltpathrec.dat\_cap\_economy\_array \leftarrow ocarfltnrs[f].dat\_cap\_economy\_array$
        $fltpathrec.dat\_cap\_total\_array \leftarrow ocarfltnrs[f].dat\_cap\_total\_array$
        $fltpathrec.dat\_day\_indicator \leftarrow ocarfltnrs[f].dat\_day\_indicator$
        if $i = 0$ then
            $vec[p + 1] \leftarrow fltpathrec$
            $tempvec[k + 1] \leftarrow vec$
        else
            for $j \leftarrow 0$ to $l$ do
                $oldvec \leftarrow possiblefltpaths[j]$
                $oldvec[q + 1] \leftarrow fltpathrec$
                $tempvec[k + 1] \leftarrow oldvec$
            end for
        end if
    end for
    $possiblefltpaths \leftarrow tempvec$
    $last\_from \leftarrow oagrec.key\_to$
end for
Return $possiblefltpaths$
```

7.6 Pseudo-code to determine whether the flight path is valid

Algorithm 5 Create valid flightpaths

```
Input: A vector possible_flightpaths storing $n \geq 1$ vectors of fltpathrecords storing
$m \geq 1$ fltpathrecords
Output: A set midtset storing u datrecords
for i is 0 to n do
    fltpathrecord vec $\leftarrow possible\_flightpaths[i]$
    datrecord datrec
    $origin \leftarrow vec[0].key\_opcarrier$
    $destination \leftarrow vec[0].key\_to$
    $opcarrier \leftarrow vec[0].key\_opcarrier$
    a vector viapoints storing $p \geq 0$ shorts
    a vector fltpath storing $q \geq 0$ shorts
    a vector $cap\_day$ storing $s \geq 0$ shorts
    a vector $day\_indicators$ storing $t \geq 0$ shorts
    for j is 0 to m do
        if $vec[j].key\_opcarrier$ is $opcarrier$ then
            if $j$ is not 0 then
                $viapoints[p] \leftarrow destination$
                $destination \leftarrow vec[j].key\_to$
            end if
            $fltpath[q] \leftarrow vec[j].key\_opcarfltnr$
            $cap\_day \leftarrow vec[j].dat\_cap\_array$
            $day\_indicators \leftarrow vec[j].dat\_day\_indicator$
        else
            return
        end if
    end for
    Fill datrec record
    if datrec is not in midtset then
        Insert the datrec in midtset
    end if
end for
```

7.7 OAG data

OAG Field Name	**Field Description**
Carrier1	Carrier code.
Carrier1Name	Carrier1 name.
AllianceName	Alliance Carrier1 belongs To.
FlightNo1	Four figure flight number. Right justified.
CarrDom1	Domicile country code of the carrier 1.
CarrDom1Name	Domicile country name of the carrier 1.
DepAirport	Departure airport code.
DepAirportName	Departure airport name.
DepTerminal	Terminal of departure - terminal code.
DepCity	Departure city code.
DepCityName	Departure city name.
DepState	Departure state code – unique by country only.
DepStateName	Departure state name.
DepIATACtry	Departure IATA country code.
DepIATACtryName	Departure IATA country name.
DepDOTCtry	Departure US DoT country code.
DepDOTCtryName	Departure US DoT country name.
DepReg	Region code based on IATA forecasting regions.
DepRegName	Region name based on IATA forecasting regions.
ArrAirport	Arrival airport code.
ArrAirportName	Arrival airport name.
ArrTerminal	Terminal of arrival - terminal code.
ArrCity	Arrival city code.
ArrCityName	Arrival city name.
ArrState	State code - unique by country only.
ArrStateName	Arrival state name.
ArrIATACtry	IATA country code.
ArrIATACtryName	IATA country name.
ArrDOTCtry	Arrival US DoT country code.
ArrDOTCtryName	Arrival US Dot country name.
ArrReg	Arrival region code based on IATA forecasting regions.
ArrRegName	Arrival region name based on IATA forecasting regions.
LocalDepTime	Local departure time - 24 hour.
LocalArrTime	Local arrival time – 24 hour.
LocalArrDay	Marker to show if local arrival day different to local departure day.
LocalDaysOfOp	Local days of operation at departure airport.
ArrDaysOfOp	Local days of operation at arrival airport.
Service	Type of service of departure aircraft.

Seats	Available seat capacity (total) on departure aircraft.
FstSeats	Available seats capacity in first class cabin.
BusSeats	Available seats capacity in business class cabin.
EcoSeats	Available seats capacity in economy/coach class cabin.
EffFrom	Local effective from date of departure.
EffTo	Local effective to date of departure.
LocalDaysOfOp1	Local days of operation at departure airport (Monday).
LocalDaysOfOp2	Local days of operation at departure airport (Tuesday).
LocalDaysOfOp3	Local days of operation at departure airport (Wednesday).
LocalDaysOfOp4	Local days of operation at departure airport (Thursday).
LocalDaysOfOp5	Local days of operation at departure airport (Friday).
LocalDaysOfOp6	Local days of operation at departure airport (Saturday).
LocalDaysOfOp7	Local days of operation at departure airport (Sunday).
ElapsedTime	Total time value for flight including time on ground in multi leg flights in hours and minutes.
FlyingTime	Time value for flight, for stopping flights total of block times.
GroundTime	Time value for flight, for stopping flights total of time spent on ground at intermediate airports.
Stops	Number of en-route stops.
IntAirports	Airport codes of the en-route airports for stopping flights.
IntCities	City codes of the en-route cities for stopping flights.
IntCountries	Country codes of the en-route countries for stopping flights.
AcftChange	Marker to show aircraft change en route.
AcftChApt1	1st Airport code where there is change of aircraft en-route.
AcftChApt2	2nd Airport code where there is change of aircraft en-route.
AcftChApt3	3rd Airport code where there is change of aircraft en-route.
GeneralAcft	General aircraft type for departure aircraft.
GeneralAcftName	General aircraft type name for departure aircraft.
SpecificAcft	Specific aircraft type for departure aircraft.
SpecificAcftName	Specific aircraft type name for departure aircraft.
SecondAcft	Aircraft type at first aircraft change airport.
ThirdAcft	Aircraft type at second aircraft change airport.
FourthAcft	Aircraft type at third aircraft change airport.
Freightons	Available typical freight capacity in tonnes (metric).
PassClass	Passenger classes available on flight.
FreightClass	Cargo classes on flight.
Routing	Complete full routing of flight up to a maximum of 15 airports.
Statmiles	Great Circle Distance in Statute Miles.
Nautmiles	Great Circle Distance in Nautical Miles.

Km	Great Circle Distance in Kilometres.
DistStMiles	Great Circle Distance in Statute Miles (Direct).
DistNtMiles	Great Circle Distance in Nautical Miles (Direct).
DistKm	Great Circle Distance in Kilometres (Direct).
Restrictions	Restriction on flight for passenger, cargo or mail.
ShAirlDes	The carrier code of the operating carrier where provided by the non-operating carrier.
MultCDes	The Carrier code of a number of carriers which jointly operate the flight.
DupMarker	Duplication marker resulting from OAG data processing.
DupCar1	Every carrier and flight number that duplicates with the flight.
DupCar2	As DupCar1 above.
DupCar3	As DupCar1 above.
DupCar4	As DupCar1 above.
DupCar5	As DupCar1 above.
DupCar6	As DupCar1 above.
DupCar7	As DupCar1 above.
DupCar8	As DupCar1 above.
OpCar	A marker to show if the carrier is operating or non-operating.
Comment	Comment codes.
AcftOwnerCode	Carrier that owns the aircraft.
AcftOwnerCodeName	Name of carrier that owns the aircraft.
CockpitCrewCode	Carrier responsible for cockpit crew.
CockpitCrewCodeName	Name of carrier responsible for cockpit crew.
CabinCrewCode	Carrier responsible for cabin crew.
CabinCrewCodeName	Name of carrier responsible for cabin crew.
LongLeg	Marker to show multi-leg flight with all sectors – the full flight description.
MaxTakeOffWeight	Maximum Take off Weight - typical values used in tonnes (metric).
HoldVolume	The typical hold volume of the aircraft baggage and cargo space in cubic metres.
RangeKm	The typical range over which the aircraft can fly in Kilometres.
RangeStatMiles	The typical range over which the aircraft can fly in Statute Miles.
RangeNautMiles	The typical range over which the aircraft can fly in Nautical Miles.
CruiseSpeed	The typical speed at which the aircraft cruises in Statute Miles per Hour.
Category	Category of aircraft.
Manufacturer	The Manufacturer code of the aircraft type.

Ghost	A marker to show non-operating ghost/funnel flights.
SubGovnApp	Flight is subject to government approval.
Flt Dup	Duplication marker provided by the carrier.
Frequency*	The number of flights occurring in a specific time period.
Available Seat Miles*	The total number of available seats multiplied by the distance in miles flown.
Available Seat Kilomteres*	The total number of available seats multiplied by the distance in kilometres flown.

7.8 MIDT data

Field name	**Description**	**Length**	**Explanation**
Origin	Origin airport	char(3)	Three character airport code.
Dest	Destination airport	char(3)	Three character airport code.
Flight	Dominant flight	char(1)	One digit indicating the position of the dominant flight in the flight path.
Month	Month	char(4)	Two digit month and two digit year (1205 means december'05).
GDS	GDS	char(2)	Two character GDS code.
Agent	Point of Sale agent	char(7)	Seven digit agent number with leading zeros.
Pos	Point of Sale country	char(2)	Two character country code.
Class	Dominant subclass	char(1)	One character booking class code.
Apd	Advance purchase date	char(1)	One digit indicating the advance purchase bucket.
Bookings	Number of bookings	varying	Number of bookings.
Fltpath	Flight path	varying	List of two character airline and 4 character flightnumber combinations containing all flights in the O&D (flightnumber with leading zeros).
Via	Via points	varying	List of three character airport codes containing only the via points (O&D are not in this list).

7.9 Planned O&D capacity

Origin	Destination	Carrier	Capacity total	Capacity first	Capacity business	Capacity economy
AAL	**LHR**	**All**	**31497**	**0**	**3767**	**27232**
		SK	23537	0	1977	21062
		KL	7960	0	1790	6170
AMS	**BOG**	**All**	**33391**	**150**	**3262**	**29269**
		UA	8897	150	496	8029
		AF	7975	0	870	7105
		LH	8831	0	1416	7191
		DL	7688	0	480	6944
AMS	**JFK**	**All**	**187396**	**90**	**22558**	**160133**
		BA	30087	0	6385	23702
		UA	1550	0	0	1550
		AF	14530	0	2057	12473
		LH	18772	0	3936	14484
		EI	19964	0	0	18476
		FI	7083	90	390	6593
		TK	12502	0	1204	11158
		DL	20028	0	1404	17363
		KL	26730	0	3097	23633
		LO	3966	0	0	3966
		OS	7668	0	0	7634
		SU	4370	0	628	3742
		LX	8404	0	2790	4440
		UX	2340	0	156	2184
		AY	5769	0	47	5722
		AZ	3633	0	464	3013
AMS	**LAS**	**All**	**96034**	**610**	**2494**	**82437**
		BA	9264	0	1788	7250
		UA	24154	610	16	20043
		DL	60152	0	690	52848
		US	2464	0	0	2296
AMS	**LAX**	**All**	**189053**	**684**	**19428**	**159825**
		BA	26278	0	5181	20335
		UA	10272	126	64	8588
		AF	5759	0	712	5008
		LH	19490	0	3762	15648
		EK	15996	434	2356	13206
		TK	10447	0	868	9497
		DL	59675	0	1896	52460
		KL	10604	0	1302	9302

		US	5456	0	0	5084
		SU	3810	0	548	3262
		LX	6789	0	1457	4440
		DY	4092	0	0	4092
		AZ	3627	0	538	3013
		CZ	6758	124	744	5890
AMS	**NBO**	**All**	**60968**	**248**	**7818**	**52236**
		BA	9279	0	1826	7129
		EK	10886	248	1302	9336
		TK	4709	0	524	4185
		KL	13268	0	1302	11966
		KQ	9982	0	868	9114
		MS	3312	0	398	2760
		LX	5192	0	990	4026
		EY	4340	0	608	3720
FRA	**LAX**	**All**	**174916**	**1848**	**16383**	**146250**
		AA	7657	496	0	6014
		BA	13684	0	2925	10440
		UA	30833	1024	0	25755
		IB	1950	0	0	1950
		AC	11758	0	1212	10546
		AF	8421	0	1269	7140
		LH	25205	328	5048	19581
		TK	10447	0	868	9579
		DL	20645	0	1014	17271
		KL	3100	0	620	2480
		US	28584	0	676	26244
		SU	6340	0	918	5328
		LX	4216	0	1457	2232
		AZ	2076	0	376	1690
OSL	**BCN**	**All**	**91221**	**0**	**11699**	**79500**
		AB	4104	0	0	4104
		AF	8654	0	544	8108
		LH	26296	0	2844	23452
		SK	7661	0	688	6966
		KL	16027	0	4997	11030
		BT	1035	0	81	954
		SU	4320	0	620	3688
		LX	3572	0	1892	1680
		AY	3652	0	33	3618
		DY	13020	0	0	13020
		VY	2880	0	0	2880

JFK LAS	**All**	**227822**	**13044**	**5908**	**208870**
	B6	27850	0	0	27850
	AA	46798	5304	0	41494
	BA	8768	420	1794	6554
	UA	16777	276	462	16039
	DL	76197	4664	2496	69037
	US	20982	1820	0	19162
	VX	26628	488	952	25188
	SY	3822	72	204	3546