Erasmus University Rotterdam

Erasmus School of Economics

Thesis MSc Econometrics & Management Science
Operations Research & Quantitative Logistics

Structurally improving well-being of pilots: the airline crew pairing problem re-solved

Authors
Marieta Batelaan (434899)
Date
May 1st, 2023

Supervisor
dr. Twan Dollevoet
Second assessor
dr. Wilco van den Heuvel

Abstract

This thesis describes a new approach to transforming an airline's flight schedule into a schedule of multiple-day shifts for its crew, also known as pairings. This problem is called the airline crew pairing problem. This study is focused on airline pilots specifically and attempts to create a method that improves their work satisfaction and well-being without causing excessive additional costs for airlines. To do this while also decreasing the computation time of the problems, flight services are grouped into clusters using a special model balancing several aspects, such as the number of flights in each cluster as well as the number of countries and airports visited. This ensures every pilot can have a varied schedule.

Data is provided by Dutch airline KLM and covers their full schedule for the Summer of 2018, and the main model is a set partition model, covering each flight service by exactly one pairing. For the airlines' intercontinental services, the model can be solved directly after enumerating all possible pairings. European services require column generation using a shortest-path algorithm to find an optimal non-integer solution and a truncated branch-and-price algorithm to find a final solution. This approach can be used on both clustered and full European instances and the results show clustering can be beneficial to the size of problem instances and to computation time.

Contents

Chapter 1

Introduction

Even though being an airline pilot is considered a dream job by many, the reality is that it has its drawbacks and limitations. Due to the irregular and unpredictable work schedules, it can be challenging to combine the career and interactions with family and friends at home. The schedules can also make it extremely difficult to build relationships with coworkers since all pilots and co-pilots are scheduled individually. It is likely that every shift, which can last up to a week, starts with new introductions. Frequently working with new coworkers can diminish the well-being of crew members and take away the chance of getting attuned to each other, which can be vital in cases of emergency.

Grouping the pilots into clusters during scheduling can diminish these problems. Because of the clusters, pilots get a better chance to get to know their co-workers, swapping shifts becomes easier, and cases of illness are less problematic. This could improve the pilots' well-being, potentially benefiting the company and its passengers in the long run. The grouping cuts the problem into smaller subproblems, reducing computational difficulties found in methods that do not group crew members. It could, however, limit efficiency and increase costs due to the reduced flexibility.

To analyse the effects of clustering, the cockpit crew pairing problem is solved for various instances. The problem can be defined as the generating of shifts for pilots, where each shift consists of several days work.

For airlines, the salaries of pilots and all other costs involved with taking care of pilots while on a destination, such as hotel and meal costs, make up a crucial part of their expenses and can heavily impact the annual profit. This is why the main research question is as follows:
How can pilots be grouped into clusters in the cockpit crew pairing problem such that work satisfaction and well-being are increased without unacceptable increases in costs for the airline?

One of the problems faced by researchers in the field is being limited by technology resulting

in long computational times, especially when dealing with large groups of flight services. This has restricted research and forced researchers to turn to heuristics to find solutions. This has been the case for several researchers, such as for Vance et al., 1997. The second research question in our study is as follows:

How can the grouping of pilots in the cockpit crew pairing problem help to reduce the computation time of the problem?

We retrieve data from the Dutch airline KLM to answer the research questions. We model the original problem as a set partitioning Mixed Integer Program (MIP), presented in Chapter 3. As becomes clear in Chapter 4, it is not possible to directly solve this problem for the European flight services because of the number of flight services and various ways to combine them. This is why we use a column generation method with a shortest-path algorithm and a branch-and-price algorithm.

For both the clustered and the full European instances, the search for a complete set of schedules consists of determining an initial solution, generating high-quality week schedules to add to the model and finally, a method to come to an integer solution. The initial solution of clustered instances is determined using a special clustering model which balances different aspects of the clusters, such as the number of flights and countries visited.

A model developed for clustering flights shows promising results in finding clusters that can provide varied schedules for pilots. Comparing the results of clustered and unclustered instances shows a large reduction in problem instance size when using clustering, as well as a shorter computation time.

The methods are simpler for the airline's intercontinental services. These services are quickly optimally placed into schedules using the set partitioning model directly.

The remainder of this report is organised as follows. Chapter 2 discusses the existing literature relating to the research field. Chapter 3 describes the data and the relevant regulations and restrictions and gives a problem formulation. Chapter 4 explains the methods used, after which the results are discussed in Chapter 5. Finally, the thesis is summarised, and conclusions are drawn in Chapter 6.

Chapter 2

Literature review

A complete and flexible crew schedule is created from a full flight schedule with aircraft assignment by following three steps (Wen et al., 2021). At first, pairings are generated. These are complete shifts where pilots start and end at their home base. This can consist of just two flights with a relatively long rest period at the destination, as is often the case with intercontinental flights, or a sequence of flights with shorter rest periods, as is the case with European flights. This is the crew pairing problem. In this first step, only flights are considered. The demands of pilots and other crew members are not taken into account yet. In the second step, crew members are matched with the pairings. This is called crew assignment. After this, a complete schedule has been created. However, this is actually made in advance and does not take into account any circumstances that may arise last-minute. In the third step, during the operational phase, the schedule is adapted to deal with these changes.

The literature shows that researchers usually focus on one of the three steps and leave the remaining to others. Out of the three, the first (pairings generation) is the most intensively studied.

2.1 Pairings generation

Pairings have to be generated in such a way that all flights are included exactly once, which constitutes a set partitioning problem, as described in Vance et al. (1997). Various methods, both exact and heuristic, have been proposed for the generation of pairings. Some of the methods use the flights as they are to build the pairings, while others combine flights into duties. A duty can be seen as a day of work, starting with a briefing, followed by one or several connecting flights and ending with a debriefing. A pairing is then made up of duties where the first duty starts and the last duty ends at the home base and rests are scheduled in between duties. When

looking at (long) intercontinental flights, often only one flight is included in a duty, combined with a briefing and debriefing. Combining such a duty of flying to a destination with a rest at the destination (often overnight) and a duty back to the home base can result in a complete pairing. When it comes to European flights, a duty often consists of more flights and a pairing consists of more duties.

Studies generally view the problem as a set partitioning (SP) or a set covering (SC) problem. In the SP problem, all flights have to be included in exactly one of the pairings. In the SC problem, they have to be included at least once. Since the SP approach gives such a strict requirement, it can be more difficult and require more computation time to solve. However, it ensures a solution that does not contain any deadheading (crew members flying as passengers) because of overcovered flights which airlines usually consider expensive and inefficient.

The majority of researchers prefer the SP approach, but some find that the SC approach is better suited to their methods, such as Lavoie et al. (1988). They wish to allow deadheading in their solution because this limits the number of variables in the model.

2.1.1 Column generation & enumeration based methods

Most studies view the problem as a network where flights or duties are depicted as nodes, connected when it is, or might be, possible to combine them in a pairing. The source and sink nodes both represent the home base and are connected to duties departing and arriving at the home base, respectively. Valid paths always start at the source and end at the sink. As many requirements and conditions as possible are included in the network. If not all of them can be included directly, some paths in the network will likely be invalid.

Several methods, both exact and heuristic, can be used to find pairings in the network. Most studies viewing the problem as a network make use of a column-generation algorithm.

A possible approach to solve the set partitioning problem is to use branch-and-bound with dynamic column generation, finding new pairings in every iteration. This is also known as the branch-and-price approach. These pairings are found using either enumeration-based methods, heuristic methods or a multi-label-shortest-path approach.

Enumeration-based methods implicitly enumerate every possible pairing in every iteration of the branch-and-price algorithm, saving on the use of memory as they are not all saved. With large flight networks however the total number of possible pairings can be very large, making it difficult to enumerate all in every iteration. Despite this, some studies have found the method can be used on large networks when some adaptions are made. An example is Makri and Klabjan (2004), combining enumeration with approximate and exact pruning rules.

Heuristic methods do not guarantee the best pairings but can still produce good results, especially when combined with exact methods. In Quesnel et al. (2020), the researchers use a heuristic to include language constraints in the crew pairing problem, ensuring that required languages are spoken by the cabin crew. Their pricing problem is also based on these constraints as only subproblems not yet covered by crew members with the required language qualifications and crew members with these language qualifications are considered.

Vance et al. (1997) approach the problem as set partitioning and use the branch-and-price algorithm with a multi-label-shortest-path approach to solve it. By using this approach, the feasibility of paths can be checked easily. Some requirements and conditions cannot be incorporated into the network directly. The use of a multi-label rather than a single-label approach is needed because intermediate nodes could be included in many different paths. The calculation of the costs of a pairing can be complicated and partially depend on nodes in a path beyond the intermediate node that is considered for comparison. In the objective defined by the researchers, the costs are made up of more than just the sum of the cost of nodes. Knowledge of a full path is necessary to evaluate the costs of the path, making comparisons of paths at intermediate nodes difficult. Because of this, a great deal of tracking of possible paths was required to find the best paths, defined by not only costs but also the restrictions by the rules.

As a branching rule, Vance et al. (1997) use a set partitioning problem rule developed by Ryan and Foster (1981). This rule attempts a balanced branching tree by either not using two fractional duties at all or by forcing the use of both. In this case, the duties have to follow onto each other. In order to decide on which fractional follow-on pair the method will branch, the sum of the fractional values of the pairs is calculated, and the highest is chosen.

Vance et al. (1997) also explored the option of using a flight-based network with the same methods and found it to be faster because of a smaller use of memory than the duty-based network.

Lavoie et al. (1988) make the requirements set by airlines more explicit than Vance et al. (1997), exploring the situations that may arise and enter these as nodes into the network. This enables them to fully express the possible pairings of duties into an acyclic graph with topological ordering. This can be used to solve the problem with column generation. The set-up effectively finds integer solutions, even without the use of branch-and-bound techniques.

Vance et al. (1995) do not take the duties as given input but explicitly generate them by partitioning the flights. After that, they use them in the set partitioning problem of finding pairings. They found this approach gives a more robust linear programming bound on the optimal integer programming solution than using flights directly. The downside of the approach

is its complexity.

Rasmussen et al. (2011) use a similar branching rule as Vance et al. (1997) based on follow-on flights. However, they do not rely on column generation but on a method based on subsequence generation in an integer programming framework. The method is based on flight combinations within a pairing, one following the other. Some combinations can be more attractive than others, for example because of regulations. A solution can be found by creating a group of attractive subsequences and enumerating the pairings containing them.

Muter et al. (2013) use the branch-and-price algorithm with a multi-label-shortest-path approach for a robust method where flights can be added to the schedule and pairings can be adapted to contain them. To do this, they use both column and row generation. Pairings can be adapted to cover an extra flight in one of two ways; flights can be swapped with a different pairing to make room for the new flight, or the flight is put in between two other flights in a pairing where there is enough connection time.

Some methods specifically focus on the cockpit crew such as Yan and Chang (2002). Costs are calculated using average crew salaries rather than hours flown, as other studies have done. Optimal pairings are found within a flight-based network by using column generation.

2.1.2 Integer programming

AhmadBeygi et al. (2009) develop an alternative approach that does not require enumeration, based on integer programming. The developed model is a mixed-integer programming (MIP) model with binary decision variables. These decide whether or not flights follow up on each other and whether this happens within a duty or across two duties. The model also includes variables indicating whether a flight is the first flight of a pairing, the last, or neither.

2.1.3 Stochastic approach

It is also possible to find pairings using a stochastic approach. Examples of studies that do so are Dück et al. (2012) and Ionescu and Kliewer (2011).

Dück et al. (2012) developed a stochastic model for aircraft and crew scheduling. It focuses on creating a highly stable schedule, minimising the total costs of the pairings and any reactionary delay. Reactionary delay is the delay that occurs because of decisions made after the primary delay occurs. Primary delay is caused by factors outside the airlines' control. An example of a cause for primary delay is a decision by the airport's control tower to wait for a heavy storm to blow over. Reactionary delays could then occur on other flights because the delayed plane and its pilots are not available as planned. This is influenced by the airline's decisions and priorities.

Ionescu and Kliewer (2011) uses the stochastic approach to deal with delay as well. They created a model with so-called swap points in the schedule, moments when crews can be swapped with each other if necessary to minimise the delay.

2.1.4 Meta-heuristic approach

The meta-heuristic approach has gained interest in recent years. An example of a study with an evolutionary algorithm is Arayikanon and Chutima (2018). They compare two methods; the multi-objective evolutionary algorithm (based on the decomposition algorithm) and the honey bees' mating optimisation. With the first method, the problem is decomposed into subproblems which are solved simultaneously using neighbourhoods. The second method is based on single-point cross-over to produce offspring, after which mutation is applied.

2.2 Crew assignment

After all the pairings are generated in such a way that all flight services are taken into account, each pairing is assigned to crew members. Even though fewer studies have focused on this step, various methods have been developed to assign pairings optimally. Traditionally, the most common methods are generate-and-optimise and column generation, such as in Day and Ryan (1997) and Ryan (1992). Fahle et al. (2002) found that the increasingly complicated regulations for European airlines could not be captured fully by pure column generation. This is why the researchers of this study use constraint programming in the subproblem to incorporate the collection of rules and regulations posed on crew members' schedules. They found that this approach, combined with a shortest path algorithm also expressed as a constraint in the subproblem, showed to be promising.

Aside from the methods mentioned above, research has been performed towards the use of (hybrid) genetic algorithms. An example of a study with a genetic algorithm is Ozdemir and Mohan (2001), where three operators are available for reproduction in the algorithm. The first is a set-based operator which attempts to combine the duties in the two selected pairings as much as possible. The second operator is a time-based operator, cutting the first pairing at a certain time point and combining it with the second. The third operator is distance preserving, matching up the two pairings as much as possible and filling up the rest of the new pairing with new elements not present in either of the two parents.

2.3 Operational

Once the pairings are generated and assigned to crew members, the schedule can be used in practice. During this implementation, also called the operational phase, situations arise frequently which makes it necessary to adjust the initial plan. Since the adjustments are often made by planners based on their expertise, not many studies have focused on the optimisation of this phase. One study which provides some insights is Stojković et al. (1998). Since time can be critical in decision-making in the operational phase, one of the objectives of Stojković et al. (1998) is to solve the optimisation problem as quickly as possible. They also focus on minimising the total costs and the number of changes made to the schedule. Two main types of test problems were tested, one with three delayed flights making crew members late for their next flight services and one with one unavailable crew member needing to be replaced. The problems were solved with a column-generation method in a reasonable amount of time.

2.4 Integrated approach

Even though most researchers prefer to focus on one of the three steps of scheduling, some do seek approaches that integrate more aspects of the process into a single model. Some involve the schedules created before pairing generation such as aircraft routing. A method used by several of these integrating studies is Benders decomposition. Four of these studies are Mercier et al. (2005), Mercier and Soumis (2007), Papadakos (2009) and Sandhu and Klabjan (2007).

Mercier et al. (2005) integrates aircraft routing and crew scheduling into one model with Benders decomposition and column generation. The master problem is based on the crew scheduling problem, while the subproblem solves the aircraft routing problem. This integrated approach helps them optimise over the constraints affecting both stages.

Mercier and Soumis (2007) do not only integrate aircraft routing and crew scheduling into one algorithm but also flight retiming. The algorithm uses Benders decomposition, column generation and dynamic constraint generation. Dynamic constraint generation is necessary because of the problem's increased complexity compared to Mercier et al. (2005) due to the aspect of flight retiming. They find their integrated approach effectively reduces cost and the number of used aircrafts.

Papadakos (2009) also uses Benders decomposition for an integrated approach. In this case, fleet assignment, maintenance routing and crew pairing are optimised simultaneously. This is done with accelerated column generation. The acceleration comes from using deepest-cut pricing, adding the pairings with the lowest reduced costs.

Sandhu and Klabjan (2007) integrate fleet assignment and crew pairing. They explore two different models to achieve an optimal solution, Lagrangian relaxation with column generation and Benders decomposition. In most of their instances, the Benders decomposition method is outperformed by the method employing Lagrangian relaxation.

Chapter 3

Problem description

We need to solve the Crew Pairing Problem (CPP) for cockpit crew. This means we try to find the best way to create optimal pairings of flights while keeping all regulations in check. The goal is to do this in a way that will improve the lives of the pilots, being able to build meaningful relationships with coworkers and swap shifts.

3.1 Input data

To create the pairings, flight data is required. This data has been made available by Air France-KLM through an Application Programming Interface (API). It contains information on KLM's weekly flight schedule during several seasons during the years 2017 through 2019. For this paper, the summer of 2018 is selected, defined by KLM as the period from 25-03-2018 to 27-10-2018, the period of daylight savings. According to KLM's raw data, this season 5067 flights were operated by KLM's pilots per week.

Data is available for each individual flight. The weekday, time and place of departure and arrival are given, as well as the aircraft (sub)type which is important because pilots are usually only licensed for one type. In this case there are 11 subtypes present in the data. These can be grouped into 5 main license types, summarised into Table 3.1.

Table 3.1: Summary of the main aircraft license types

License type	**Subtypes**	**Weekly flights**	**Destination**
Embraer	E170, E190	2,228	Europe
A330	A332, A333	244	Intercontinental
B737	B737, B738, B739	1,882	Europe
B747	B744	132	Intercontinental
B777 & B787	B772, B77W, B789	581	Intercontinental

3.2 Regulations and restrictions

Both pairings and duties are subject to certain rules. Each duty requires a briefing period at the beginning and a debriefing period at the end. The elapsed time of a duty, including briefing and debriefing, cannot be longer than a certain maximum. The time between two flights in a duty is bound by a minimum and a maximum as well. Duties and pairings are always scheduled within a license type, meaning pairings and duties always contain flights of one license type only.

There is a difference between European and intercontinental flights for many of the regulations. For the two European license types, the main restrictions are as follows:

- Maximum elapsed time in a duty is 11 hours, including briefing of 1 hour and debriefing of 30 minutes.
- The time between two flights in a duty has to be at least 30 minutes and at most 3 hours.
- A duty cannot include multiple flights visiting the same destination. In a duty, an airport can only function once as an arrival destination and once as a departure destination. The home base, Amsterdam, is excluded from this regulation.
- A pairing can be maximally 5 days.
- Maximum of 5 flights on the first day of a pairing and 4 flights per day for the rest of the pairing.
- Rest periods in between duties have to be at least 12 hours when it takes place at the home base and at least 10 hours otherwise.
- The rest period has to be at least as long as the duty preceding it.

When flying intercontinental routes, a pairing usually consists of 2 duties, both with 1 flight and a long rest period in between. On certain routes, duties can consist of two flight services. One service to or from the home base or hub (Amsterdam) and another from one abroad destination to another. This is usually only the case when two destinations are being served with one aircraft. More details about these special cases are given in Chapter 5. An intercontinental duty can only exceed 12 hours if it consists of one long flight (over 10.5 hours) and a third officer is present (assigned in the Crew Assignment problem). The rest period in between duties has to be at least 24 hours. An intercontinental pairing can last up to 7 days maximum. It is possible a pairing consists of three duties when flights have to operate between local destinations.

3.3 Problem formulation

To solve the problem as described, a mathematical model is required. The necessary notation to present our model mathematically is available in Table 3.2.

Table 3.2: Description of the sets, parameters and variables used for the problem formulation

I	Set of license types
i	Indices of license type in I
P_i	Set of all available pairings for license type i
p	Indices for pairings in P_i
D_i	Set of all duties in license type i
d	Indices for duties in D_i
M_i	Set of all flight services in license type i
m	Indices for flight services in M_i
f_{mp}	Binary parameter which is 1 when flight service m is present in pairing p
e_p	Parameter equal to the elapsed time of a pairing p, expressed in minutes
l_p	Parameter equal to the number of times on destination for the full hour between 12:30 and 1:30 PM during p
d_p	Parameter equal to the number of times on destination for the full hour between 6:30 and 7:30 PM during p
h_p	Parameter equal to the number of nights spent in a hotel on destination during a pairing p
c_p	Parameter equal to the costs of a pairing p, expressed in euros, defined below
x_p	Binary (decision) variable which is 1 when pairing p is used in the solution
π_m	Dual variable of flight service m

The set partition model is as follows for all licence types i in I:

$$min \sum_{p \in P_i} c_p x_p \tag{3.1}$$

$$\sum_{p \in P_i} f_{mp} x_p = 1 \qquad \forall m \in M_i \tag{3.2}$$

$$x_p \in \{0, 1\} \qquad \forall p \in P_i \tag{3.3}$$

In this model, the total sum of (selected) pairings is minimised in (3.1). This is restricted by (3.2) which demands every flight to be part of exactly one selected pairing. Finally (3.3) denotes the selection of pairings as a binary decision, where a pairing can either be selected fully (value 1) or not at all (value 0).

c_p, the cost of pairing p, can be defined in different ways, depending on the airline's cost structure and wage system. Pilots at KLM receive a fixed salary, independent on how much the KLM schedulers decide to employ them. This means the pairings' structure does not influence salaries directly, but it is likely that more pilots are required when a schedule is inefficient, dividing flight services into a large number of pairings or using unnecessarily long layovers. To compensate pilots for their costs abroad while on duty, they do receive a small amount to pay for their lunches and dinners on destination. Hotels, needed when staying overnight in cities other than Amsterdam, are always booked by the KLM office in advance and include a breakfast

service.

Summarising, c_p should be based upon two factors: efficient use of the available pilots and their time as well as the costs made to compensate pilots while on destination. The first factor can be measured by a combination of two elements, a constant cost element c_c and by calculating the elapsed time of the pairing e_p multiplied by an approximation of the salary of a pilot or copilot per minute of elapsed time, denoted by s. The second factor can be measured by using parameters l_p, d_p and h_p as mentioned in Table 3.2. These parameters are multiplied by an approximation of their costs, denoted by c_l, c_d en c_h. The final cost function is then as follows:

$$c_p = c_c + se_p + c_l l_p + c_d d_p + c_h h_p \tag{3.4}$$

Chapter 4

Solution approach

This chapter explains how the partition model described in the previous chapter can be solved. In our case, it depends on whether a schedule is created for a European or intercontinental set of flights and whether a full set of a license type is considered or if the set has been grouped into clusters. The simplest problems here are those where a schedule is created for an intercontinental set because the number of flights per licence type is sufficiently small to directly enumerate all possible pairings, as is explained at the end of this chapter.

Due to the large number of flights in the two European license types seen in Chapter 3, it is not possible to enumerate all possible pairings directly for these license types. To find an integer solution, a truncated branch-and-price algorithm is used, which relies on column generation to produce the needed pairings. First, a master and a restricted master problem are constructed. Here, the master problem is the LP relaxation of the set partition problem containing all possible pairings. This means the master problem is equal to the model presented in the previous chapter when the binary requirement for the pairing variables is replaced by a non-negative requirement. The restricted master problem is equal to the master problem but only contains generated pairings. To find a non-integer optimal solution, column generation can be used with the restricted master problem. The relaxation allows for the partial use of individual pairings, and it starts with an initial set of pairings that needs to be found before starting the algorithm. The pricing problem for all column generation iterations is based on finding pairings with negative reduced cost, described later in this chapter. After an optimal non-integer solution is found, an integer solution can be found by fixing flight pairs on an iterative basis with column generation after each iteration.

As seen in the literature earlier, two starting points are possible for creating pairings. They can be created by combining flight services directly or by combining duties that consist of one or more flight services. Most methods in the literature are duty-based, allowing the incorporation

of some of the restrictions early on. Due to the number and complexity of restrictions on duties presented in the previous chapter, our methods will also be duty-based. In contrast with most literature, however, duties are not available as input data in our case, so they need to be generated from the flights to be used when creating pairings. The methods used to generate the needed duties are explained in this chapter, as well as how to combine them into pairings. It also explains the adaptions necessary for the clustering of flight services.

This chapter discusses all aspects involved in finding an integer solution. It starts with discussing duty generation in Section 4.1, which is necessary for the construction of the duty node network used in column generation. This is followed in Section 4.2 by information on the generation of pairings in the initial solution for the European license types when no clustering is taking place. Section 4.3 describes the method used to cluster the involved airports and flight services. More details on the duty node network can be found in Section 4.4. This network is used in Section 4.5, which combines all elements to describe the full algorithm transforming the initial ineffective solution to a fully integer solution through column generation and branching. Finally, the methods used for intercontinental flight services are discussed in Section 4.6.

4.1 Duty generation

Duties can be generated by enumeration with a depth-first search approach, as suggested by Vance et al., 1995. Every individual flight is considered a tree's root node, and flights eligible as its connections are depicted as its child nodes. Connections of descendants are added as long as regulations allow. The most important regulations are the minimum and maximum connection time between flights, the maximum number of flights allowed within one duty and the maximum duty length. Every path from the root node to any of its descendants is a possible duty. The root node by itself is also a possible duty. This automatically guarantees every flight is included in at least one duty.

In other words, every flight is seen as a starting point for a duty, connecting to other flights departing from the city it arrives in as long as regulations are met, such as the minimal and maximum connection time and maximum duty duration.

4.2 Initial pairings generation European flight services

As discussed earlier, the algorithm is a branch-and-bound algorithm with dynamic column generation, also known as the branch-and-price algorithm. Since not all nodes in the brand-and-bound tree are explored in this study, we are working with a truncated branch-and-price algorithm. In

this version of the branch-and-price algorithm, we dive down the tree depth-first and only branch out towards other nodes when a node turns out to be infeasible. We start with a restricted master problem containing a small set of pairings. This set has to be able to feasibly solve the LP relaxation of the set partition problem, covering each flight exactly once. In literature, the initial set of pairings needed to start the algorithm often comes from solutions of the airline in the past. Those are not available for this study, so a solution must be found through heuristics or the use of artificial flight services and duties.

For the creation of an initial set of pairings, various methods are possible. The first and most simple method is to create an artificial flight service for every flight service present in the problem and combine each pair into a pairing. This creates a solution that is suitable as a starting point since it contains all flight services exactly once, but it contains only artificial pairings, none of which are eligible for the final solution. The second option is to combine all (non-artificial) flight services into pairings of two flight services, each with one flight service departing from the home base (outbound) and one returning to the home base (inbound). This leads to a feasible but inefficient solution with more pairings than necessary, but the symmetric nature of the problem makes it possible to cover the vast majority of flight services without the need for artificial services. If any flight services do remain uncovered after the algorithm is executed, the remaining flights can be matched with an artificial flight and combined into an artificial pairing with an increased cost, giving an incentive to the model to not use the pairing unless absolutely necessary.

A third option is a multi-step approach, attempting to find longer and more efficient pairings containing a large number of flight services per pairing. Since the quality of the pairings is higher, it is more likely these pairings will continue to be selected throughout the algorithm and in the final solution. The major downside of this approach, however, is how it places flight services in large pairings, making it difficult for the column-generation algorithm to change the pairing selection later on in the truncated branch-and-price algorithm. When deactivating a pairing, all flight services of that pairing need to be placed into other pairings. To provide the best starting point possible, the second option is selected, providing the flexibility necessary without the addition of artificial flight services.

The approach is described in Algorithm 1. As noted above, it focuses on generating pairings containing two flight services per pairing. It is possible both flight services take place on one day, generating a single-duty pairing, or each flight can take place on a separate day, generating a two-duty pairing. As shown in the algorithm below, the method utilises the single-duty pairing options as much as possible without covering flight services more than once before continuing to

two-duty pairings. These are generated by looking at single-flight duties not covered yet by the single-duty pairing generation starting in the airline's home base Amsterdam. These duties can then each be combined into a pairing with a non-covered connecting single-flight duty back to Amsterdam. When more than one return duty is available, the connection creating the shortest overnight lay-over is selected.

Algorithm 1 Heuristic to find the first group of initial pairings European flight services

Require: $i = 0$ or $i = 2$ ▷ Only for European services
$U_i \leftarrow M_i$ for uncovered flights ▷ All flights are uncovered
$A_i \leftarrow D_i$ for available duties ▷ All duties are available
$P_i \leftarrow \emptyset$ for generated pairings
for all $m \in U_i$ **do** ▷ Every flight is present in a set of duties
    $l \leftarrow$ list available duties flight m
    $L_i \leftarrow (m, l)$
end for
$T_i \leftarrow$ two-flight duties $\in D_i$ starting and ending in home base Amsterdam
$S_i \leftarrow$ single-flight duties $\in D_i$ starting in home base Amsterdam
for all $d_t \in T_i$ **do**
    if $d_t \in A_i$ **then**
        new pairing p_t^* generated with duty d_t
        $P_i \leftarrow P_i \cup \{p_t^*\}$ ▷ Update generated pairings
        for all flight services $m \in d_t$ **do**
            $U_i \leftarrow U_i \setminus \{m\}$ ▷ Update uncovered flights
            $A_i \leftarrow A_i \setminus \{l\}$ ▷ Update available duties
        end for
        update L_i according to changes in U_i and A_i
    end if
end for
for all $d_s \in S_i$ **do**
    if $d_s \in A_i$ **then**
        if any single-flight connecting duties in A_i to Amsterdam exists **then**
            select connecting duty $a_s \in A_i$ creating the shortest pairing with d_s
            pairing p_s^* generated with duties d_s and a_s
            $P_i \leftarrow P_i \cup \{p_s^*\}$ ▷ Update generated pairings
            for all flight services $m \in d_s \cup a_s$ **do**
                $U_i \leftarrow U_i \setminus \{m\}$ ▷ Update uncovered flights
                $A_i \leftarrow A_i \setminus \{l\}$ ▷ Update available duties
            end for
            update L_i according to changes in U_i and A_i
        end if
    end if
end for

4.3 Adaptions for clustering flights

An advantage of generating duties ourselves is that we can enforce additional restrictions in the process. In the adapted model, flight services are grouped into smaller clusters, and duties can

be generated for each cluster. In this case, where we are faced with grouping the flights of a single-hub airline, it is preferable to group all the flights servicing a particular airport abroad into the same cluster. This ensures connections within airports can take place as efficiently as possible. Taking this into account, we will group airports in order to cluster flights. When doing so, we can focus on pilots' well-being, keeping their schedules varied with flight services of various lengths and visiting several countries. In practice, this means the flights and airports are grouped in a small number of clusters per license type, optimising the number of countries in each cluster while keeping the number of flights in each cluster at a similar level. By using a MIP model, not only can the number of flights, airports and countries per cluster be optimised, but the initial solution for each cluster can also be produced. The total costs of the pairings in these initial solutions can be taken into account as an element in the objective as well. The initial solution value is also part of the objective in the model because it can give an indication of the final solution value, as is explained later on. Optimising through this model eventually leads to optimal and balanced clustering and a feasible initial solution, providing the best possible starting point for the rest of the algorithm.

The number of clusters per license type is kept low because a large number of small clusters could be difficult to manage during operations and have an adverse effect on the pilot's well-being, servicing the same airports from a small set time after time. For this study, we focus on grouping flights into three clusters per European license type, with tests expanding into grouping flights into two and four clusters in order to give a broader analysis of the effects of clustering.

The main decision variable of the clustering MIP is based on the idea of combining flights into basic two-flight pairings. These consist of three elements; one outbound flight, a layover at the destination and an inbound flight back to the home base. The pairings are enumerated before initialising the model and only in the instances where there is a feasible amount of layover. This is defined in line with the restrictions explained earlier. Flights are combined into a pairing when there is either a layover between 30 minutes and 3 hours (in line with placing them together in a duty) or a layover between 10 hours and the remainder of the arrival day plus 48 hours (in line with placing them in separate duties). The complete model and its variables and parameters are shown and explained below.

Table 4.1: Description of the sets, parameters and variables used for the clustering model

G_i	Set of all clusters in license type i
g	Indices for cluster in G_i
A_i	Set of all airports in license type i
a	Indices for airport in A_i
N_i	Set of all countries in the schedule of license type i
n	Indices for country in N_i
P_i^2	Set of all possible two flight pairings in the schedule of license type i
p	Indices for pairing in P_i^2
F_i	Set of all flight services in the schedule of license type i
f	Indices for flight service in F_i
v_{an}	Binary parameter which is 1 when airport a is in country n
o_{fp}	Binary parameter which is 1 when flight f is present in pairing p
t_{af}	Binary parameter which is 1 when airport a is the destination or origin of flight f
h_a	Parameter which is equal to number of flights connected to airport a
c_p	Parameter which is equal to the costs of pairing p
x_{pg}	Binary (decision) variable which is 1 when pairing p is used by cluster g
v_{ng}	Binary variable which is 1 when at least one airport of country n is in cluster g
s_{ag}	Binary variable which is 1 when airport a is in cluster g
y	Variable used with minimisation of difference of number of pairings across clusters
w_y	Parameter used in the objective function for weight of variable y
m	Variable used with minimisation of difference of number of countries across clusters
w_m	Parameter used in the objective function for weight of variable m
z	Variable used with minimisation of difference of number of airports across clusters
w_z	Parameter used in the objective function for weight of variable z

The objective of the model is to optimise several aspects which are combined by the model and its objective function (4.1). First of all, the initial solution is optimised by calculating the total costs of the selected pairings x_{pg}. Secondly, a balance is found by using y (related to the number of pairings per cluster), m (related to the number of countries per cluster) and z (related to the number of airports per cluster), all weighted by a separate parameter w.

The initial solution is taken into account in the objective function because it is indicative of the final solution. The initial solution consists of pairings made up of two flight services forming a pair. Even though the two flight pairings will most likely not be used in the final solution, it is probable the pairs will be incorporated in larger pairings produced later on, which could be part of the final solution.

Constraint (4.2) is included in the model to make sure every flight $f \in F_i$ is present in exactly one selected pairing, preventing under and over-coverage of flights. Constraint (4.3) ensures every airport $a \in A_i$ is covered by exactly one cluster. Constraint (4.4) is involved in setting v_{ng} to 1 when at least one airport $a \in A_i$ in a certain country $n \in N_i$ is selected by a cluster $g \in G_i$. Constraint (4.5) sets v_{ng} to 0 when no airports in the country $n \in N_i$ are selected by cluster $g \in G_i$. Constraint (4.6) ensures that when a flight $f \in F_i$ is allocated to

a cluster $g \in G_i$ through a pairing $p \in P_i^2$, the associated airport $a \in A_i$ and any other flights connected to airport a are also included in the same cluster g.

Constraint (4.7) is present in the model to establish the minimum number of pairings covered across clusters. This minimum is maximised through the objective function. Constraint (4.8) does the same for the minimum number of countries covered across clusters, and Constraint (4.9) for the minimum number of flights. Adding these constraints and the factors into the objective function ensures every cluster covers as many pairings, countries and airports as possible without causing a great imbalance between the clusters. Constraints (4.10) through (4.15) set the boundaries of all variables involved.

From the results of this model, the clusters are defined by the airports selected through s_{ag}, splitting up all airports and, by extension, all flight services across the clusters. The initial pairing solution needed for each cluster to continue is provided by the variables x_{pg} and their values.

The clustering model is as follows for all European licence types i:

$$min \sum_{g \in G_i} \sum_{p \in P_i^2} c_p x_{pg} - w_y y - w_m m - w_z z \tag{4.1}$$

$$\sum_{g \in G_i} \sum_{p \in P_i^2} o_{fp} x_{pg} = 1 \qquad \forall f \in F_i \tag{4.2}$$

$$\sum_{g \in G_i} s_{ag} = 1 \qquad \forall a \in A_i \tag{4.3}$$

$$s_{ag} v_{an} \leq v_{ng} \qquad \forall g \in G_i \quad \forall n \in N_i \quad \forall a \in A_i \tag{4.4}$$

$$\sum_{a \in A_i} s_{ag} v_{an} \geq v_{ng} \qquad \forall g \in G_i \quad \forall n \in N_i \tag{4.5}$$

$$\sum_{p \in P_i^2} x_{pg} o_{fp} = \sum_{a \in A_i} t_{af} s_{ag} \qquad \forall g \in G_i \quad \forall f \in F_i \tag{4.6}$$

$$\sum_{p \in P_i^2} x_{pg} \geq y \qquad \forall g \in G_i \tag{4.7}$$

$$\sum_{n \in N_i} v_{ng} \geq m \qquad \forall g \in G_i \tag{4.8}$$

$$\sum_{a \in A_i} h_a s_{ag} \geq z \qquad \forall g \in G_i \tag{4.9}$$

$$x_{pg} \in \{0, 1\} \qquad \forall p \in P_i^2 \quad g \in G_i \tag{4.10}$$

$$v_{ng} \in \{0, 1\} \qquad \forall n \in N_i \quad g \in G_i \tag{4.11}$$

$$s_{ag} \in \{0, 1\} \qquad \forall a \in A_i \quad g \in G_i \tag{4.12}$$

$$y \in \mathbb{N} \tag{4.13}$$

$$m \in \mathbb{N} \tag{4.14}$$

$$z \in \mathbb{N} \tag{4.15}$$

4.4 Network generation and data management

The network needed for generating pairings can be built in a way suggested by Vance et al., 1997. All generated duties are depicted as nodes, and arcs are placed between nodes whenever two duties can be connected.

Connections are always sought among duties of the same license type. The seeking of connections is slightly different for European flights than for intercontinental flights due to the differences in regulations. When considering European-based duties, duties of the same license type starting the following day and the day after are considered possible connections. It is

checked if the arrival airport of the current duty and the departure airport of the connection candidate match and if the resting period between duties is long enough. Since duties are only allowed to contain five flights on the first day of a pairing, it is only possible to select duties with five flight services as the first duty of a pairing, never as a connection candidate for another duty.

All nodes representing duties departing from the home base (in this instance, Amsterdam) are connected to the source of the network, and all nodes representing duties arriving at the home base are connected to the sink. The duty nodes store several components; the departure and arrival airport, start and end time, the total flight time and the total of the dual values of the flights in the duty. These dual values change with the iterations of the algorithm to be in line with the LP solution of the restricted master problem.

Given the data structure, it is likely that the number of duties generated will be too large to be consistently managed in the computer's local memory, especially when they are used to find connections and pairings. This is why the flight and duty data is stored in an SQLite database. This database does not only contain data on the flights and duties themselves but also information on connections to other duties and the presence of specific flights in certain duties.

4.5 The complete algorithm for European flight services

After an initial set of pairings for a license type i is established, as well as a network of nodes (duties) and arcs (connections between duties), pairings for European flight services can be generated using paths in the network with the algorithm as follows, in line with Vance et al., 1997. The initial solution is calculated using the restricted master problem and the initial set of pairings. After this, we use the network for column generation. The following defines the reduced cost of pairing x_p with π_m as the dual variables:

$$RC(x_p) = c_p - \sum_{m \in M_i} \pi_m f_{mp} \tag{4.16}$$

The pricing problem consists of finding the pairing with the lowest reduced cost and is solved with every column generation iteration, where any negative reduced cost pairings found are added to the model. When no more negative reduced cost pairings can be found, column generation can be ended. The reduced cost of a pairing can be calculated by using the details from the pairing directly or through a network by using data present in duty nodes and connection arcs. When searching for pairings with the generated network, the lowest found reduced costs are

saved in the duty nodes. Calculating the reduced cost of a potential pairing is then done by summing the reduced costs of the predecessor, the additional cost on c_p saved in the connection arc and the values of the dual variables relevant to the added duty. When the calculated reduced cost is lower than the previously lowest found reduced cost or when it is the first time reduced cost are calculated in the node, the reduced cost are saved in the node.

After the LP relaxed restricted master problem is solved to optimality by finding all negative reduced cost pairings, the truncated branch-and-price method is used. Both are further explained below.

4.5.1 Column generation through a shortest-path method

Two procedures are used to extract all pairings with a negative reduced cost. Both use the same column generation method with the same network, but the first solves a shortest-path problem in a somewhat greedy manner to extract the highest-quality pairings in an efficient way, while the second procedure always fully explores the entire network to extract any possible remaining pairings. This is done to increase efficiency and decrease computation time.

In the first method, a small network is set up from scratch for every duty starting at the home base, using the database of duties and connections, only using the duties needed for a network around the starting duty. At first, all direct connections from the starting duty are analysed, and the reduced costs are calculated and updated as described earlier. The algorithm then proceeds using a depth-first approach, selecting the nodes with the lowest reduced cost found to proceed and updating reduced-cost labels as it continues. This increases the probability of finding high-quality pairings. During this search, the number of so-called good and bad ends found are counted. These are defined as encounters in the paths with duty nodes ending at the home base with either a negative (good) or positive (bad) reduced cost. When either of these counters reaches its threshold, the search is stopped, and the found paths with a negative reduced cost are used to create new pairings. This threshold is defined in advance before starting the algorithm to limit the run time of the algorithm. When the threshold for good ends is reached first, the search has found a sufficient number of high-quality pairings using the depth-first approach to end the search. When the threshold for bad ends is reached first despite using a depth-first approach, few high-quality pairings are present in the network when the starting duty is used as its first duty and continuing will likely not give good results, so the search is ended. A maximum number of pairings to add with every starting duty is set beforehand. When more negative reduced-cost pairings than the maximum are found, the highest-quality pairings are added. In order to increase efficiency, the starting duties are split up into groups, where

each group is then taken care of by their own thread across a multi-threaded system. After the pairings of all groups are found, they are added to the restricted master problem, which is then solved. The procedure is repeated until no more than a small number of pairings is found. When this is the case, the algorithm switches to the second method to find the remaining pairings.

This second method is focused on finding all remaining negative reduced cost pairings by following the same procedure as in method one but without any thresholds set for good and bad ends found. This means a small network is set up for every starting duty, and every duty node in each network is explored, following the same approach as in the first method. All pairings with negative reduced costs found are added to the restricted master problem, which is then resolved. This procedure is repeated until no more pairings are found.

Once the method is completed, the solution is checked. If the solution is integral, the problem can be considered solved, and no further steps are required. It is also possible an integer solution with the optimal value is found in an earlier iteration of column generation. This is then also considered a full solution. To make sure it is possible to retrieve the solution when the situation has taken place, the integer solution with the lowest objective value is saved and updated throughout the first two methods. If the optimal integer solution is not found at the end of the second method, a truncated branch-and-price algorithm is started to find an integer solution by fixing the so-called best follow-on flight pairs.

4.5.2 Truncated branch-and-price method

Branching is based on forcing two flights to appear consecutively or never consecutively together in the same pairing. The flight pairs to branch on are found by looking closely at all partially used pairings in the fractional solution. These pairings are defined as having a solution value between 0 and 1. In these pairings, we look for flight pairs with flights that appear together in at least one fractional pairing and separate from each other at least once as well. If more than one of these pairs is found, the pair occurring consecutively the most gets priority to be branched on. To determine this, a value is calculated for every possible flight pair. This is the sum of solution values of partially selected pairings where they appear consecutively in. The flight pair with the value closest to 1.0 is then selected as the best follow-on pair.

When branching on the chosen follow-on, in one branch, the pair always has to occur consecutively in a pairing and in the other, the pair cannot be combined consecutively into a pairing together. These changes can be implemented into the network by removing arcs and nodes. In the first case, the adaptions depend on whether the best follow-on pair selected is part of one duty or if it is a flight pair with an overnight layover. When the flights of a follow-on pair take

place on the same day during one duty, all nodes representing a duty with only one of the two flights have to be removed. In the case of an overnight layover, the first flight service of the pair takes place as the last flight of a duty, and the second flight of the pair is the first flight of the following duty. In this case, all nodes representing a duty with the first flight service in any spot other than as its last flight have to be removed, just as duties with the second flight service in any spot other than as its first flight. All arcs connecting duties ending with the first flight to duties that do not start with the second flight have to be removed.

This same procedure is applied to any flight pairs appearing in fully used pairings. Every time a follow-on pair is selected and the network is restricted, all fully selected pairings are analysed as well. The consecutive flight pairs within these pairings are fixed in the same way as the follow-on pairs. Note that this ensures the flight services appear consecutively in the final solution, but the pairing that is fully selected at the moment of fixing is not necessarily fixed. The flight pairs could possibly end up in a different final pairing.

In the second case of branching, which is only used when the first case of branching gives an infeasible result, the network has to be adapted so the flights cannot be combined consecutively into a pairing. This means all duties containing the flights consecutively have to be removed, as well as any arcs between duties combining the flight consecutively.

Branching is focused on flight services rather than on duties because it is more complicated to define the branches when using duties rather than flight services. When branching on duties, we could force duties to always appear consecutively in one branch or never together in the same pairing in the other. This can, however, create problems because flight services often have many duties to choose from, and if necessary, the duty pair will simply be not used. In the other branch, where duties are not allowed to be in the same pairing together, it depends on whether the duties have a direct connection or if it is possible to form a pairing with the two duties and a duty in between. In the latter case, this could cause a problem during column generation.

During the truncated branch-and-price procedure, a depth-first search is performed, fixing the best follow-on as explained above each time, restricting the network further and further, forcing good follow-on pairs to appear consecutively. After fixing the best follow-on pair, as explained above, requiring two flights to appear consecutively, it has to be checked whether the remaining pairings are capable of forming a solution covering each flight in the network exactly once. This has to be done because it is possible too many earlier generated pairings are deleted with the restricting of the network. To prevent infeasibility because too many pairings are removed, an artificial pairing with high pairing cost is introduced for all unrestricted flights (flights that are not fixed yet). These pairings consist of an unrestricted flight together with

one artificial flight, taking place right before or right after the original flight, depending on whether the original flight is inbound or outbound. Adding these artificial pairings ensures a full solution with valid dual variables, enabling the algorithm to continue. The setting of a high cost of artificial pairings has two main effects. First, the model selects as few of them as possible, preferring the originally created and cheaper pairings. Secondly, when an artificial pairing is selected, the high cost of the pairings influences the dual values of flights included in them, encouraging the column generation algorithm in the next step to generate a non-artificial pairing to use in its place. Over time, before the next follow-on pair is selected and fixed, the algorithm should be able to eliminate the use of artificial pairings completely, and this can also be monitored.

In our truncated branch-and-price algorithm, for the pricing problem any pairings with negative reduced costs have to be found, and an adapted version of procedure two is used. In this approach, a restricted network is built from every remaining starting duty and explored fully. Duties are explored in the order they were added to the explorations list. When all pairings with negative reduced costs are found through one or several iterations, the restricted master problem is solved once again. If the solution is integer and does not use any of the artificial pairings, the algorithm is stopped. Otherwise, the next best follow-on pair is found, and the algorithm continues as described above.

If it is impossible to find an integer solution without the consistent use of artificial pairings with the truncated branch-and-price algorithm as described, backtracking through the fixed best follow-on pairs is necessary, and the other branches are used, forbidding flights to appear consecutively.

4.5.3 Adaptions clustered European flight services

Very few adaptions are necessary to be able to use the algorithm for the earlier described clusters. Rather than using the algorithm for a full license type, it is used for the separate clusters, with a smaller group of flights, duties and a different initial solution. Using these smaller networks, the algorithm can be used in the same way as described above. To come to a final full solution, the solution values of the clusters can be summed per license type.

4.6 Pairing generation for intercontinental flight services

As mentioned earlier, the smaller number of flight services and connections for intercontinental flight services make it possible to exhaustively enumerate all possible pairings and use them directly in the set partition model to find an optimal solution. A truncated branch-and-price

algorithm is not necessary.

When international pairings are enumerated, this is done in a similar way as was explained earlier in Section 4.1 for duties. Every duty starting in Amsterdam is considered as the root of a tree, with duties connecting to this duty as its child nodes. As long as the ending city is not equal to the home and the maximum pairing length of seven days is not reached on a path, connections of a descendant are added. As soon as all pairings ending at the home base are found, the next duty starting in Amsterdam is considered.

4.6.1 Adaptions clustered intercontinental flight services

It is possible to use the clustering method used on European flight services on the intercontinental flight services as well, but because of the nature of the intercontinental flight schedule, this would not change the final solution. Without clustering, each intercontinental pairing only visits one or two destinations and splitting up destinations would not result in any significant changes to the schedule or the computational time. If clustering is still desired to group flights and pilots, the solution pairings can be grouped by airline schedulers after the final solution is determined.

Chapter 5

Results

All results were obtained by running Java 8.0 on a computer with Intel Core i5-10210U CPU 1.60GHz 2.11GHz and 8GB RAM. Where required models were solved using IBM CPLEX 20.1.0.0, and data was managed by SQLite 3.39.4. Details on the code written for this thesis can be found in the appendix.

5.1 Data preparation

Before the methods explained in Chapter 4 can be applied, the data has to be checked and prepared. During exploration, both duplicate and missing flights were encountered. In addition to these, the data included flights with the wrong arrival date due to time zone shift errors.

The initial data contained a small number of duplicate intercontinental flights, occurring in instances where a long flight service is performed by flying via a different destination, for example to drop off or pick up some of the passengers, to fill up the fuel tank or to switch out pilots or cabin crew. Depending on the nature of the stop, flights are documented in the flight data in one of two ways. When passengers and pilots can use the stop to board or de-board, the partial services making up the full service are documented in the data. If the full flight service is present in the data, it is removed as preserving it would lead to over-covering of the service. This type of stop-over is, for example used to service two islands in the Dutch Caribbean with one aircraft. When the stop is only for aircraft-related reasons, such as filling a fuel tank, only the full service is preserved, and pilots are not allowed to switch during the stop-over as is possible in the first case.

Using the method above, 154 flight services are removed from the data. The number of flights performed by the A330 aircrafts is reduced to 180 and the flights performed by the combined license type Boeing 777 and Boeing 787 is reduced to 491. The removal method does not affect

the other groups because they do not perform stop-over services.

The initial data also contained cases of missing flights. If these are not added to the data, airplanes and pilots can be “lost" in the schedule and it becomes impossible to create a fully functional solution. An example is the services with the Boeing 737 airplanes between Amsterdam and Venice, Italy. According to the raw data, pilots fly passengers to Venice three times a day, seven days a week but only return on 20 occasions. In this case, the Friday afternoon flight from Venice to Amsterdam has to be added to the data.

In total 30 flight services were added, of which 14 for the Embraer (Cityhopper) group, 1 for the Airbus (A330) group, 10 for the Boeing 737 (B737) group, 2 for the Boeing 747 (B747) group and 3 for the Boeing 777 & 787 combination group.

A small but significant error was present in the data of some flights between Amsterdam and the United Kingdom, specifically with services where the flight time was shorter than the time zone difference or equal to it. In the dataset, the arrival day was set to the day following the departure day, giving it a duration of over 24 hours. To solve this problem, the arrival date was changed to be equal to the departure day in 43 services, setting the duration of these flights back to the intended 50 to 60 minutes, depending on the destination. All cases were part of the Embraer flight group.

5.2 Duty generation

Table 5.1 summarises the duties generated and shows the relation with the number of weekly flight services for the aircrafts servicing European destinations and the difficulty of combining flight services into a duty for the intercontinental services.

Table 5.1: Summary of the enumerated duties

License type	**# weekly starting duties**	**# weekly flight services**	**destination**
Embraer	479,300	2,242	Europe
A330	218	179	Intercontinental
B737	77,262	1,892	Europe
B747	134	134	Intercontinental
B777 & B787	514	494	Intercontinental

The difference between the number of duties across the two European license types is also due to the difference in the average duration of the flight services. The Embraer aircrafts operate all of KLM’s shortest routes as well as some of their longer European routes, while the B737 aircrafts are focused on operating KLM’s longer European routes. Because of this difference, Embraer-operated duties can, on average, consist of more flight services than the B737-operated

duties.

5.2.1 Clustered

Before duties can be generated for the clusters, the clusters have to be determined using the model in Chapter 4. The weights in the objective function are set as follows. The weight w_y for variable y, which affects the balance in the number of pairings across clusters, is set to 10.0. The weight w_m for variable m, which affects the balance in the number of countries, is set to 1000.0. The weight w_z for variable z, keeping the balance in the number of airports, is set to 1000.0 as well. This gives the following objective for both $i = 0$ (Embraer) and $i = 2$ (B737):

$$min \sum_{g \in G_i} \sum_{p \in P_i^2} c_p x_{pg} - 10y - 1000m - 1000z \tag{5.1}$$

Pairing cost c_p is determined by the same estimated parameters across the entire study and assumes every pairing will be flown by two airline pilots. The constant factor of the pairing, previously denoted by c_c, is set to €500, and the salary approximation for both pilots combined, previously denoted by s, is set to €2 per minute elapsed during the pairing. Lunch cost allowances c_l are estimated at €40 per lunch taking place during layovers abroad for both pilots combined, whereas this is €70 per dinner. Hotels for layovers abroad are usually booked by the airline directly and are estimated at €120 per pilot per night, totalling €240 per night for the two pilots combined. This can be summed up in pairing cost function c_p:

$$c_p = 500 + 2e_p + 40l_p + 70d_p + 240h_p \tag{5.2}$$

When any artificial flight services are present in a pairing, the cost of the flight service is calculated as given above and multiplied by factor 50 to discourage the use.

Table 5.2 below shows the main characteristics of the formed clusters and the balance attained through the model, solved by using CPLEX. In total, the model groups 2,242 flight services serving 67 airports across 17 countries for license type Embraer and 1,892 flight services serving 56 airports across 24 countries for license type B737.

Table 5.2: Summary of the results from clustering flight services

License type	cluster	# flights	# duties	# countries	# airports	initial solution	# initial pairings
Embraer	1	746	25,387	12	22	€649,810.00	373
	2	750	20,437	12	23	€842,160.00	375
	3	746	26,903	12	22	€811,410.00	373
B737	1	630	6,493	13	19	€624,590.00	315
	2	630	8,084	13	18	€808,060.00	315
	3	632	6,156	14	19	€782,990.00	316

The table shows the effectiveness of the clustering model, balancing the number of flight services, countries and airports, as well as the balance through the initial solutions attained through the use of two-flight pairings.

When airports are clustered, as explained earlier, duties are generated per cluster using fewer flight services. Table 5.1 showed a total of 479,300 duties for the first license type. The generated duties of the three clusters splitting up the flight services of the same license type sum to 72,727, a reduction of 84.8% in the total size of the duty node network. This can potentially greatly affect the different phases of the final algorithm and limit the total computation time.

A similar comparison can be made for the other European license type. The total number of duties in the full network of this license type, shown by Table 5.1, is 77,262. Summing up the number of duties present in the three cluster networks, shown in Table 5.2, results in 20,733, a reduction of 73.2% in total network size.

Table 5.2 also shows the resulting number of pairings for each cluster, which is exactly half of the number of flight services included. This is due to the design of the model, building an initial solution with solely two-flight pairings. The model does not allow the use of artificial flight services.

As discussed in Chapter 4, this study is focused on grouping flights into three clusters per license type. To be able to review the effect of clustering, it is important to analyse the change in results when increasing or decreasing the number of clusters. Table 5.3 shows the results of solving the clustering model for two and four clusters for both of the European license types.

Table 5.3: Summary of the results of clustering flight services into a different number of clusters

License type	cluster	# flights	# duties	# countries	# airports	initial solution	cluster	# flights	# duties	# countries	# airports	initial solution
Embraer	1	1120	58,918	14	34	€1,190,650.00	1	344	3,169	11	16	€521,870.00
	2	1122	80,509	14	33	€1,112,730.00	2	726	25,288	5	17	€789,600.00
							3	812	32,283	10	17	€672,090.00
							4	360	3,223	8	17	€319,820.00
B737	1	932	16,405	18	28	€1,174,760.00	1	348	2,290	12	14	€424,700.00
	2	960	15,775	18	28	€1,040,880.00	2	484	4,844	8	14	€588,390.00
							3	650	6,597	8	14	€726,820.00
							4	410	2,651	9	14	€475,730.00

Table 5.3 shows a balanced clustering for both European license types when two clusters are formed, but the total reduction in network size is smaller than when three clusters are formed.

Here, the duty network of Embraer is reduced by 70.9% in total, and the duty network of B737 is reduced by 58.3% in total, where this was 84.8% and 73.2%, respectively, when grouping the flights into three clusters. The larger networks could cause computational challenges.

When flight services are grouped into four clusters per license type, the clusters can become more unbalanced, as is shown in Table 5.3. The table shows how increasing the number of clusters by one makes it challenging to divide the flight services across the clusters without losing balance in the number of flight services, countries, airports and the initial solution value. This also leads to clusters with varying duty network sizes; however, the total network sizes are still further reduced by the addition of the fourth cluster.

5.3 European flight services

To be able to compare the clustered and full methods, the solution of license type Boeing 737 is determined as well as reasonably possible computationally. The full solution is not determined for license type Embraer due to computational reasons.

As discussed in Chapter 4, the algorithm for pairing generations of European flight services starts with generating an initial solution. Table 5.4 shows the results of the approach discussed earlier, using only two-flight pairings to provide flexibility for the algorithm later on without having to rely heavily on artificial flight services.

Table 5.4: Summary of the initial solution produced

License type	**# flights**	**# duties**	**# initial pairings**	**# artificial flights added**	**initial solution**
B737	1,892	77,262	946	0	€2,220,100.00

The results in Table 5.4 show the algorithm can be used to find an initial solution without the use of any artificial flight services.

Once the initial set of pairings is available, the main algorithm can be executed. As described earlier, the main algorithm consists of three steps, the greedy shortest-path step, the step to extract any possible remaining pairings and the branch-and-price step. To execute the first step, the setting for the benchmarks of good and bad ends, explained in Chapter 4, are 400 and 700, respectively. This means a round of shortest-path exploration starting at one starting duty node is ended when at least 400 potential negative reduced cost pairings are found or when at least 700 positive reduced cost pairings are found, whichever comes first. When neither of the benchmarks is met, the search ends when all nodes of the network under review are explored. After the exploration is ended, a maximum of 25 pairings are added to the system for each starting duty.

The greedy approach produces over 5 million pairings in the first 12 iterations of column generation, reducing the solution value by €117,741.13. This took the program approximately 18 hours and 26 minutes. Continuing beyond this point was not possible due to computational limitations and the size of the model.

5.4 European flight services clustered

For the clustered European flight services, the initial solutions were calculated earlier within the clustering model. By definition of the clustering model, artificial flight services are not needed.

The initial solution of each cluster can be used as a starting point for the first of the three steps of the algorithm. To execute the first step, benchmarks of good and bad ends need to be set, just as a maximum number of pairings to be added for every round of one starting duty. For the larger European license type, Embraer, these are 400, 700 and 25, respectively. For B737, the benchmarks are set to 300 and 600, respectively, and the maximum is set to 50. This distinction is made on the basis of experiments and on the size of the clusters' networks. A round for a larger network with more starting duties takes longer and will produce more pairings when the maximum number of pairings added per starting duty is set to the same as a smaller network. Producing a large number of pairings per round can lead to a model that is too large to be handled with ease by a computer. To avoid or at least diminish these problems, the maximum is set at a lower number for the larger license type. In order to take maximum advantage of the long column generation rounds and to reduce the number of rounds needed, the benchmarks of good and bad ends are set to a higher level for the larger instance. The results of the first step are summarised in Table 5.5, with the number of new pairings generated, the updated solution, and the difference from the initial solution discussed earlier. It also gives an overview of the computation time of the different instances. Due to computational limits, it was not possible to obtain results for clusters 2 and 3 of the Embraer license type. The execution of cluster 1 was interrupted when during column generation, a low number of high-quality pairings were found, and the solution value was stable.

Table 5.5: Summary of the results after greedy column generation

License type	cluster	# generated	# iterations	# fully selected	# part. selected	solution value	difference initial	comp. time
Embraer*	1	2,016,559	38	39	285	€499,830.52	-€149,979.48	27h 17m 18s
B737	1	800,885	31	90	131	€525,770.00	-€98,820.00	03h 47m 33s
	2	847,627	32	14	218	€674,935.00	-€133,125.00	04h 33m 49s
	3	782,826	27	40	239	€669,313.33	-€113,676.67	06h 14m 02s

(*) Interrupted during end phase for computational reasons

It can be seen in Table 5.5 step 1 of the algorithm is capable of producing a large number of negative reduced cost pairings in a limited amount of time. The table also shows the solution values are greatly decreased, and the solution has shifted to use a large number of pairings partially.

When comparing the full and clustered instances of Boeing 737, it becomes apparent the clustered version of the method can complete the method by generating over 2.4 million pairings, while the full version generates over 5 million pairings with less success in completion. The total time to complete the method for all clusters is approximately equal to the time spent in the model by the full model. It is difficult to compare any solution values since no definite result was reached by the full instance.

Step 2 extracts all remaining negative reduced cost pairings, missed by the greedy approach. When this step is fully executed, this means no more negative reduced cost pairings can be found, and the LP relaxed reduced master problem is solved to optimality. The results are shown in Table 5.6. The table includes the found solution and statistics of the execution of step 2, such as the number of new pairings generated and the number of column generation iterations, as well as the computation time. It also includes a comparison with the solution value found after the greedy approach in step 1.

Table 5.6: Summary of the results after full column generation

License type	cluster	# generated	# iterations	# fully selected	# part. selected	solution value	difference greedy	comp. time
B737	1	74,746	16	90	129	€525,770.00	-€0.00	09h 07m 50s
	2	10,145	7	15	215	€674,935.00	-€0.00	05h 48m 21s
	3	5,415	5	39	240	€669,313.33	-€0.00	02h 09m 29s

Table 5.6 shows executing step 2 does not decrease the solution value any further compared to the values obtained by step 1. This demonstrates the efficiency of the first step to reach the optimal value. Step 2 does, however, extract negative reduced cost pairings during several column generation iterations. The lack of effect of adding these pairings could be explained by the lack of flexibility inherent to the set partition model.

For the second and third clusters of license type B737, the last step is the branch-and-price approach, transforming the non-integer solution produced by the first two steps into an integer solution. For the first cluster, this is not necessary because an integer solution with the optimal solution value has been found in the process of the first two steps, consisting of 139 fully selected pairings and a solution value of €525,770.00.

The branch-and-price method is a relatively slow method, which is caused by the nature of the method, selecting and fixing one follow-on pair at a time, followed every time by the updating of the network, the model, the dual variables and one or several rounds of column

generation. Due to computational limitations, there are no full results available for the second and third clusters of license type B737.

5.5 Intercontinental flight services

Due to the lower number of flights, duties and connections to other duties, the production of the optimal pairings is more straightforward when it comes to intercontinental services, as earlier explained in Chapter 4. All possible pairings can be enumerated and placed into the set partition model as shown in Chapter 3 as variables. This model can be solved by using CPLEX. The results of the three intercontinental service groups can be found below in Table 5.7 and show results for each type were obtained in less than one second.

Table 5.7: Results intercontinental flight services

License type	**# pairings generated**	**# pairings selected**	**solution value**	**comp. time**
A330	375	77	€823,290.00	00h 00m 00.241s
B747	297	67	€640,000.00	00h 00m 00.059s
B777 & B787	1080	227	€2,573,184.00	00h 00m 00.227s

Chapter 6

Conclusions

In this study, we consider the crew pairing problem and focus on the pairings generated for airline pilots. The main model solved is a set partition problem, placing each flight service into exactly one pairing. For intercontinental services, all possible pairings can be enumerated and used directly in the model. For European services, this is not possible, and because of this, a method to generate an initial solution is used in combination with a column generation method to find negative reduced cost pairings, which could potentially improve the solution. To limit the computational time, the column generation method is initially split into two steps, a step with a greedy shortest-path method and a step with an exact shortest-path method. Once no more negative reduced cost pairings can be found, a branch-and-price algorithm is used to come to an integer solution. The branching is based on finding and fixing the best follow-on flight pair available, slowly restricting the network and the possible solutions. In between fixing pairs, column generation is used with an exact shortest-path method, and found pairings are added. This results in a constantly optimising algorithm.

To decrease the size and computation time of the European flight service networks, as well as to increase the work satisfaction and well-being of the pilots, the airline's European flight services can be clustered. Using the model presented in this study, flight services of a license type can be clustered by grouping airports. Each airport and all services flying to and from that airport are part of exactly one cluster. Using the model, it is possible to balance the number of countries and airports included across clusters, as well as the number of flight services. This ensures pilots have a diverse schedule, visit various countries and cities and operate on services of varying lengths. The model also takes into account the initial solution values of the clusters and produces an initial solution for every cluster using enumerated two-flight pairings.

The results show the clustering method is indeed successful in reducing the size of the problem instances. The total duty network of the Embraer (Cityhopper) license type can be reduced by

84.8% through the use of clustering, which affects all methods of extracting pairings from the network. The number of duties enumerated for the Boeing 737 license type can be reduced by 73.2% using clustering.

Comparing the results of the methods using the full Boeing 737 duty network with those of the clustered network of the same license type shows the effect of the reduction in network size on the computational time. Using clusters makes it possible to retrieve more results in the same computation time. Clustered models also generate higher quality pairings, reaching the optimal value in a smaller amount of time with a smaller amount of variables needed.

The results for the intercontinental flight services can be retrieved quickly and optimally using the set partition problem with enumerated pairings. It is possible to use the clustering model with these services, but it would not change the solution value, and it is not necessary for the complexity of the problem. It would also increase the computational time. If it is wished for by the pilots, clustering could be performed afterwards.

6.1 Limitations

Some limitations were present during the study. Some errors were present in the original data, and it was not always completely clear how errors had to be corrected. This makes it possible the real flight schedule used by KLM in the summer of 2018 is slightly different than the schedule used here.

Limitations were also faced during the test and computation phases. The size of the database and the density of the networks made it difficult for the computer to handle the execution of the algorithms, even with the use of SQLite databases and multi-threading.

6.2 Further research

Since the clustering model shows promising results, implementing it further, possibly with different column generation methods or into other aspects of crew scheduling, could prove worthwhile. In the crew assignment problem, pilots can be assigned to pairings created with the model, and the effects can be analysed, as well as the effects during the operational phase. A study monitoring the well-being of pilots could help analyse the effectiveness. It would also be interesting to see the effects on the schedule of an airline with multiple home bases.

Bibliography

AhmadBeygi, S., Cohn, A., & Weir, M. (2009). An integer programming approach to generating airline crew pairings. *Computers & Operations Research*, *36*(4), 1284–1298. https://doi.org/10.1016/j.cor.2008.02.001

Arayikanon, K., & Chutima, P. (2018). Solving cockpit crew scheduling problem of a low-cost airline using metaheuristics. *AIP Conference Proceedings*. https://doi.org/10.1063/1.5080055

Day, P. R., & Ryan, D. M. (1997). Flight attendant rostering for short-haul airline operations. *Operations Research*, *45*(5), 649–661. https://doi.org/10.1287/opre.45.5.649

Dück, V., Ionescu, L., Kliewer, N., & Suhl, L. (2012). Increasing stability of crew and aircraft schedules. *Transportation Research Part C: Emerging Technologies*, *20*(1), 47–61. https://doi.org/10.1016/j.trc.2011.02.009

Fahle, T., Junker, U., Karisch, S. E., Kohl, N., Sellmann, M., & Vaaben, B. (2002). Constraint programming based column generation for crew assignment. *Journal of Heuristics*, *8*(1), 59–81. https://doi.org/10.1023/a:1013613701606

Ionescu, L., & Kliewer, N. (2011). Increasing flexibility of airline crew schedules. *Procedia - Social and Behavioral Sciences*, *20*, 1019–1028. https://doi.org/10.1016/j.sbspro.2011.08.111

Lavoie, S., Minoux, M., & Odier, E. (1988). A new approach for crew pairing problems by column generation with an application to air transportation. *European Journal of Operational Research*, *35*(1), 45–58. https://doi.org/10.1016/0377-2217(88)90377-3

Makri, A., & Klabjan, D. (2004). A new pricing scheme for airline crew scheduling. *INFORMS Journal on Computing*, *16*(1), 56–67. https://doi.org/10.1287/ijoc.1020.0026

Mercier, A., Cordeau, J.-F., & Soumis, F. (2005). A computational study of benders decomposition for the integrated aircraft routing and crew scheduling problem. *Computers & Operations Research*, *32*(6), 1451–1476. https://doi.org/10.1016/j.cor.2003.11.013

Mercier, A., & Soumis, F. (2007). An integrated aircraft routing, crew scheduling and flight retiming model. *Computers & Operations Research*, *34*(8), 2251–2265. https://doi.org/10.1016/j.cor.2005.09.001

Muter, İ., Birbil, Ş. İ., Bülbül, K., Şahin, G., Yenigün, H., Taş, D., & Tüzün, D. (2013). Solving a robust airline crew pairing problem with column generation. *Computers & Operations Research*, *40*(3), 815–830. https://doi.org/10.1016/j.cor.2010.11.005

Ozdemir, H., & Mohan, C. K. (2001). Flight graph based genetic algorithm for crew scheduling in airlines. *Information Sciences*, *133*(3-4), 165–173. https://doi.org/10.1016/s0020-0255(01)00083-4

Papadakos, N. (2009). Integrated airline scheduling. *Computers & Operations Research*, *36*(1), 176–195. https://doi.org/10.1016/j.cor.2007.08.002

Quesnel, F., Desaulniers, G., & Soumis, F. (2020). A branch-and-price heuristic for the crew pairing problem with language constraints. *European Journal of Operational Research*, *283*(3), 1040–1054. https://doi.org/10.1016/j.ejor.2019.11.043

Rasmussen, M. S., Lusby, R. M., Ryan, D., & Larsen, J. (2011). Subsequence generation for the airline crew pairing problem. https://orbit.dtu.dk/en/publications/subsequence-generation-for-the-airline-crew-pairing-problem

Ryan, D. M. (1992). The solution of massive generalized set partitioning problems in aircrew rostering. *The Journal of the Operational Research Society*, *43*(5), 459. https://doi.org/10.2307/2583565

Ryan, D., & Foster, B. (1981). An integer programming approach to scheduling. *Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling*, 269–280.

Sandhu, R., & Klabjan, D. (2007). Integrated airline fleeting and crew-pairing decisions. *Operations Research*, *55*(3), 439–456. https://doi.org/10.1287/opre.1070.0395

Stojković, M., Soumis, F., & Desrosiers, J. (1998). The operational airline crew scheduling problem. *Transportation Science*, *32*(3), 232–245. https://doi.org/10.1287/trsc.32.3.232

Vance, P., Atamturk, A., Barnhart, C., Gelman, E., Johnson, E., Krishna, A., Mahidhara, D., Nemhauser, G., & Rebello, R. (1997). A heuristic branch-and-price approach for the airline crew pairing problem. *Technical Report LEC-97-06, Georgia Institute of Technology.*

Vance, P., Barnhart, C., Johnson, E. L., & Nemhauser, G. L. (1995). Airline crew scheduling: A new formulation and decomposition algorithm. *Operations Research*, *45*(2), 188–200. https://doi.org/10.1287/opre.45.2.188

Wen, X., Sun, X., Sun, Y., & Yue, X. (2021). Airline crew scheduling: Models and algorithms. *Transportation Research Part E: Logistics and Transportation Review*, *149*, 102–304. https://doi.org/10.1016/j.tre.2021.102304

Yan, S., & Chang, J.-C. (2002). Airline cockpit crew scheduling. *European Journal of Operational Research*, *136*(3), 501–511. https://doi.org/10.1016/s0377-2217(01)00060-1

Appendix

Summary of programming classes created

Constructors:

- SingleFlight - stores all necessary flight data such as license type and departure and arrival airport, day and time.
- SingleDuty - consists of a list of SingleFlight objects and can use this list to produce its start and end day, as well as its duration.
- SinglePairing - consists of a list of SingleDuty objects and can use this list to check whether a pairing is less than five days and to calculate the pairing costs by evaluating the number of lunches, dinners and hotel nights included.
- Model & ModelCluster - consists of a set of available pairings, a list of flight services to be covered, the license type and, if applicable, the name of the cluster. Builds and solves a CPLEX model, saves the model and can return any statistics on the solution that may be needed. The class also includes methods to decide on the best follow-on pair, find any other flight pairs that need to be fixed because of fully selected pairings and find any duties and pairings that need to be removed after flight pairs have been fixed.

Assisting classes:

- TablesSQL - necessary for the creation of the flight and duties databases.
- AirportTimeZones - necessary for the handling of different time zones across the flight schedule, can evaluate time zone differences and the duration between two timestamps in cities with different timezones.
- MultiThread - allows different sections of the program to be run simultaneously in order to reduce computation time.

Main classes:

- Main - runs the entire program and calls on other classes.
- Flights - reads in the initial data obtained through an API, repairs cases of duplicate and missing flight services and groups flight services into clusters. It also contains methods to fill the flight database correctly.
- Duties - enumerates duties from flight data according to regulations for all license types and finds their connecting duties. The class also contains methods to enter these duties and connections into the duty database.
- Pairings - produces the initial solutions for the full instances as well as the final solutions of all European instances, both full and clustered.
- ShortestPath - produces pairings during column generation by using a shortest-path algorithm for all three approaches - greedy, full and during truncated branch-and-price.