



DEGREE PROJECT IN MATHEMATICS,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2018

Robust airline crew pairing optimization for short-haul flight

ALEXANDRU ANDREI RADU

Robust airline crew pairing optimization for short-haul flights

ALEXANDRU ANDREI RADU

Degree Projects in Optimization and Systems Theory (30 ECTS credits)
Degree Programme in Aerospace Engineering (120 credits)
KTH Royal Institute of Technology year 2018
Supervisor at Aviolinx Software AB: Per Genell
Supervisor at KTH: Per Enqvist
Examiner at KTH: Per Enqvist

TRITA-SCI-GRU 2018:011
MAT-E 2018:03

Royal Institute of Technology
School of Engineering Sciences
KTH SCI
SE-100 44 Stockholm, Sweden
URL: www.kth.se/sci

ACKNOWLEDGEMENTS

I would like to take this opportunity to thank everyone who has supported me through my master's until the finish of my thesis. First and foremost, I would like to thank Per Genell, the head of R&D at Aviolinx, for sharing with me his skills, experience and providing me with this wonderfully analytical thesis which helped me break through into the world of operations research in aviation. I would like to express my gratitude to my examiner, Per Enqvist, associate professor – department of mathematics in KTH, who provided me unmatched support and introduced me to systems engineering. This would have been impossible without the continuous support and encouragement of my family. Thank you for providing me the will and the love I needed to keep going. I would also like to thank Lukasz Rosikiewicz, software developer at Aviolinx, and to all other people from there for sharing with me their IT skills and guiding me on the right path of programming. Finally, I would like to thank my friends whom I have met all around the world, for sharing the knowledge, tips and tricks and sharing the fun I had in learning.

TABLE OF CONTENTS

| | |
|--|------|
| ACKNOWLEDGEMENTS | iii |
| ABSTRACT | viii |
| SAMMANFATTNING | x |
| NOMENCLATURE | xi |
| 1. INTRODUCTION | 1 |
| 1.1 Airline Planning and Scheduling | 1 |
| 1.2 Crew Pairing Concept and Constraints | 2 |
| 1.2.1 Rules and Regulations | 4 |
| 1.2.2 Problem Decomposition | 5 |
| 1.2.3 Possible Approaches | 6 |
| 2. PROBLEM APPROACH | 8 |
| 2.1 Basic Problem Framework | 8 |
| 2.2 Previous Approaches | 10 |
| 2.2.1 Flight- Based Network Pairing Generation | 10 |
| 2.2.2 Duty- Based Network Pairing Generation | 11 |
| 2.2.3 Partial Pairing Generation | 11 |
| 2.2.4 Branch-and-Price | 12 |
| 2.3 Methodology Proposed | 12 |
| 3. PAIRINGS GENERATION | 14 |
| 3.1 Concept | 14 |
| 3.2 Roundtrips | 15 |
| 3.3 Network | 16 |
| 3.3.1 Flight-Based Network | 17 |
| 3.3.2 Pairing-Based Network | 19 |
| 3.4 Searching Algorithm | 21 |
| 3.4.1 Searching on a Flight Based Network | 22 |
| 3.4.2 Searching on a pairing-based network | 25 |
| 4. OPTIMIZATION MODEL | 26 |
| 4.1 Problem Formulation | 26 |

| | |
|---|----|
| 4.2 Overcover Penalty | 29 |
| 4.3 Robustness Penalty..... | 33 |
| 4.4 Integrated Optimization Model | 39 |
| 5. SIMULATIONS | 41 |
| 6. CONCLUSIONS | 43 |
| REFERENCES | 44 |

ABSTRACT

Crew costs are the second highest costs for airlines therefore they represent a key factor for an airline survival and crew scheduling is one of the hardest combinatorial problem. The scheduling process is broken down into crew pairing and crew rostering and, in this thesis, a robust solution is described in detail for the former one.

The purpose of the thesis is to present an efficient and robust crew pairing optimization tool which minimizes the pairings costs and reduces unnecessary overcovers. The model framework is based on a new concept which involves four stages. During the first stage all roundtrip combinations are generated then in the second stage the roundtrips generated are optimized and the optimal solution is used in the third stage to generate all pairing combinations. And the last one, the fourth stage, optimizes the pairings obtained from the third stage.

An augmented set covering problem is used to for the problem formulation where the unknown variables can take just integer values. A mixed integer programming solver from Google OR has been used to solve the optimization problem.

In the last chapter numerical results are presented which show the efficiency of using this model framework.

ROBUST CREW PAIRING OPTIMERING FÖR FLYGBOLAG PÅ KORTDISTANS FLYG

SAMMANFATTNING

Besättningskostnader är den näst största kostnadsposten för ett flygbolag. De spelar därmed en nyckelroll i ett flygbolags överlevnad. Schemaläggning för besättning är ett mycket svårt kombinatoriskt problem.

Schemaläggningsprocessen är indelad i två delmoment: crew pairing och crew rostering. I detta arbete presenteras en robust lösning på det tidigare problemet.

Syftet med rapporten är att presentera ett effektivt och robust optimeringsvektyg för att minimera kostnaderna för pairingar och minska ickenödvändig övertäckning.

Ramverket för modellen är baserat på ett nytt koncept vilket involverar fyra steg.

I första steget skapas pairingar som rundresor, dvs. de slutar så snart en flight i pairingen når flygplatsen som pairingen började på. I det andra steget löses ett optimeringsproblem för att hitta den optimala kombinationen av dessa rundresor,

därefter genereras pairingar på nytt i det tredje steget. I detta steg genereras pairingar baserade på lösningen i det förra steget.

Slutligen i det fjärde steget erhålles en optimal lösning baserat på en optimeringsmodell som använder sig av pairingar från det tredje steget,

Optimeringsproblemet är formulerat som ett utvidgat övertäckningsproblem där variablerna enbart kan anta heltalsvärden, och en heltalslösare från Google OR tools används för att lösa detta problem.

I det sista kapitlet presenteras numeriska resultat från modellen.

NOMENCLATURE

Leg - flight

Non-stop flight – a flight between two airports which does not touch the ground until the destination.

crew complement – crew configuration needed on a flight.

duty - any task that a crew member performs for the operator, including flight duty, administrative work, giving or receiving training and checking, positioning, and some elements of standby [1];

duty period - a period which starts when a crew member is required by an operator to report for or to commence a duty and ends when that person is free of all duties, including post-flight duty [1];

flight duty period (FDP) - a period that commences when a crew member is required to report for duty, which includes a sector or a series of sectors, and finishes when the aircraft finally comes to rest and the engines are shut down, at the end of the last sector on which the crew member acts as an operating crew member [1];

rest facility - a bunk or seat with leg and foot support suitable for crew members' sleeping on board an aircraft [1];

positioning - means the transferring of a non-operating crew member from one place to another, at the behest of the operator [1];

home base - the location, assigned by the operator to the crew member, from where the crew member normally starts and ends a duty period or a series of duty periods and where, under normal circumstances, the operator is not responsible for the accommodation of the crew member concerned [1];

acclimatized - a state in which a crew member's circadian biological clock is synchronized to the time zone where the crew member is. A crew member is considered to be acclimatized to a 2-hour wide time zone surrounding the local time at the point of departure. When the local time at the place where a duty commences differs by more than 2 hours from the local time at the place where the next duty starts [1];

RMP – restricted master problem

1. INTRODUCTION

1.1 Airline Planning and Scheduling

Airline business has greatly evolved in the last decades and automatically implied more and more complex planning and scheduling requirements. As it is a highly competitive industry the cost of operations is a key factor of this business and most of the airlines are using operations research techniques, which have been used in this field since 1950s [1], to survive and to make sure that their resources meet the demands and are used in an efficient way.

The number of variables in this process is gigantic and not even the computational power in these days is able to solve such large problems. Therefore, due to its complexity, the planning process is divided into multiple stages and usually airlines have departments assigned for each stage. There are four main stages and they have a logical sequence being approached (see Fig. 1) as the result of some stage is dependent on the data provided from a previous stage.

The first stage is the Flight Scheduling. Here, the answers of two questions are sought; “where to fly?” and “when to fly?”. In this process, the flight network is created where the destinations to fly to and the time at which the flight should take place are decided. These decisions are usually influenced by many factors. Some of them are market demand forecast, types of fleets, number of aircraft, benchmarks etc. [2].

The timetable is created with all the destinations and the times of each flight but there is no information regarding the fleets which will be flying these legs (nonstop flights). This stage is known as the Fleet Assignment. The purpose of the fleet assignment is to assign as many flights as possible to the right fleet (not to be confused with the fleet planning, where the number of the aircraft to be purchased is decided). The airlines which operate multiple fleets must take into considerations different characteristics of each fleet such as the cost of operating them, maintenance required and maintenance cost, fuel information, seating capacity etc. [2].

Now, with the solution from the previous step, the problem can be divided by fleet and the next stage is taking place, the aircraft routing, where each leg from the network

must be assigned to a specific aircraft. Also known as the aircraft rotation or tail assignment, this stage aims at reducing the operating cost by assigning each aircraft within a fleet to a specific set of legs. In this process each flight must be covered by one aircraft, a balance utilization load is required for each aircraft and the required maintenance must be assured as well [2].

The fourth main stage is the crew scheduling. Each flight has a crew complement and the aim of this process is to satisfy the required crew complement for each leg by minimizing the operating cost or maximizing the crew utilization. As the crew cost is one of the largest costs, this stage has been deeply researched both by academia and industry.

| <i>Airline</i> | <i>Crew</i> | <i>Fuel</i> | <i>Maintenance</i> | <i>Ownership</i> | <i>Total</i> |
|-----------------------|--------------------|--------------------|---------------------------|-------------------------|---------------------|
| <i>Continental</i> | \$510 | \$430 | \$651 | \$698 | \$2,291 |
| <i>United</i> | \$927 | \$487 | \$1048 | \$510 | \$2,974 |
| <i>Southwest</i> | \$388 | \$537 | \$251 | \$350 | \$1,526 |

Example: Boeing 737-500 flight operating cost per block hour. Source: ICAO

Since this is amongst the most computational intensive combinatorial problems [2], crew scheduling is divided into two subproblems: crew pairing optimization and crew rostering optimization. The reason for this is to reduce the size of the problem by first creating the pairings to cover all the flight legs and then assigning them to the crew. Finding an optimal solution to the former subproblem, crew pairing optimization, is the aim of this thesis.



Figure 1: Main stages in airline resource planning

1.2 Crew Pairing Concept and Constraints

Crew pairing is the impersonal phase of the crew scheduling process where we design all the routes to be flown by a crew, but we don't know which crew will be flying them. As mentioned before, the main idea under this concept is to reduce the size of the

problem. Therefore, a definition of the crew pairing can sound like this: *Airline crew pairing is a set of flight legs within the same fleet or fleet family flown by an unknown crew, which ends at the same crew base it started.*

Airlines can have, and usually do have, multiple crew bases. These crew bases are locations assigned by the operator. The operator is not responsible of the accommodation of the crews when they are at the base they have started from [3] that is why, normally, the crews are living at one of these bases. Therefore, the reason of bringing the crew back to the same base is because we want to bring them back home.

A pairing must contain at least two sectors to satisfy the round-trip requirement. The time between the sectors within a pairing is called *sit connection* and together create a Flight Duty Period (FDP). The sit connection is when the crew is waiting to board for the next flight. Usually the sit connection can't be more than couple of hours, as the crew is waiting in the airport. The maximum time of a sit connection is mainly decided by the operator and if it becomes too high the crew must be provided with a rest facility. The FDP is part of a duty period as the crew is required to perform some activities other than the ones inside the FDP. At this point we have described a pairing which is composed just of one duty period.

There are many rules and regulations imposed on all the concepts above, both from the governmental regulations and collective agreements. As there are maximum values for FDP or duty period the crew might be in the situation where it has exhausted all their allowed working time, but it is located at a base other than the home base. In this case the operator must provide the crew with an overnight accommodation. This time is called *layover*. If there is a layover the crew will automatically work two duty periods. As stated above that a pairing must end at the same base it has started it means that the two duty periods will be in the same pairing. Therefore, a pairing can contain two or multiple duty periods if there are layovers to separate them.

Figure 2 shows a two duty periods pairing and each of them contains three flights inside. The overnight rest/layover can be seen in between duties.

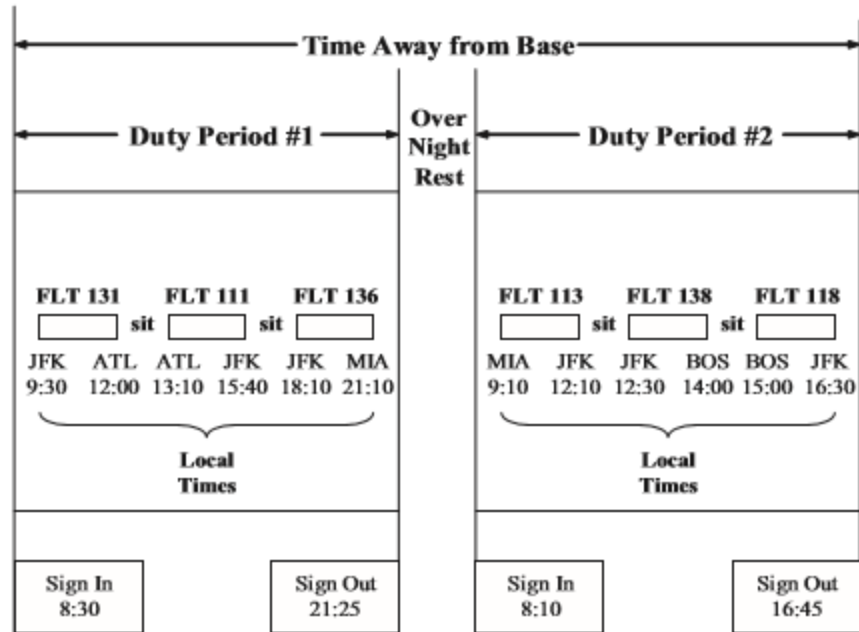


Figure 2: Pairing with two duty periods and one layover/overnight rest. Pairing starts and ends at JFK.
Source: [2]

1.2.1 Rules and Regulations

Aviation sector has many rules and regulations due to the high requirements for safety. For the case of the crew, the rules and regulations often become complex both because of safety and working regulations. All these restrictions on pairings usually come from different sources. From EASA in Europe or FAA in USA and from unions and operators as well. Therefore, the task of creating pairings becomes very difficult as the rules are very complex.

Defining all the rules is out of scope for this thesis hence this matter is briefly explained under this heading. A simple FDP limit rule from EASA can be seen in Figure 3.

The crew pairing optimization tool described in this paper is using RAIDO's legalities engine called Mimer. *RAIDO is an aviation management system which puts the user in complete control of all strategic, financial and operational business processes* [4]. After the pairing generation phase is done, as explained more in Chapter 3, Mimer is checking all the pairings to decide if they are legal or not. A legal pairing varies a lot, as different operators have different rules; the same with the union regulations. The rules can

be further broken down to the crew type level. For example, for pilots some rules are different compared to the ones for cabin crews.

| Start of FDP at reference time | Sectors | | | | | | | | |
|--------------------------------|---------|-------|-------|-------|-------|-------|-------|------|------|
| | 1 – 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 06:00 – 13:29 | 13:00 | 12:30 | 12:00 | 11:30 | 11:00 | 10:30 | 10:00 | 9:30 | 9:00 |
| 13:30 – 13:59 | 12:45 | 12:15 | 11:45 | 11:15 | 10:45 | 10:15 | 9:45 | 9:15 | 9:00 |
| 14:00 – 14:29 | 12:30 | 12:00 | 11:30 | 11:00 | 10:30 | 10:00 | 9:30 | 9:00 | 9:00 |
| 14:30 – 14:59 | 12:15 | 11:45 | 11:15 | 10:45 | 10:15 | 9:45 | 9:15 | 9:00 | 9:00 |
| 15:00 – 15:29 | 12:00 | 11:30 | 11:00 | 10:30 | 10:00 | 9:30 | 9:00 | 9:00 | 9:00 |
| 15:30 – 15:59 | 11:45 | 11:15 | 10:45 | 10:15 | 9:45 | 9:15 | 9:00 | 9:00 | 9:00 |
| 16:00 – 16:29 | 11:30 | 11:00 | 10:30 | 10:00 | 9:30 | 9:00 | 9:00 | 9:00 | 9:00 |
| 16:30 – 16:59 | 11:15 | 10:45 | 10:15 | 9:45 | 9:15 | 9:00 | 9:00 | 9:00 | 9:00 |
| 17:00 – 04:59 | 11:00 | 10:30 | 10:00 | 9:30 | 9:00 | 9:00 | 9:00 | 9:00 | 9:00 |
| 05:00 – 05:14 | 12:00 | 11:30 | 11:00 | 10:30 | 10:00 | 9:30 | 9:00 | 9:00 | 9:00 |
| 05:15 – 05:29 | 12:15 | 11:45 | 11:15 | 10:45 | 10:15 | 9:45 | 9:15 | 9:00 | 9:00 |
| 05:30 – 05:44 | 12:30 | 12:00 | 11:30 | 11:00 | 10:30 | 10:00 | 9:30 | 9:00 | 9:00 |
| 05:45 – 05:59 | 12:45 | 12:15 | 11:45 | 11:15 | 10:45 | 10:15 | 9:45 | 9:15 | 9:00 |

Figure 3: Maximum daily FDP – acclimatized crew members. Source [3]

1.2.2 Problem Decomposition

The crew is usually licensed on a single aircraft type or a aircraft family therefore the problem is decomposed by fleet or aircraft type. Also, the crew category must be taken into consideration as some of the regulations are different based on the crew category. This leads to a further decomposition of the problem, the crew category decomposition.

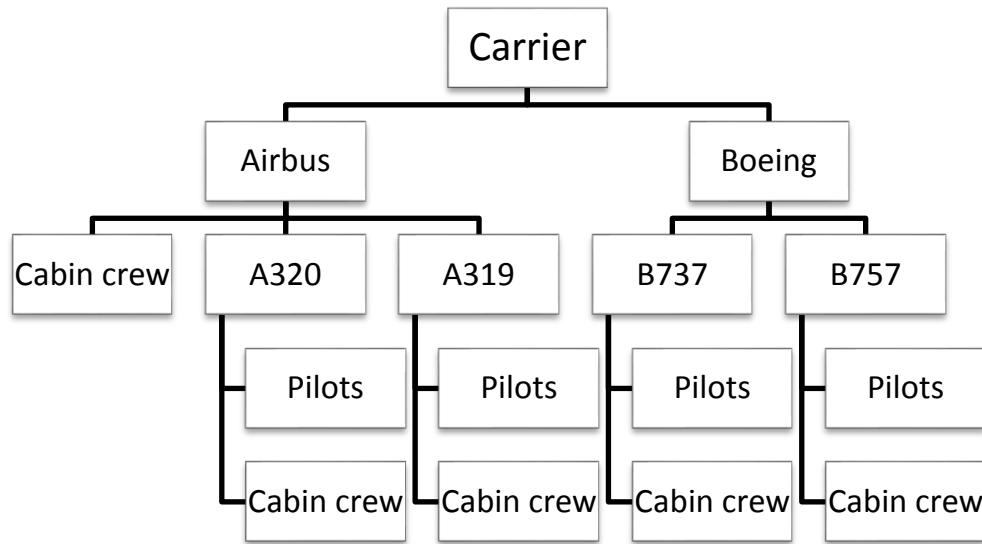


Figure 4: Example of crew pairing problem decomposition

It can be seen from Figure 4 that optimizing the crew pairings over the entire schedule has to be done separately for each fleet and crew category. It is important for the data not to be mixed up as this can lead to unnecessary overcovered legs and illegal pairings. Here the RAIDO filters are used to decompose the schedule correctly. The filters can be created by the user and saved as templates into a database.

1.2.3 Possible Approaches

Section 1.2.2 described how the problem must be decomposed and the reason for that was the licensing regulations. But it is important to mention that the fleet decomposition reduces the search area in both the pairing generation and optimization phases, which will be introduced in the next chapters.

As the crew scheduling is one of the most intensive combinatorial problem one should understand that any method that will reduce the search area might have a big impact on the overall computational time.

This section describes three different approaches on the crew pairing problem, the daily problem, the weekly problem and the full dated problem. The reason for having these approaches is that the first two can greatly reduce the search area and provide the same solution as using the fully dated approach.

The daily solution assumes that the schedule within a timespan is daily repetitive with some exceptions during the weekends. A solution for an arbitrary day is calculated and repeated until the schedule is fully covered. The pairings here can't be longer than a day. It is also important not to get overcovers because of the reduced search area. This type of approach does not work in most of the situations as operators don't have a daily repetitive schedule.

The weekly approach considers the schedule weekly repetitive within a timespan. The pairings here can be as long as a week. A solution for an arbitrary week can be generated and used for the rest of the weeks. This approach compared with the daily approach is better as layovers are allowed within a week. But the search area is bigger which leads to more computational time required. Here, it is important that all the legs within a week must be covered. The weekly approach is more common compared to the daily one as some operators have weekly repetitive schedule.

The fully dated approach is considering no repetitive schedule. This solution is the preferred one in terms of optimal solution, but it is expensive in terms of computational power required. Usually, an entire month is loaded and solved.

The crew pairing optimization model presented in this paper can use all the approaches from this section.

2. PROBLEM APPROACH

2.1 Basic Problem Framework

Usually airline crew pairing optimization is formulated as a set partitioning problem or set covering problem where a subset of feasible pairings which minimizes the total cost by covering all the flights is sought [4].

$$\min \sum_{j \in J} c_j x_j \quad (1)$$

$$s. t. \sum_{j \in J} a_{ij} x_j = 1 \quad \forall i \in I \quad (2)$$

$$x_j \in \{0,1\} \quad (3)$$

J – set of pairings

c_j – cost of pairing j

x_j – decision variable which is an integer and it is 1 if pairing j is part of the optimal solution and 0 otherwise.

I – set of flights

a_{ij} – binary constraints coefficient matrix. Rows represent the flights and columns represent the pairings. If flight i is part of the pairing j , $a_{ij} = 1$ otherwise is 0.

The set partitioning model above reflects a basic optimization model for the crew pairing problem. The basis of the optimization model used in this paper is an extension from the one above. The constraint of this model is basically allowing one coverage per flight. The inputs for this are the flight schedule, the pairings and the costs. After the schedule is loaded into RAIDO, a functionality of selecting the timespan is available. Based on the schedule within the time horizon selected we start creating the pairings. All the connections between the flights are stored in a network structure and a depth-first search algorithm is used to search and return all the feasible pairings. Now, after the pairings have

been created, the costs are calculated, and the model is ready for being optimized with a branch-and-bound solver.

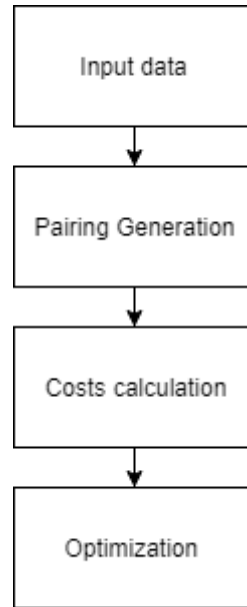


Figure 5: Basic problem framework

Solving this crew pairing model is difficult. First, even generating pairings for a small schedule is usually very difficult, given the wide array of legalities that must be enforced. The number of pairings is often more than millions for a small schedule and this leads to difficulties in the optimization phase, as the number of variables is equal to the number of pairings. Second, having the constraint that the solution must be integer, further complicates the process [4].

There are three reasons for which these problems are difficult. First, as it is a combinatorial problem, even finding combinations which are feasible can be non-trivial. Second, due to the huge number of variables, it makes it hard for any optimization algorithm to provide optimal or near optimal solutions. And third, all the variables are integer which complicates the process even more.

2.2 Previous Approaches

The methodology from Figure 5 can work for small airlines but it will be infeasible for big carriers. Excluding the big CPU time required to get a solution, a normal computer will run out of memory before generating all feasible pairings (see Table 1). The larger the number of pairings the larger the columns set for the optimization problem. The more columns the constraints matrix has the more CPU time is required.

| Flights | Time horizon | No. of bases | RAM memory | Processor | CPU Time until crash | Pairings generated until crash |
|---------|--------------|--------------|------------|--------------------------------------|----------------------|--------------------------------|
| 6,000 | 30 days | 6 | 16 GB | Inter Core i7-6600U CPU: 2.80 GHz | 4 hours | ~ 20 million |

Table 1: Example of a solution attempt with a basic methodology

Table 1 shows an example where multiple constraints had been applied on the pairing generation algorithm. If the searching algorithm was not constrained the basic approach wouldn't have been feasible not even for small carriers. These constraints have been applied in previous research work as well and will be discussed in more details under chapter 3.

Different methods have been studied where feasible solutions can be generated for big carriers. We will briefly discuss a couple of them.

2.2.1 Flight- Based Network Pairing Generation

This approach uses a flight network. Flights are represented by nodes and the connections between flights are represented by links. If the departure of a flight is the same with the arrival of another flight and the connection is possible (the time difference between the arrival of the first flight and the departure of the second flight is greater than a specified minimum connection time) a connection between them can be created. After all the connections have been created a depth-first search algorithm is used to generate all the legal pairings. An experienced user can create many constraints on the searching algorithm where useless pairings are pruned to be generated. This will greatly improve the

computational time. Constraints can be also enforced when creating the links. All the constraints enforced on this approach should not affect the final solution and if that is not the case then we must refer to the partial pairing generation.

After all the pairings are generated a branch-and-bound algorithm can be used to solve the optimization problem. For complex networks with millions of pairings generated it can be difficult to solve the optimization problem. Therefore, different heuristic methods, out of scope for this paper, can be used to get a close to optimal solution.

2.2.2 Duty- Based Network Pairing Generation

Pairings can also be generated based on duties (see [5], [6], [7]). First, all duties are created based on a flight network. A duty does not necessary have to end or start at a crew base. In this way all the schedule can be covered. After the duties are generated based on a depth-first search algorithm, they can be used for generating pairings.

This approach together with the pairing generation based on flight network should generate the same pairings in the end. The advantage is that, duty network based pairing generation, is faster in terms of computational time.

A branch-and-bound algorithm is used here as well to optimize the problem with the generated pairings as unknown variable.

2.2.3 Partial Pairing Generation

This approach has been described in [7] and focuses on generating just a subset of all possible pairings. It is better in terms of computational time and memory needed compared to a complete pairing generation as it allows just a limited number of possible connections between flights to be created. This will reduce the complexity of the network and implicitly will reduce the number of generated pairings. On the other hand, there is the risk that the flight schedule will not be fully covered, and the final solution will be far from optimality. An experienced user might be able to cut “bad” connections to prune the algorithm from creating pairings which won’t be used in the final solution.

This approach uses a flight network where nodes are represented by flights and links by connections.

A branch-and-bound algorithm is then used to find the optimal solution for the generated pairings.

2.2.4 Branch-and-Price

A state-of-the-art solution based on the literature (see [6]) for solving crew pairings for airline industry is column generation combined with branch-and-bound for obtaining an integer solution. Columns are generated at each node of the branch-and-bound tree to improve the LP relaxation.


Variables are generated based on the reduced cost and introduced into the restricted master problem (RMP) until no pairings with reduced cost can be generated. For large schedules it is time consuming or impossible to generate all reduced costs columns to find which one has the lowest reduced cost. There are different heuristic algorithms to find good column for the RMP. One of them is label-pulling or label-setting algorithm which is said to be one of the most efficient and it is described in [9].

The RMP needs an initial solution to start. The easiest way is to provide slack variables with high penalty costs which will be eliminated during the iterations.

The advantage of this approach is that it uses less memory which does not increase with the running time as there is a maximum number of possible pairings which can be saved at each iteration. But when it comes about the computational time it has a big drawback as it must check if a pairing is legal for each label.

2.3 Methodology Proposed

The methodology proposed in this section is meant to improve both the pairing generation and optimization phases.

For the pairing generation phases a constrained depth-first search algorithm for short pairings and another one for long pairings are proposed and integrated accordingly to the framework from  Figure 6 will reduce the number of variables used in the optimization problem.

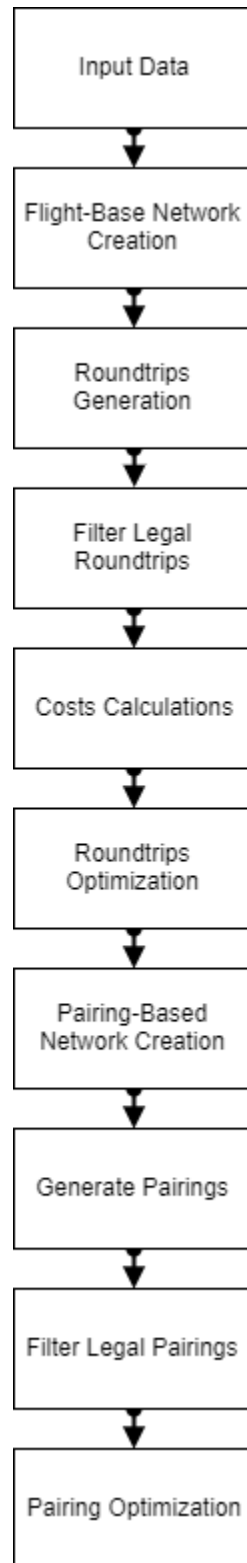


Figure 6: Framework proposed

3. PAIRINGS GENERATION

3.1 Concept

The optimization model searches to find an optimal pairing solution, therefore one of the model's inputs are the pairings. The pairings along with the flight schedule create the constrained matrix from Eq (2). The rows represent the flights and the columns represent the pairings.

In the pairing generation phase, all the feasible pairings are sought and given to the optimization model as columns for the constraint matrix. The input for the pairing generation is the flight schedule. When generating pairings, one should cover the entire given schedule. In previous approaches on pairing generation, solutions are given just for schedules where the first and last flights in a pairing must have the same base. When a schedule is loaded for the crew pairing generation stage, not all the flights which don't start at a base have a previous connection and not all the flights which don't end at a base have a successor connection. These flights are called *carry in* and *carry out* flights and, in this paper, we present covering solution for these types of flights as well. An example of a minimal flight schedule with carry in and carry out flights can be seen in Figure 7. There is one base with the short code ABZ but it can be seen that flight 397 starting from EDI has no previous connection to ABZ and flight 357 ending at LSI has no successor connection to ABZ neither. Therefore flight 397 is called carry in and flight 357 is called carry out.



Figure 7: Example of schedule with carry in and carry out concept where ABZ is a base. The rectangles represent flights and the numbers inside them represent the tail number. The three letter codes represent the airports.

The methodology framework uses two types of network, Flight based network and pairing based network.

3.2 Roundtrips

How pairings are generated is an important factor for any airline crew pairing optimization tool as this can lead to high computational time. Both the searching algorithm and the number of pairings generated are a key factor not just for the computational time of the pairing generation stage but also for the pairing optimization one.

One of the main goals of this paper is to provide a solution for an efficient tool. The concept behind the roundtrips fulfils this goal by improving the computational time of creating pairings and decreasing the number of generated pairings. It is important to mention that from the operational point of view the final solution is not affected if we compare it with the traditional pairing generation concept presented in other papers (see [10], [11]) and which will be also described below.

| Flights | Departure | Arrival | Departure Time | Arrival Time |
|----------------|------------------|----------------|-----------------------|---------------------|
| FL1 | Stockholm | Oslo | 0800 | 1000 |
| FL2 | Oslo | Copenhagen | 1100 | 1300 |
| FL3 | Copenhagen | Stockholm | 1330 | 1530 |
| FL4 | Stockholm | Oslo | 1600 | 1800 |
| FL5 | Oslo | Stockholm | 1830 | 2030 |

Table 2 : Example of a simple flight schedule to emphasis the difference between the traditional pairing creating and roundtrips creation

Considering the schedule from Table 2 where Stockholm is a base. If one would create all the possible pairings to be used as variables for the optimization problem with the traditional approach one would get the pairings from table below.

| | |
|----|-----------------------------|
| P1 | FL1 - FL2 - FL3 |
| P2 | FL1 - FL2 - FL3 - FL4 - FL5 |
| P3 | FL4 - FL5 |

Table 3: Pairing creation based on the schedule from Table 2

Pairing P2 from Table 3 touches the base in between. It starts at Stockholm and reaches Stockholm again with FL3 after which continues to F14 and then ends at FL5 when the pairing reaches the base from where it left. It can be noticed that an optimal solution for the schedule above, if the cost would increase with the number of variables in the solution, is to use just P2 as just one variable is needed.

Now, let us describe the simple idea behind the roundtrips. Say that the searching algorithm which generates pairings is constrained such that it can't generate pairings which touch the base at the arrival destination if that is not the final flight in the pairing. Using the same schedule as above, suppose one uses a search algorithm and constraints the generation as it has been described above. The generated pairings solution will be the same as the one from Table 3 but one would not generate P2.

Indeed, as mentioned before P2 is the optimal solution found. But from the operational perspective using P1 and P3 instead of P2 is the same because if one merges P1 and P3 one will end up with the same pairing as P2.

Generating roundtrips will reduce the search area for the depth-first search algorithm used in this paper and it will also reduce the number of pairings generated after which a new process will take place which will be using the optimal solution from roundtrips to create the optimal pairings.

3.3 Network

The network concept is used to represent the flight schedule of a carrier making it easier for different searching algorithms to be applied. Often, these networks have a huge number of connections even for small carriers, many of them being useless. To reduce the searching space different link constraints will be applied to eliminate "bad" connections. In this paper all the constraints presented are general and can be applied for most of the carriers. But there can be airline-specific constraints as well which can reduce the complexity of the network even more, making it more efficient for the searching algorithms to find paths.

3.3.1 Flight-Based Network

This network is used by the depth-first search algorithm to generate the roundtrips described in Section 3.2. Nodes are represented by flights and links are represented by flight connections. If the arrival of a flight matches the departure of another flight and the time difference between the departure of the second flight and the arrival of the first flight is positive, then a link between those two flights can be created. Each flight starting from one of the bases or if it is a carry in flight represent a source node and each flight ending at one of the bases or if it is a carry out flight represent a sink node in the network.

| Flights | Departure | Arrival | Departure Time | Arrival Time | Day | Type of Flight | Tails |
|---------|------------|------------|----------------|--------------|-----|----------------|-------|
| F1 | Stockholm | Oslo | 0800 | 1000 | 1 | Source | T1 |
| F2 | Bucharest | Helsinki | 0700 | 1100 | 1 | Carry In | T2 |
| F3 | Oslo | Copenhagen | 1100 | 1300 | 1 | Node | T3 |
| F4 | Oslo | Helsinki | 1100 | 1400 | 1 | Node | T1 |
| F5 | Helsinki | Copenhagen | 1500 | 1700 | 1 | Node | T1 |
| F6 | Helsinki | Stockholm | 1600 | 1700 | 1 | Sink | T2 |
| F7 | Copenhagen | Stockholm | 1800 | 2000 | 1 | Sink | T3 |
| F8 | Stockholm | Oslo | 0800 | 1000 | 2 | Source | T3 |
| F9 | Oslo | Stockholm | 1100 | 1300 | 2 | Sink | T3 |
| F10 | Copenhagen | Madrid | 1200 | 1400 | 2 | Carry Out | T1 |

Table 4: An instance of a simple schedule where Stockholm is the base

| Links | |
|-------|------------|
| F1 | F3, F4, F9 |
| F2 | F5, F6 |
| F3 | F7, F10 |
| F4 | F5, F6 |
| F5 | F7, F10 |
| F6 | F8 |
| F7 | F8 |
| F8 | F9 |
| F9 | |
| F10 | |

Table 5: Links based on the nodes from Table 4

Table 5 shows all possible links of the schedule from Table 4. Based on the links the network from Figure 8 is created and a search algorithm will be applied to generate all possible short pairings described above.

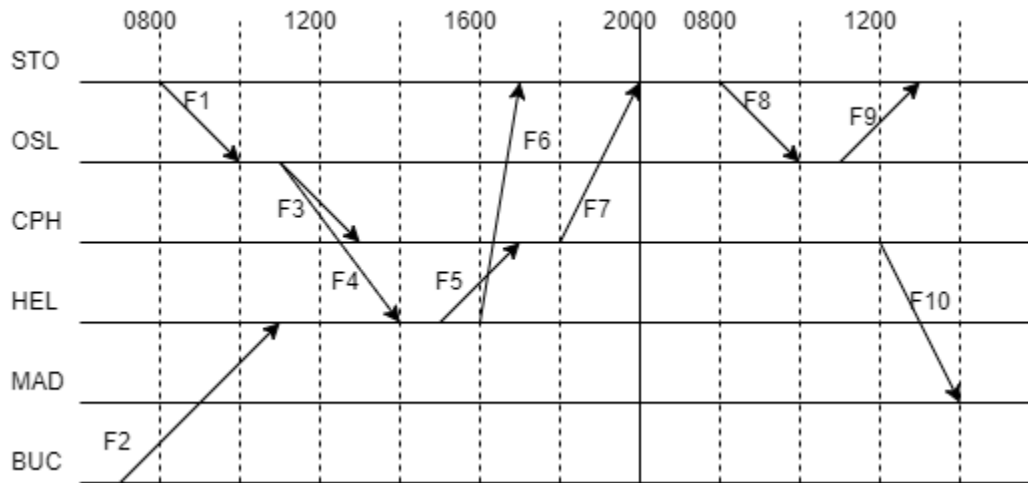


Figure 8: Timeline flight network based on the network from Table 5

Airlines usually have a huge number of links and the network can become more complex than one could handle. Therefore, different link constraints are applied to reduce the number of links but not to affect the optimal solution.

The following link constraints are used in this paper:

- Minimum transit time
- Maximum transit time
- Minimum layover time
- Maximum layover time
- Maximum duty time
- Maximum pairing time

The values of the constraints differ from a carrier to another. Let us assume:

- Minimum transit time = 15 minutes
- Maximum transit time = 5 hours
- Minimum layover = 8 hours
- Maximum layover = 24 hours

- Maximum duty time = 13 hours
- Maximum pairing timespan = 2 days

Using the values from above, the new links of the schedule from Table 4 can be seen in Table 6. The link from F1 to F9 has disappeared as the maximum layover time is violated.

| Links | |
|-------|---------|
| F1 | F3, F4 |
| F2 | F5, F6 |
| F3 | F7, F10 |
| F4 | F5, F6 |
| F5 | F7, F10 |
| F6 | F8 |
| F7 | F8 |
| F8 | F9 |
| F9 | |
| F10 | |

Table 6: Flight links with constraints

3.3.2 Pairing-Based Network

In a pairing based network, the nodes are represented by pairings or duties and the links by the connections between them. The input given to this network is the optimal solution from the roundtrips. A depth-first search algorithm will be applied here as well, but this time longer pairings will be created. The length of a pairing depends on the user preference.

A connection between two nodes of this network is created in the same way as the one from the flight-based network. If the arrival of the last flight in a pairing matches the departure of the first flight in another pairing and the difference between the departure time of the first flight from the second pairing and the arrival time of the last flight of the first pairing is positive, and all the constraints are satisfied, then a link can be created between these two pairings.

| Pairings | Flights | Departure | Arrival | Departure Time | Arrival Time |
|----------|-------------|-----------|-----------|----------------|--------------|
| P1 | F1-F3-F7 | Stockholm | Stockholm | 0800 (day 1) | 2000 (day 1) |
| P2 | F1-F4-F5-F7 | Stockholm | Stockholm | 0800 (day 1) | 2000 (day 1) |
| P3 | F1-F4-F6 | Stockholm | Stockholm | 0800 (day 1) | 1700 (day 1) |
| P4 | F1-F3-F10 | Stockholm | Madrid | 0800 (day 1) | 1400 (day 2) |

| | | | | | |
|----|--------------|-----------|-----------|--------------|--------------|
| P5 | F1-F4-F5-F10 | Stockholm | Madrid | 0800 (day 1) | 1400 (day 2) |
| P6 | F2-F6 | Bucharest | Stockholm | 0700 (day 1) | 1700 (day 1) |
| P7 | F2-F5-F7 | Bucharest | Stockholm | 0700 (day 1) | 2000 (day 1) |
| P8 | F2-F5-F10 | Bucharest | Madrid | 0700 (day 1) | 1400 (day 2) |
| P9 | F8-F9 | Stockholm | Stockholm | 0800 (day 2) | 1300 (day 2) |

Table 7: Input for the pairing-based network

| Links | |
|-------|----|
| P1 | P9 |
| P2 | P9 |
| P3 | P9 |
| P4 | |
| P5 | |
| P6 | P9 |
| P7 | P9 |
| P8 | |
| P9 | |

Table 8: Pairing links

The same constraints as the ones described in 3.3.1 are applied to the pairing links created in Table 8.

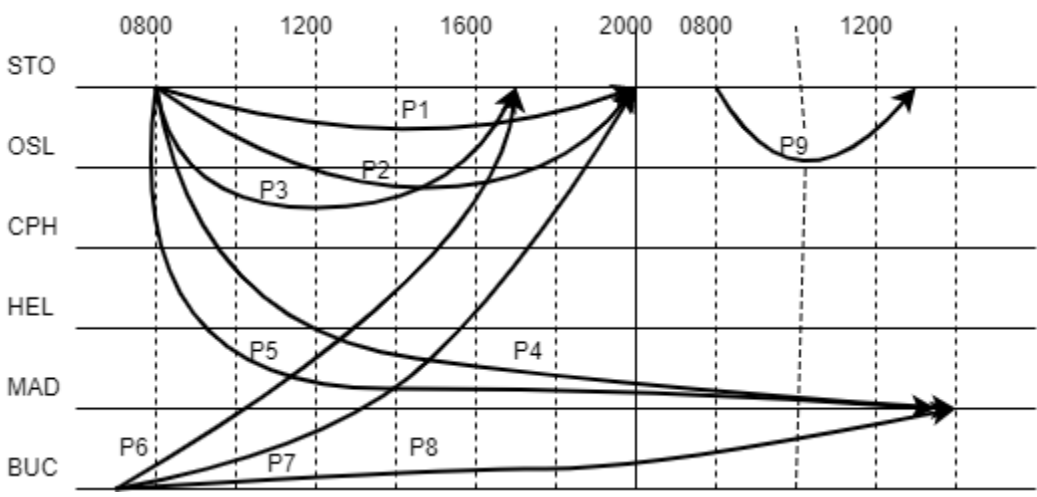


Figure 9: Timeline pairing network

In Figure 9 it can be noticed that the complexity of the network has reduced as less possible connections from a node to another exist. If just a flight-based network would be

used, and the searching algorithm would be applied such that a pairing can touch base multiple times, then a larger search area would have been used compared to the search area used in the approach presented here.

The purpose of the pairing base network is not to create a network with the pairings generated in the first stage but to create the network with the optimal solution of those pairings.

3.4 Searching Algorithm

The searching algorithm used in this paper to find all possible paths inside the network is the depth-first search (see [12]).

Given a network all possible paths which satisfy different pairing constraints will be created between all pair nodes. Pair nodes are created between two flights or pairings which create a round trip starting from the base, starting with a carry in flight and ending at any base. If there are carry out flights in the network then one should add an extra set of flights in the schedule starting after the arrival of the last flight in the current schedule, so there can exist round trips for carry out activities as well.

| Pair Flights | Comments |
|--------------|----------|
| F1 - F6 | - |
| F1 - F7 | - |
| F1 - F9 | - |
| F8 - F9 | - |
| F2 - F7 | Carry in |
| F2 - F9 | Carry in |

Table 9: Example of pair flights based on Table 4

| Pair Pairings | Comments |
|---------------|----------|
| P1 - P9 | - |
| P2 - P9 | - |
| P3 - P9 | - |
| P6 - P9 | Carry in |
| P7 - P9 | Carry in |

Table 10: Example of pair pairings based on the network from Figure 9

3.4.1 Searching on a Flight Based Network

Let us explain the way in which the depth-first search algorithm is implemented for a flight-based network and let us also see the difference of creating roundtrips compared to pairings.

When applying the searching algorithm, before a new flight is added to the partial path a set of constraints must be checked. All constraints must be satisfied for a flight to be added to a path. The reason behind it is to reduce the search area and implicitly the number of pairings generated by avoiding useless nodes.

The constraints are allocated to two different categories. One category contains hard constraints and another one contains soft constraints. If the hard constraints are not satisfied the current loop which iterates over the adjacent nodes will terminate. If the soft constraints are not satisfied the current iteration will be skipped and the algorithm will go to the next adjacent node (see Algorithm1).

Before calling Algorithm 1 the flights must be ordered increasingly based on departure time inside the data structure which contains them, in our case a list. If this has been done, then the hard constraints are satisfied if:

- When iterating over the adjacent nodes of a node, the departure time of the current node is less than the departure time of the target node;
- The number of duties is within the parameters selected by the user;
- The number of sectors inside a duty is within the parameters select by the user;

whereas the soft constraints are satisfied if:

- The maximum pairing timespan is less or equal than the maximum pairing timespan allowed;
- All link constraints are satisfied.

Algorithm 1 shows the pseudocode of the depth-first search algorithm which has been use for testing purposes in this paper to generate the traditional pairings whereas Algorithm 2 has been used for generating roundtrips. *startNode* is a variable which represent a flight. When calling the algorithm for the first time, *startNode* will be the departure flight of the pair flights whereas the *endNode* will always represent the arrival flight of the pair flights. The *visitedList* is a data structure which is a list in our case and it is used for holding partial paths and *pathList* is the list which saves the created pairings.

Algorithm 1 Depth-first search algorithm to generate traditional pairings for a pair of nodes

```

1:  $visitedList \leftarrow visitedList \cup startNode$ 
2:  $pathList$  ▷ all created pairings will be added to this list
3: procedure DFS( $startNode, endNode$ )
4:   for all adjacent nodes  $adjNode \in startNode$  do
5:     if hard constraints are not satisfied then
6:       Terminate iterations
7:     if soft constraints are not satisfied then
8:       Skip current iteration
9:     if  $endNode = adjNode$  then
10:       $visitedList \leftarrow visitedList \cup adjNode$ 
11:       $pathList \leftarrow pathList \cup visitedList$ 
12:       $visitedList \leftarrow visitedList - adjNode$ 
13:      Terminate iterations
14:   for all adjacent nodes  $adjNode \in startNode$  do
15:     if  $adjNode \notin visitedList$  and  $adjNode \neq endNode$  then
16:       if hard constraints are not satisfied then
17:         Terminate iterations
18:       if soft constraints are not satisfied then
19:         Skip current iteration
20:        $visitedList \leftarrow visitedList \cup adjNode$ 
21:       DFS( $adjNode, endNode$ )
22:       Remove last element from  $visitedList$ 

```

Algorithm 2 Depth-first search algorithm to generate end-at-first-base pairings for a pair of nodes

```

1:  $visitedList \leftarrow visitedList \cup startNode$ 
2:  $pathList$   $\triangleright$  all created pairings will be added to this list
3: procedure DFS( $startNode, endNode$ )
4:   for all adjacent nodes  $adjNode \in startNode$  do
5:     if hard constraints are not satisfied then
6:       Terminate iterations
7:     if soft constraints are not satisfied then
8:       Skip current iteration
9:     if  $endNode = adjNode$  then
10:       $visitedList \leftarrow visitedList \cup adjNode$ 
11:       $pathList \leftarrow pathList \cup visitedList$ 
12:       $visitedList \leftarrow visitedList - adjNode$ 
13:      Terminate iterations
14:   for all adjacent nodes  $adjNode \in startNode$  do
15:     if  $adjNode \notin visitedList$  and arrival station of  $adjNode \neq$  arrival
    station of  $endNode$  then
16:       if hard constraints are not satisfied then
17:         Terminate iterations
18:       if soft constraints are not satisfied then
19:         Skip current iteration
20:        $visitedList \leftarrow visitedList \cup adjNode$ 
21:       Dfs( $adjNode, endNode$ )
22:       Remove last element from  $visitedList$ 

```

Consider the network from Figure 8. Applying the algorithms above on this network with Stockholm as a base and F2 as carry in and F10 as carry out one would get the set of pairings from Table 11 when using Algorithm 1 and the set of pairings from Table 12 when using Algorithm 2.

| | |
|----|-----------------------------|
| P1 | F1 - F3 - F7 |
| P2 | F1 - F3 - F7 - F8 - F9 |
| P3 | F1 - F3 - F10 |
| P4 | F1 - F4 - F5 - F7 |
| P5 | F1 - F4 - F5 - F7 - F8 - F9 |
| P6 | F1 - F4 - F5 - F10 |
| P7 | F1 - F4 - F6 |
| P8 | F1 - F4 - F6 - F8 - F9 |
| P9 | F8 - F9 |

| | |
|-----|------------------------|
| P10 | F2 - F5 - F7 |
| P11 | F2 - F5 - F10 |
| P12 | F2 - F5 - F7 - F8 - F9 |
| P13 | F2 - F6 |
| P14 | F2 - F6 - F8 - F9 |

Table 11: Pairings generated with Algorithm 1 based on network from Figure 8

| | |
|----|--------------------|
| P1 | F1 - F3 - F7 |
| P2 | F1 - F3 - F10 |
| P3 | F1 - F4 - F5 - F7 |
| P4 | F1 - F4 - F5 - F10 |
| P5 | F1 - F4 - F6 |
| P6 | F8 - F9 |
| P7 | F2 - F5 - F7 |
| P8 | F2 - F5 - F10 |
| P9 | F2 - F6 |

Table 12: Pairings generated with Algorithm 1 based on network from Figure 8

3.4.2 Searching on a pairing-based network

Generating pairings from a pairing-based network is almost the same as generating pairings from a flight-based network. As a pairing contains multiple flights, it is recommended to assume that the pairing has departure and arrival times, and departure and arrival stations. The departure time and station of a pairing is the departure time and station of the first flight of the pairing and the arrival time and station is the arrival time and station of the last flight of the pairing. The only algorithm used to generate pairings based on a pairing-based network is Algorithm 1. In this case *startNode* and *endNode* are pairings.

Given the pairings from Table 12 and the network from Figure 9, if one would apply Algorithm 2 one would end up with the same pairings as in Table 11.

4. OPTIMIZATION MODEL

4.1 Problem Formulation

The problem is formulated as a set covering problem. The set covering problem is almost the same as the set partition problem with the difference that the former requires each set element to be found in at least one subset whereas the latter requires each set element to be found in exactly one subset. One reason for which the set covering model is preferred in crew pairing optimization is because, for real schedules, the problem will be almost always infeasible if the set partition model is used, as usually overcovers are required which excludes all the possibilities of a partition to exist.

$$\begin{aligned}
 & \min \sum_{j \in J} c_j x_j \\
 & \text{s. t. } \sum_{j \in J} a_{ij} x_j \geq 1 \quad \forall i \in I \\
 & \quad x_j \in \{0,1\}
 \end{aligned} \tag{4}$$

As it can be seen the only difference between the formulation above and the one from the heading 2.1 is that the equality constraint has been changed to inequality.

The cost model is assumed to be a linear function. The scope is to fit as many flights as possible inside a pairing, therefore three different costs will compute the final cost of a pairing. One of it is a constant (*CT*) which is a fix cost for all pairings, another one is the sit connection cost (*SCC*) which is a function of the time spent inside a duty between flights and the last one is the layover cost (*LC*) which is a function of the rest time spent between duties.

The pairing cost (*PC*) is a sum of the costs described above.

$$PC = CT + SCC + LC \tag{5}$$

Example 1: Set covering formulation for the traditional pairings

Input Data:

Pairings: Table 11

Costs:

- *CT*: 20
- *SCC*: 1/h
- *LC*: 1/h

Solver: Google OR MIP

Optimization problem

$$\min_{x \in [0,1]^{14}} [26 \ 41 \ 44 \ 23 \ 36 \ 41 \ 23 \ 29 \ 21 \ 25 \ 43 \ 38 \ 25 \ 41] \begin{bmatrix} x_1 \\ \vdots \\ x_{14} \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ x_{14} \end{bmatrix} \geq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \tag{6}$$

The optimal solution of the problem above is selecting variables x_3, x_7, x_{12} to be 1 and rest of them 0, and the objective value is 105.
 This optimization problem can't be formulated as a set partition problem because there is a position required and the problem would turn to be infeasible.

Example 2: Set covering formulation for the roundtrips.

Input Data:

Pairings: Table 12

Costs:

- *CT*: 20
- *SCC*: 1/h
- *LC*: 1/h

Solver: Google OR MIP

Optimization problem

$$\min_{x \in [0,1]^9} [26 \ 44 \ 23 \ 41 \ 23 \ 21 \ 25 \ 43 \ 25] \begin{bmatrix} x_1 \\ \vdots \\ x_9 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_9 \end{bmatrix} \succeq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad (7)$$

The optimal solution of the problem above is selecting variables x_2, x_3, x_6, x_9 to be 1 and rest of them 0, and the objective value is 113.

Now, based on the optimal solution from the roundtrips one should create the optimization formulation for the next step which takes the variables input based on the pairings generated from the pairing-based network.

It is easy to see that all possible combinations are: P3-P6 and P9-P6.

Example 3: Set covering formulation for the roundtrips solutionInput Data:

Pairings: Table 12

Costs:

- CT : 20
- SCC : 1/h
- LC : 1/h

Solver: Google OR MIP

Optimization problem

$$\min_{x \in [0,1]^6} [44 \ 23 \ 21 \ 25 \ 36 \ 41] \begin{bmatrix} x_1 \\ \vdots \\ x_6 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_6 \end{bmatrix} \geq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad (8)$$

The optimal solution of the problem above is selecting variables x_1, x_4, x_5 to be 1 and rest of them 0, and the objective value is 105.

It can be noticed that the pairing solution is different compared to the Example 1, but the objective value is the same and this is due to fact that the optimization solution from Example 1 has multiple optimal solutions.

4.2 Overcover Penalty

The set covering formulation let the possibility of having unnecessary overcovers open. For real world schedules there are usually multiple optimal solutions and many of them will lead to overcovers which are neither required nor cheaper.

One overcover requires that an extra set of crew must be on the flight as this could be the only link to another flight or the solution is optimal. The purpose of the inequality in (4) is exactly to take care of the instances where the only possibility to cover some flights is by making use of the overcovers. Take into consideration the Example 1 where the optimal solution requires one overcover to be feasible as both F3 and F4 have a predecessor connection just to F1.

$$\begin{aligned}
 \min \quad & \sum_{j \in J} c_j x_j + \sum_{i \in I} P_i \left(\sum_{j \in J} a_{ij} x_j - 1 \right) \\
 \text{s. t.} \quad & \sum_{j \in J} a_{ij} x_j \geq 1 \quad \forall i \in I \\
 & x_j \in \{0,1\}
 \end{aligned} \tag{9}$$

In (9) can be seen that one more term has been introduced. This term is P and represent the penalty coefficient for each flight. The reason for having penalty coefficients on each flight assumes that from the operations point of view there is a priority on which flights to be overcovered as some of them might be fully booked and others might still have some free seats.

Example 4: Set covering formulation for traditional pairings with overcover penalty

Input Data:

Pairings: Table 11

Costs:

- CT : 20
- SCC : 1/h
- LC : 1/h
- P_i : 10

Solver: Google OR MIP

Optimization problem

$$\min_{x \in [0,1]^{14}} [56 \ 91 \ 74 \ 63 \ 96 \ 81 \ 53 \ 79 \ 41 \ 55 \ 73 \ 88 \ 45 \ 81] \begin{bmatrix} x_1 \\ \vdots \\ x_{14} \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_{14} \end{bmatrix} \geq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad (10)$$

The optimal solution of the problem above is selecting variables x_3, x_7, x_{12} to be 1 and rest of them 0, and the objective value is 215.

Example 5: Set covering formulation for roundtrips with overcover penalty

Input Data:

Pairings: Table 12

Costs:

- CT : 20
- SCC : 1/h
- LC : 1/h
- P_i : 10

Solver: Google OR MIP

Optimization problem

$$\min_{x \in [0,1]^9} [56 \ 74 \ 63 \ 81 \ 53 \ 41 \ 55 \ 73 \ 45] \begin{bmatrix} x_1 \\ \vdots \\ x_9 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ x_9 \end{bmatrix} \geq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad (11)$$

The optimal solution of the problem above is selecting variables x_2, x_3, x_6, x_9 to be 1 and rest of them 0, and the objective value is 223. All penalty coefficients for all flights have been considered equal.

Example 6: Set covering formulation for roundtrips solution with overcover penaltyInput Data:

Pairings: Table 12

Costs:

- CT : 20
- SCC : 1/h
- LC : 1/h
- P_i : 10

Solver: Google OR MIP

Optimization problem

$$\min_{x \in [0,1]^6} [44 \ 23 \ 21 \ 25 \ 36 \ 41] \begin{bmatrix} x_1 \\ \vdots \\ x_6 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_6 \end{bmatrix} \geq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad (12)$$

The optimal solution of the problem above is selecting variables x_1, x_4, x_5 to be 1 and rest of them 0, and the objective value is 215.

It can be noticed that the pairing solution is different compared to the Example 4, but the objective value is the same and this is due to fact that the optimization solution from Example 1 has multiple optimal solutions.

4.3 Robustness Penalty

Airlines usually create the pairings six months in advance as it is a time-consuming stage and they also need to create the crew rosters which are depended on the pairings solution. But usually in the day-of-operations many of the pairings created won't be used

anymore due to a disruptive schedule. Therefore, many airlines adjust and create some of the pairings manually in this stage. Disruptive schedules can appear due to the weather conditions or if some technical problem appears to an aircraft. There can also be many other reasons for a schedule to change and all these changes will lead to flight delays and it can become impossible for some of the crews to be in time for their connecting flights. Here, a method through which the disruptive schedules are handled is presented.

As flight delays can make the crew's connections impossible one way to remove this situation is to keep the crew with the tail. This will be impossible in some of the situations due to the rules and regulations which apply for crews, therefore enforcing a hard constraint to keep the crew with the tail will lead in infeasible solutions. The solution proposed in this paper is to keep the crew with the tail by eliminating the hard constraints. This can be achieved by applying a penalty for the number of tails changes inside a pairing where the connection is less than a specified threshold. Basically, the penalty to be applied will try to keep the crew with the tail and this will happen at a higher cost.

Equation (13) shows the mathematical formulation of the set covering problem with the vehicle change penalty. T represents the coefficient of the penalty and n_j represents the number of tails changes inside pairing j .

$$\begin{aligned}
 & \min \sum_{j \in J} c_j x_j + T \sum_{j \in J} n_j x_j \\
 & \text{s. t. } \sum_{j \in J} a_{ij} x_j \geq 1 \quad \forall i \in I \\
 & \quad x_j \in \{0,1\}
 \end{aligned} \tag{13}$$

Let's consider the schedule form Table 4 and the pairing from Table 11. Say 4 hours is the threshold for the penalty to be applied on a vehicle change and 100 is the coefficient of the penalty. If a connection is larger than 4 hours, then no vehicle penalty will be applied. Table 13 shows the number of tail changes inside a pairing.

| Pairings | n_j | Tn_j |
|-----------------|-------|--------|
| P1 | 1 | 100 |
| P2 | 1 | 100 |
| P3 | 1 | 100 |
| P4 | 1 | 100 |
| P5 | 1 | 100 |
| P6 | 0 | 0 |
| P7 | 1 | 100 |
| P8 | 1 | 100 |
| P9 | 0 | 0 |
| P10 | 2 | 200 |
| P11 | 1 | 100 |
| P12 | 2 | 200 |
| P13 | 0 | 0 |
| P14 | 0 | 0 |

Table 13: No of tails changes and the associated penalties for the schedule from Table 4 when the threshold is set to 4 hours and the coefficient of the penalty is 100. Traditional pairings.

The optimal solution of this schedule with the vehicle change penalty is to be found in Example 7.

Example 7: Set covering formulation for traditional pairings with vehicle change penalty.

Input Data:

Pairings: Table 11

Costs:

- CT : 20
- SCC : 1/h
- LC : 1/h
- T : 100
- Threshold for penalty: 4 h

Solver: Google OR MIP

Optimization problem

$$\min_{x \in [0,1]^{14}} [126 \ 141 \ 144 \ 123 \ 136 \ 41 \ 123 \ 129 \ 21 \ 225 \ 143 \ 238 \ 25 \ 41] \begin{bmatrix} x_1 \\ \vdots \\ x_{14} \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ x_{14} \end{bmatrix} \geq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad (14)$$

The optimal solution of the problem above is selecting variables x_2, x_6, x_{13} to be 1 and rest of them 0, and the objective value is 207.

It can be noticed that now we have just one vehicle change in the solution which can lead to an disruptive schedule compared to the solution from Example 1, where x_3, x_7, x_{12} were set to 1 and there was 4 vehicle changes.

Now, let us consider the instance with the schedule from Table 12. The same 4 hours we apply for the threshold and we set the penalty coefficient T to 100 as well. Table 14 shows the number of changes and the penalty to be applied for each variable.

The optimal solution of the problem above is selecting variables x_1, x_4, x_6, x_9 to be 1 and rest of them 0, and the objective value is 213.

It can be noticed that the optimal solution is different compared to the one from Example 2, as now the solution has one vehicle change compared to two vehicle changes in Example 2

If the depth-first search algorithm is applied on the pairing solution from Example 8 and the result is added in the optimization problem along with the pairings from the solution, Example 9 can be created. All possible combinations are $P1 - P6$ (which is $P2$ in Table 13) and $P9 - P6$ (which is $P14$ in Table 13). And the associated penalty table for all pairings is:

| Pairings | n_j | Tn_j |
|----------|-------|--------|
| P1 | 1 | 100 |
| P2 | 0 | 0 |
| P3 | 0 | 0 |
| P4 | 0 | 0 |
| P5 | 1 | 100 |
| P6 | 0 | 0 |

Example 9: Set covering formulation for pairings out of the roundtrips with vehicle change penalty.

Input Data:

Pairings: solution from Example 8 + all feasible combinations ($P1 - P6$ and $P9 - P6$)

Costs:

- CT : 20
- SCC : 1/h
- LC : 1/h
- T : 100
- Threshold for penalty: 4 h

Solver: Google OR MIP

Optimization problem

$$\min_{x \in [0,1]^6} [156 \ 81 \ 41 \ 45 \ 141 \ 41] \begin{bmatrix} x_1 \\ \vdots \\ x_6 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_6 \end{bmatrix} \geq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad (16)$$

The optimal solution of the problem above is selecting variables x_2, x_4, x_5 to be 1 and rest of them 0, and the objective value is 207.

It can be noticed that both the objective value and the pairing solution is the same as in Example 7.

4.4 Integrated Optimization Model

The overcover and the vehicle change penalties are two very important factors in an airline crew pairing optimization tool. With the model described in this paper one can

prioritize vehicle change over overcovers or the other way around. This makes an efficient tool when it comes about the cost of operations and the day-of-operations.

Equation (17) shows the integrated optimization formulation.

$$\begin{aligned}
 \min \sum_{j \in J} c_j x_j + \sum_{i \in I} P_i \left(\sum_{j \in J} a_{ij} x_j - 1 \right) + T \sum_{j \in J} n_j x_j \\
 \text{s. t. } \sum_{j \in J} a_{ij} x_j \geq 1 \quad \forall i \in I \\
 x_j \in \{0,1\}
 \end{aligned} \tag{17}$$

5. SIMULATIONS

In the previous chapters the problem has been described in detail and the difference has been emphasized between the traditional pairing generation and the pairings generated based on the roundtrips solution. Under this heading different simulations to compare the processing time on a real schedule will be presented.

The difference between our model framework and crew pairing optimization traditional framework will be compared. Recall that the traditional framework is when one generates all feasible pairings and then one optimizes over them whereas our model framework is represented by four stages; in the first stage, all feasible end-at-first-base pairings are generated using Algorithm 2, then, in the second stage we optimize over the pairing generated in the previous stage and then we generate pairings again based on the solution from the second stage and using Algorithm 1, and, finally, in the fourth stage we optimize over the pairings generated in the third stage.

The processing time presented in Table 15 represents the sum between the generation time and the optimization time, *TP generated* represents the number of pairings generated using the traditional method, *NCP generated* represents the pairing generated with the model framework presented in this paper and it is the sum of the pairings from stage 1 and 3.

The following constraints are enforced for the results presented in Table 15:

- Minimum transit time = 30 min
- Maximum transit time = 240 min
- Minimum layover time = 10 h
- Maximum layover time = 15 h
- Maximum duty time = 12 h
- Maximum pairing time = 2 days
- Maximum number of duties = 2

| Scenarios | A | B | C | D |
|---------------------------------------|-------|-------|--------|--------|
| No of Flights | 421 | 873 | 1524 | 3085 |
| TP generated | 21884 | 79149 | 222076 | 574916 |
| NCP generated | 3604 | 10125 | 27145 | 73835 |
| Processing Time with TP [sec] | 19 | 93 | 388 | 3171 |
| Processing Time with NCP [sec] | 1,47 | 9,89 | 44,68 | 196 |

Table 15: Results which show the efficiency of the model framework presented in this paper.

Looking at Table 15 it can be noticed the superiority of the concept presented in here. The processing time of using the new concept is significantly lower compared to the old approach. This plays a key role for an efficient airline crew pairing optimization tool. A mixed integer programming solver from Google OR has been used to solve the optimization problems. All the algorithms have been programmed in C# and the tests took place on a laptop with the following specifications:

- Processor: Inter Core i7-6600U CPU @ 2.60 GHz
- RAM: 16,0 GB
- System type: 64-bit Operating System

6. CONCLUSIONS

Crew pairing optimization problem plays an important role in airline's industry as it can reduce the crew costs significantly. This thesis has presented a complete approach for this problem with robust pairings and unnecessary overcovers elimination. One of the key factors was the roundtrips generation which led both to a reduction of the search area when generating pairings and to a reduction of the number of pairings generated. Reducing the number of pairings plays an important role in the optimization stage. As the variables from the optimization formulation represent all the generated pairings it is crucial to reduce them as the processing time of the optimization stage is polynomial. Two types of network have been presented, a flight-based network used to generate the roundtrips and a pairing-based network which has been used to generate new combinations of pairings from the optimal solution of the roundtrips. It has been shown in the examples from Chapter 4 that this approach leads to the same objective value compared to the traditional approach when all feasible pairings were generated and then the optimization stage was taking place.

One could consider as an extension from this thesis, for future work, to implement a method through which base manpower constraints is taken into consideration for each base. Many carriers would prefer a tool where, they can distribute the crews to be used by the pairings created.

The idea behind pairings is to reduce the crew scheduling cost but as this stage is divided in crew pairing and crew rostering the final solution will be heuristic. Therefore, another extension would be to integrate these two stages rather than using them separately and to find a solution closer to optimal.

Even with the model presented in this paper, sometimes the generated pairings could reach a high number and so the optimization time needed. A heuristic preprocessing of the constraints matrix to reduce the number of columns and implicitly the optimization processing time could be a good extension from this thesis.

REFERENCES

- [1] EASA, "Flight and Duty Time Limitations and Rest Requirements," 2016.
- [2] Barnhart and Talluri, 1997.
- [3] M. Bazargan, *Airline Operations and Scheduling* 2nd Edition, 2010.
- [4] E. L. J. G. L. N. Cynthia Barnhart, *Handbook of transportation Science - Routing and Network Models - Crew Scheduling*.
- [5] Lavoie, 1998.
- [6] V. Dück, F. Wesselmann and L. Suhl, "Implementing a branch and price and cut method," 2011.
- [7] A. Ranga, E. Gelman, B. Patty and R. Tanga, "Recent Advances in Crew-Pairing Optimization at American Airlines," pp. 62-74, 1991.
- [8] C. Banhart, "Airline crew scheduling," 2003.
- [9] M. Desrochers and F. Soumis, "A generalized permanent labelling algorithm," *INFOR*, 1988.
- [10] E. Andersson, E. Housos, N. Kohl and D. Wedelin, "Operations Research In The Airline Industry," pp. 228-256.
- [11] E. L. J. BALAJI GOPALAKRISHNAN, "Airline Crew Scheduling: State-of-the-Art," 2005.
- [12] T. Cormen, C. Leiserson, R. Rivest and S. Clifford, "Introduction to Algorithms, Third Edition," pp. 603-612, 2009.
- [13] R. T. E. J. Anbil, "A global approach to crew-pairing optimization," *IBM Systems Journal*, vol. 31, no. 1, pp. 71-78, 1992.
- [14] G. YU, "Or in Airline Industry".
- [15] "www.aviolinx.com".

TRITA -SCI-GRU 2018:011