An Aircraft and Schedule Integrated Approach to Improve Cockpit Crew Pairings

Johanna Korte

TU Delft

transavia

An Aircraft and Schedule Integrated Approach to Improve Cockpit Crew Pairings

by

Johanna Korte

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday December 12, 2019 at 3:00 PM.

Student number:	4697928	
Project duration:	February 4, 2019 – December 12, 2019	
Thesis committee:	Dr. N. Yorke-Smith,	TU Delft, supervisor
	Dr. Ir. J. T. van Essen,	TU Delft
	Dr. B. F. Santos,	TU Delft
	K. van Eeden MSc.	Transavia

TU Delft

Abstract

For airlines, crew costs make up the second largest expense, behind fuel costs. Because these costs are very high, there is a large potential gain in improving the crew efficiency within the bounds set by the law and collective labor agreements. This thesis investigates the dated crew pairing problem, and how the crew pairing problem can be integrated with aircraft routing and flight retiming in order to achieve more crew-efficient schedules for a low-cost airline operating a point-to-point network. Three different levels of problem integration, non-integrated, aircraft routing integrated, and aircraft routing integrated including retiming, are investigated on real point-to-point airline data, leading to five different models using either a generate-and-test or branch-and-price approach. It is shown that the currently presented models return pairings that reduce the number of duties up to 10% and increase the crew productivity up to 1.5%.

Preface

Before you lies the thesis *"An Aircraft and Schedule Integrated Approach to Improve Cockpit Crew Pairings"*, which examines how several problems within airline scheduling can be solved integrally in order to create more efficient crew pairings. This thesis was written to fulfill the last graduation requirement and to obtain the degree of Master of Science in Computer Science at the Delft University of Technology.

It has been a privilege to work under the supervision of Dr. Yorke-Smith, whom I would like to thank for the countless meetings we had to discuss my progress and new ideas, posing questions to lead me in the right direction, and the overall kind supervision that kept me motivated throughout the entire process.

The questions that form the basis of the present research have been posed by Transavia, a Dutch low-cost airline, where I have also had the pleasure to intern for the duration of this project. I would like to thank Karin and Maarten, my supervisors at Transavia, for their guidance, contributing ideas, and practical insights. And to all my other Transavia colleagues: thank you for all the brainstorming sessions, critical questions, great working environment and fun moments around the office.

Lastly, I would like to thank my family for their listening ear and endless faith in me; and in particular my partner Niels for his unwavering support, keeping me motivated, and providing the beautiful picture on the front cover.

I hope you enjoy reading my thesis.

Johanna Korte
Schiphol, December 5, 2019

Contents

List of Figures

List of Tables

List of Abbreviations

BH	Block Hours - all time between push-back and back on gate
CPP	Crew Pairing Problem
DP	Duty Period - Time from start to end of duty including positioning
FDP	Flight Duty Period - Time from check-in to arrival time of last flight
GVWT	"Gecorrigeerde Vlieg Werk Tijd" - Time from check-in to check-out corrected for extra stretches
lt	Local time
RMP	Restricted Master Problem
SPPRC	Shortest Path Problem with Resource Constraints
VND	Variable Neighborhood Descent

1 Introduction

To establish the context of this thesis, this chapter gives an introduction to airline and crew scheduling. In addition, the currently addressed problem, its significance, and the context in which this research was performed will be explained. To conclude the chapter, a summary of contributions to the current state-of-the-art is given.

1.1. Background

Airline scheduling consists of many different problems that are often solved independently and subsequently. The first phase is network development, where the desired origin-destination pairs and their corresponding frequencies are determined. It is then possible to construct a flight schedule, where each origin-destination pair gets scheduled at specific days and times. Subsequently, using demand forecasts, the *fleet assignment problem* is solved. In this problem, all flights in the schedule are assigned an aircraft type. The output of the fleet assignment problem will be the input for the fourth stage, i.e. the *aircraft routing problem*, where specific aircraft registrations are put on each flight, taking into account maintenance requirements. The last step of the process is scheduling crew, such that all flights can be operated. This is called the *crew scheduling problem*. Because this problem is NP-hard, it is usually solved in two phases: *crew pairing* and *crew rostering*. An overview of the entire airline scheduling process can be seen in Figure 1.1.

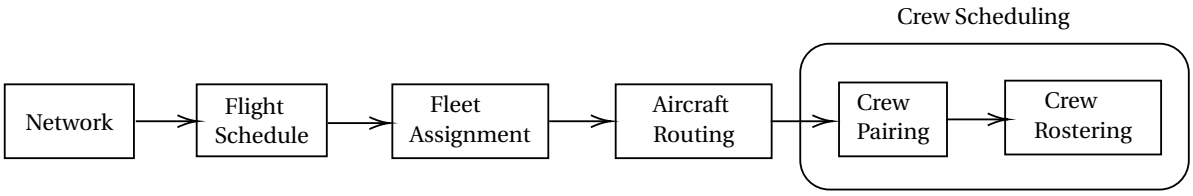


Figure 1.1: An overview of the airline planning process

The crew pairing phase aims to generate a set of crew duties, called *pairings*, that will cover all flights in the schedule, and minimize the cost of the set of pairings. A pairing is a sequence of assignments that begins and ends at the same base, and respects the law and all labor agreements between the airline and its labor unions. Next to flights, also called *legs*, these assignments can also include positioning by car, taxi, or aircraft, and hotel layovers. Positioning by aircraft is also called *deadheading*, and means that a crew member will be transported as a passenger [21]. Additionally, all pairings that contain a flight leg will include a briefing and debriefing period. If two flight legs are scheduled right after each other, a minimum *turnaround time*, needed for refueling, (de)boarding, and other flight preparations needs to be factored in. A few example pairings, with and without positioning activities and layovers, can be seen in Figure 1.2. Once a set of pairings has been found that covers all flights and minimizes the operational cost, the pairings are used as input for the crew rostering problem. Here, the pairings, free time, and other required activities are assigned to individual crew members, while aiming to distribute the workload fairly over all crew members [21]. This results in a roster for each crew member, and completes the airline's schedule.

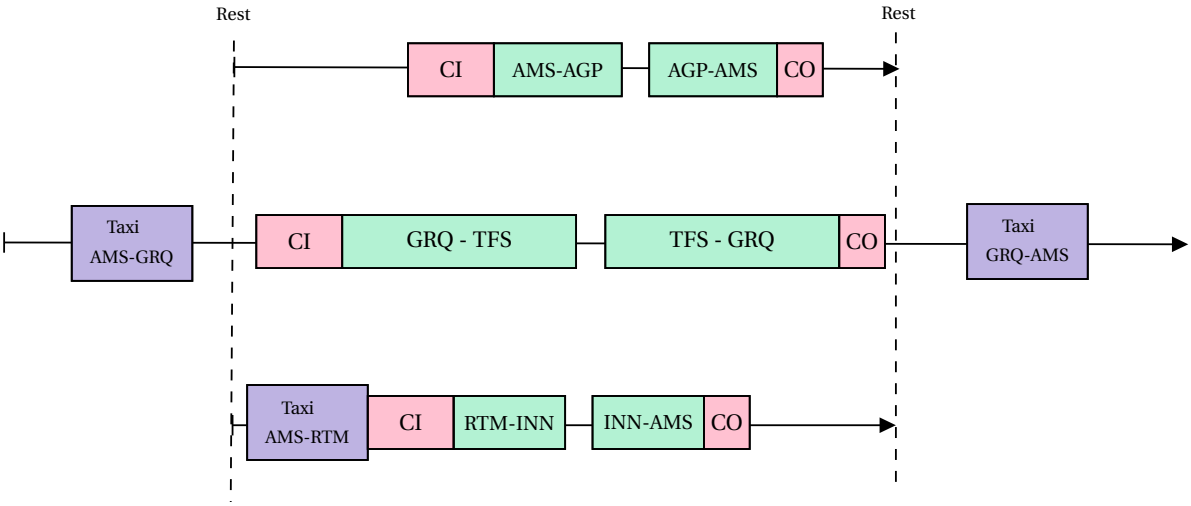


Figure 1.2: Examples of 3 different pairings. All flight activities are colored green, while the check-in and check-out activities are shown in red, and the taxi activities are shown in blue. The dashed lines indicate the location of a rest period. The upper and lower example both span 1 day, while the middle pairing spans 3 days.

Although all airlines operate differently, a large difference between airlines in the United States and Europe can be observed. This difference also influences scheduling decisions, especially with regard to the objective of scheduling problems. Many airlines in the US operate a hub-and-spoke network. All the airline's flights either depart from or arrive at one of its hub airports, such that a large number of origin-destination pairs can be served by having passengers connect at the hub. The flight schedules of hub-and-spoke airlines are designed to facilitate these transfers. On the other hand, many European airlines operate a point-to-point network. This type of operation consists of only offering flights directly from A to B, without transfers. Flight schedules are thus not designed to facilitate connections, and flights do not necessarily concentrate at hubs [16]. Another large difference between US and European airlines is in the crew pay structure. American airlines tend to pay flight crew based on a number of guaranteed hours per duty or time period, while European airlines often hire crew on fixed salary contracts [7].

1.2. Problem description

Crew costs are an airline's second highest expense, being only less expensive than fuel. Therefore, it is desirable to assign crew to the developed flight schedule as efficiently as possible. Although individual crew members only get assigned to specific flights in the crew rostering phase, the cost-determining phase of the crew scheduling problem is the pairing problem [34]. This thesis focuses on optimizing and automatizing solutions of the crew pairing problem for a low-cost airline operating a point-to-point network. Point-to-point networks, which have not been studied extensively in this context, consist mainly of short-haul back-and-forth stretches, from one of the bases to an outstation and back [16]. A flight schedule in such a network differs from a hub-and-spoke network in that there are less defined 'waves' of flights arriving at a hub airport, and leaving again. This results in fewer crew connection opportunities. Because there are fewer options for crew pairings, integrating the crew pairing problem with the aircraft routing problem and small flight schedule alterations could facilitate the optimization of the crew productivity. This specific problem has not been investigated extensively for European point-to-point airlines, and especially the focus on crew efficiency and the inclusion of integral schedule changes is new.

1.3. Research questions

In order to fill the presented research gaps, this thesis attempts to answer the following main research question, with its corresponding subquestions:

- How can the crew pairing problem be integrated with the aircraft routing problem and schedule optimization, so as to improve the crew productivity of a low-cost carrier operating a point-to-point network?
  - How can the cost of the generated pairings be formulated, such that it represents how efficient a pairing is?
  - What is the impact of integrally generating crew pairings and aircraft routes on the quality of the pairings in a point-to-point network?
  - How can realistic changes to the flight schedule be integrated in the pairing model in order to accommodate the optimization of crew pairings?

1.4. Context

This project was performed at Transavia, a Dutch low-cost airline[1]. Transavia operates a point-to-point network consisting of about 100 destinations with three main hubs: Amsterdam Airport Schiphol, Eindhoven Airport, and Rotterdam The Hague Airport. While a large part of Transavia's cabin crew work on seasonal or flexible contracts, almost all of the cockpit crew work on permanent contracts, where the pilots receive a fixed salary independent of the number of duty or flight hours. To give an idea of Transavia's flight operations, Figure 1.3 shows Transavia's flight network for the period between March and November of 2019.

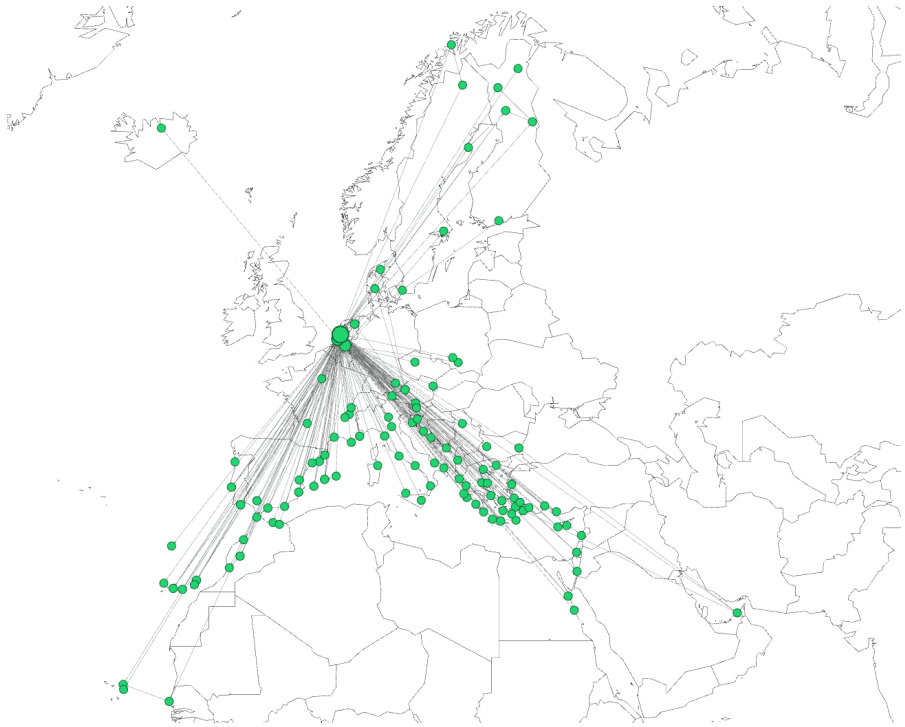


Figure 1.3: Transavia's flight network

[1] www.transavia.com

1.5. Contributions

This thesis provides a comparison of different models for the dated crew pairing and integrated crew pairing problems, on data from a point-to-point airline. Contrary to previously published studies, the presented models are used to investigate the possibilities of crew efficiency, by formulating the objective as minimizing idle time. This is especially new in the context of a European point-to-point airline. Additionally, the presented models are greatly expressive and thus increase the applicability of the solutions in airline operations. The last pages of this thesis additionally provide a draft of what a paper on one of the currently presented models would like.

2

Literature Review

This chapter surveys the existing literature on the crew pairing problem and previous solution approaches. First, common problem formulations and objective functions for the crew pairing problem are discussed, after which methods to generate a set of initial feasible pairings are examined. A large section is dedicated to the variety of ways in which the pairing problem can be solved. Lastly, literature on integrated airline scheduling will be discussed.

2.1. Crew Pairing Problem

2.1.1. Objectives

The objective of the pairing problem is highly dependent on how the airline in question operates. In general, studies aim to minimize the cost of executing a set of pairings, as defined by fixed and variable crew salary components, crew transportation costs, and the costs of layovers [6][17][25][38]. Others also include cost penalties based on time away from base and sitting time [18][35]. Especially with American airlines the concept of *pay-and-credit*, also known as *excess cost*, is frequently used. The pay-and-credit (PaC) for a pairing p containing f_p flight hours, with a minimum of g guaranteed hours is calculated according to Equation 2.1. Another frequently used and similar metric for these airlines is the *flight time credit (FTC)*, given by Equation 2.2[15][31]. The use of these objectives can be explained by the structure of crew costs for North-American airlines. As there is a minimum guaranteed number of paid hours, it is advantageous to first use these hours, before generating pairings that have a lot of excess cost.

$$PaC(p) = \max(g - f_p, \quad 0) \tag{2.1}$$

$$FTC(p) = \frac{PaC(p)}{f_p} \cdot 100 \tag{2.2}$$

In contrast, many European airlines hire pilots on fixed salary contracts, and thus do not know the concept of excess cost and FTC. Instead, objective functions for these problems are often designed to minimize the total costs of all pairings as described above [19][22][25][52], or in some cases to minimize the total number of pairings in the final solution [4]. An overview of studies investigating the crew pairing problem, their respective data sources, and used objective function can be found in Table 2.1. Surprisingly, very little is currently known about the relationship between the value of the objective function in the pairing problem and the performance of the resulting set of pairings in the subsequent rostering problem. This is important to consider, as the output of the pairing problem will be used as the input for the rostering problem. When rostering crew, the first pairing of a crew member's sequence might have an effect on their availability for subsequent pairings. This is especially the case with very long or late duties, which require longer rest periods in between pairings. Consequently, a set of pairings might minimize operating cost or require fewer pairings to cover all flights, but might not be desirable in the rostering phase. This is especially the case for airlines that hire flight crew on fixed salaries, as no costs are associated with individual pairings, but rather with the required number of crew members.

Table 2.1: Formulation of the objective function across previous studies, studying only the crew pairing problem and sorted by data source

Year	Authors	Data	Objective function formulation
1997	Chu et al. [15]	American Airlines	Minimize pay-and-credit
2001	Klabjan et al. [31]	United Airlines	Minimize pay-and-credit
2003	Klabjan et al. [33]	United Airlines	Minimize pay-and-credit and cost of deadheads, maximize repetition in schedules
2013	Saddoune et al. [44]	Major US airline	Minimize pay-and-credit, pairing duration, duty duration and deadheads
2017	Quesnel et al. [40]	Major US airline	Minimize pairing costs (duty length, worked time)
1997	Desaulniers et al. [19]	Air France	Minimize total cost
2015	Erdogan et al. [25]	European airline	Minimize crew-related costs (deadheads, layovers and ground transportation)
2017	Agustin et al. [4]	European airline	Minimize number of pairings
2016	Zeren and Özkol [52]	Turkish Airlines	Minimize operating costs (cost per duty day, deadhead hour, layovers)
2018	Deveci and Demirel [22]	Local Turkish airlines	Minimize pairing costs (duty costs, rest expenses, connection time expenses)
2001	Ozdemir and Mohan [38]	Multiple airlines	Minimize total costs (pay-and-credit, deadheads, sitting time, layovers)
2013	Aydemir-Karadag et al. [5]	Randomly generated	Minimize total pairing cost (pay-and-credit, time away from base and total duty cost)
2013	Azadeh et al. [6]	Randomly generated	Minimize total crew cost (flight payments, rest expenses, deadhead costs)

2.1.2. The Problem Horizon

With regards to the problem horizon, a few different variants of the crew pairing problem exist. Due to the hardness of the problem and its rapidly increasing size, many studies solve a daily problem where the flight schedule that serves as input contains all flights for an airline on 1 day [6][10][36]. Another approach that is used regularly is that of the weekly problem [33]. The daily and weekly problems have as an advantage that a single problem can be solved first, after which the solution is copied onto the next day or week, after which any remaining inconsistencies are solved. These kinds of approaches are therefore very suitable for airlines that have repetitive schedules: something that is mostly seen in hub-and-spoke networks. In many cases, the daily problem is solved first, and it is assumed that the resulting solution can be repeated 7 times, such that an entire week is covered. Then, an exceptions problem is solved, where the infeasible pairings, resulting from a not completely repetitive schedule, are repaired. Although this approach is time-efficient compared to solving an entire week, it results in a lot of deadheading for the crew [30][31][33]. A different approach has been taken by Klabjan et al. on the data of United Airlines, where instead of repeating the daily problem, a weekly problem with regularity variables is proposed [33]. This resulted in crew pairings that were more regular, contained fewer deadheads and had a lower FTC. Far fewer studies aim to solve a monthly problem. Erdoğan et al. solve a monthly problem by using a heuristic approach consisting of integer programming, enumeration, and large neighborhood search, while Saddoune et al. applied a rolling horizon technique [25][44]. Lastly, for schedules without a repetitive nature, a dated problem can be solved. However, this kind of problem is relatively rare in literature, as few of the investigated airlines have variable schedules [12].

2.2. Exact approaches

2.2.1. Problem Formulation

From the first studies onwards, a vast majority of research has formulated the crew pairing problem as either a set partitioning or a set cover problem [25][31][41][52]. These formulations enforce all flights to be covered by a pairing, and can easily be extended to capture the scope of the problem by adding constraints. Whereas the set partitioning formulation requires all flights to be covered by exactly one pairing, the set cover formulation allows pairings to overlap. In the latter case, the superfluous crews assigned to a flight leg will deadhead the flight. The basic formulation of the crew pairing problem as a set cover problem can be seen in Equation 2.3

[7]. In this formulation, F represents the set of flights that need to be covered, while P represents the set of pairings to choose from. The cost of each pairing is c_p, while x_p represents whether a pairing is included in the final set of pairings. The first constraint makes sure that all flights are covered by at least one pairing. This is done by means of a constraint matrix with columns p and flights i. If a pairing p contains flight i, the value at (i, p) will be 1. If it does not contain i, its value will be 0 [7]. Once a set of candidate pairings has been generated, the pairing problem can be solved using various approaches. The applicability of each method depends on the desired solution quality and time constraints. The following subsections will describe exact and metaheuristic solving methods that have previously been investigated for the crew pairing problem.

$$\begin{aligned} \min_{p \in P} \quad & \sum_{p \in P} c_p x_p & & & & \\ \text{s.t.} \quad & \sum_{p : i \in p} x_p & \geq & \ 1 & \quad \forall i \in F \\ & x_p & \in & \ \{0,1\} & \quad \forall p \in P \end{aligned} \tag{2.3}$$

2.2.2. Pairing Generation Approach

To solve the pairing problem as a set cover or set partition problem, a set of candidate pairings is required. This set can be generated by enumeration or with the use of heuristics. Due to the many possible combinations of flight legs, layovers and taxi rides, enumerating all possible pairings is often too computationally expensive and results in many unnecessary pairings. This is especially the case in flight networks that contain many flights to or from a small number of hubs [28]. To generate the pairings, Goumopoulos and Housos [27] created a directed acyclic graph using all activities. A node in the graph represents an activity, while an edge represents a legal combination of the two activities for a pairing. This means the first activity ends before the second starts, and the first activity ends at the position where the second one starts. Nodes corresponding to the airline's start and end bases are identified in the graph as start and terminal nodes respectively. Each start node now represents the root of a search tree, and all pairings can thus be found by applying depth first search on all trees. In order to decrease runtimes while still providing a good set of pairings, Goumopoulos and Housos pruned the previously described search trees using a rule-based approach [27]. Starting from the root node, depth-first search (DFS) was applied until one of the pairing constraints violates. In this case, the algorithm will backtrack to the previous node and explore the next branch. Because this approach was still computationally inefficient on some types of graphs, Goumopoulos and Housos used an enhanced search graph instead, to predict inefficient branches as soon as possible [27]. The search graph was enhanced by adding additional information that could be exploited to prune branches as soon as they were expected to violate rules. Aggarwal et al. applied a comparable depth-first search technique, but proposed two modifications to the DFS algorithm and executed the generation process in parallel to further decrease its runtime [2]. Using depth-first search for enumeration may, depending on the data, not always finish in finite time. To still get a set of pairings that will cover at least all flight legs, and therefore be usable as input for the optimization phase, the depth-first search is modified to include variable-backtracking. When the end of a branch is reached in the original DFS algorithm, backtracking to the previous node will take place, after which all the other child nodes are explored. The modified variable-backtrack DFS proposed by Aggarwal et al. aims to create more diverse pairings from the start, and attempts to achieve this by introducing a step-length. Instead of backtracking to the previous node, the DFS algorithms will go to the node that is step-length away, counted from the root of the already explored branch. This process is executed with step-lengths from 0 to a finite limit, such that a set of pairings that covers at least all flights once is created. In order to make the search in very large search spaces even more efficient, Aggarwal et al. propose a second modification where the search space is reduced by using an additional constraint on a pairing's excess-pay. The excess-pay cost of a pairing is defined as its non-productive hours (i.e. waiting, extra long turn around periods, deadheads). The addition of an extra constraint is based on the expectation that good quality pairings will have a very small amount of excess-pay associated with them, and will thus result in the generation of better pairings in fewer time. The upper-bound of the new excess-pay constraint can additionally be varied, in order to obtain pairings with a larger variety in excess cost. Aggarwal et al. choose to execute their altered pairing generation process on multiple processors by first decomposing the large process into smaller independent subproblems. This division is made on the basis of crew base. The created subproblems can be run simultaneously on multiple processors to further reduce the runtime of the pairing generation process. This approach delivered results approximately 10 times faster than the sequential approach.

2.2.3. Branch-and-Price

Traditionally, the pairing problem has been solved optimally with the help of column generation and branch-and-bound [29][28][51][52]. Because the crew pairing problem is an example of a binary integer problem, column generation is often used to solve the relaxed problem, while branch-and-bound is used afterwards to obtain the optimal integer solution from the relaxed solution. The relaxed problem form of Equation 2.3 can be seen below in Equation 2.4. Column generation enables large linear programming problems to be solved faster by iteratively solving smaller problems, rather than the entire problem [20]. Initially, the problem is solved using only a subset of the decision variables, which are columns in matrix representation. This smaller problem is called the restricted master problem (RMP). After the RMP is solved, the columns that have a negative reduced cost, could potentially improve the current objective function, and are therefore added to the RMP. If this is the case, the RMP is solved again, until no new columns can be added. The result of the last run gives the optimal solution for the entire problem.

$$
\begin{aligned}
\min_{p \in P} \quad & \sum_{p \in P} c_p x_p & & & \\
\text{s.t.} \quad & \sum_{p : i \in p} x_p & \geq & 1 & \forall i \in F \\
& x_p & \geq & 0 & \forall p \in P \\
& x_p & \leq & 1 & \forall p \in P
\end{aligned}
\tag{2.4}
$$

To obtain an integer solution, branch-and-bound will use the relaxed solution as the root of its search tree. From the root of the tree, two child nodes are created by branching on a variable, chosen by using a branching heuristic (e.g. choosing the variable whose relaxed value is smallest). This variable is set to 0 on the first branch, and 1 on the second branch. For both child nodes, the objective value is calculated, and branching takes place from the node with the lowest objective value on the leftover non-integer variable whose value is closest to 0. This process is repeated until an integer solution has been found. The objective value of the first integer solution is saved as the current best integer solution, and this best integer solution value is updated throughout the process once an integer solution with a lower objective value has been found. The process of branching and calculating objective values repeats until there are no nodes left that have a lower objective value than the current best integer solution. Some branches will thus not be explored, as their relaxed objective value is already higher than the current best integer value. This helps speed up the process of obtaining an integer solution. When all branches have been explored or pruned, the optimal integer solution is represented by the current best integer solution [48].

An example of combining column generation and branch-and-bound is given by Yan et al. who solved the pairing problem for cabin crew, while also taking into account different cabin classes, mixed-aircraft types, and home bases [51]. Yan et al. formulated the crew pairing problem using flight networks, similar to Deng and Lin [18] and Ozdemir and Mohan [38] as explained in Section 2.1. To be able to use column generation, a heuristic is used to pick the initial columns. These are then used to construct the first restricted master problem, which is subsequently solved using the simplex method. Using the resulting dual variables, the flight arc costs are adjusted and the problem is divided into subproblems based on home bases and cabin class. These subproblems are formulated in such a way that the shortest path length in the resulting graph corresponds to the smallest reduced cost among all pairings of the subproblem. The columns corresponding to the shortest paths smaller than 0 are then added to the restricted master problem, and the process of solving the restricted master problem and adding columns repeats until there are no shortest-paths with cost lower than 0 left. If this relaxed solution is not integer, branch-and-bound is used to obtain the optimal integer solution. This entire process is called branch-and-price.

2.2.4. Benders Decomposition

Besides branch-and-price, Benders decomposition, and Lagrangian relaxation are techniques that are also frequently used. The goal of Benders decomposition is, in contrast to column generation where columns are generated dynamically, to generate rows dynamically. Benders decomposition is mostly used when the computation time grows greatly when the number of constraints increase [36]. The technique is also frequently combined with column generation and often reduces the number of iterations needed [39][17][36].

2.3. Metaheuristic approaches

Aiming to get good pairings for realistic problem sizes in less time, studies have also started to apply metaheuristic methods, such as genetic algorithms [38], general variable neighborhood search [4], and ant colony optimization [18]. The following sections describe how metaheuristics have been used to solve the crew pairing problem in the past, and how these approaches formulated the problem.

2.3.1. Problem Formulation

In contrast to most studies, Deng and Lin [18], and Ozdemir and Mohan [38], formulated the problem using a flight graph, as to facilitate the use of metaheuristic methods. In the flight graph, the nodes represent the flights that need to be covered, while a directed edge (a, b) exists between nodes a and b when flight b could be operated after flight a, taking the departure and arrival airports and times into account. The base where all pairings should begin and end are represented by connecting flights with the base as origin and destination with a source and sink node respectively. An example of such a flight graph can be seen in Figure 2.1. A final solution to the problem is represented as a set of paths from the source to the sink node that respects all additional constraints (e.g. regarding work time) and minimizes total cost [38].

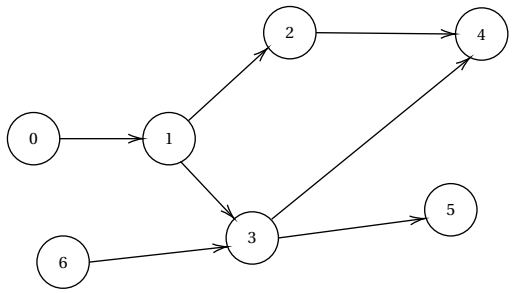


Figure 2.1: Example of a flight graph representing the given schedule

#	Departure City	Arrival City	Departure Time	Arrival Time
0	A	B	07:00	08:00
1	B	A	08:30	10:00
2	A	C	10:30	11:30
3	A	C	12:00	13:00
4	C	A	13:30	15:00
5	C	B	12:00	13:00
6	B	A	06:00	07:30

2.3.2. Ant Colony Optimization

Using the previously presented flight graph, Deng and Lin [18] solved the crew pairing problem using ant colony optimization. Their approach was inspired by the similarities between the crew pairing problem and the travelling salesman problem, which is also an NP-hard constrained combinatorial optimization problem. The authors formulated the crew pairing problem as a traveling salesman-like problem. This was done by building the flight graph with nodes representing flights and edges representing constraints between flights, and subsequently finding minimum cost paths by using ant colony optimization. Ant colony optimization is based on the idea of a set of workers, called ants, that explore the solution space and communicate promising search directions by means of pheromones. Each path then corresponds to a pairing in the solution. Deng and Lin found the ant colony optimization algorithm to be a promising solution technique, as it returns better solutions than a genetic algorithm in almost all, and especially large problem size cases [18].

2.3.3. Variable Neighborhood Search

Agustin et al. used variable neighborhood search to solve the crew pairing problem, with the aim of providing solutions of good quality, in reasonable time [4]. The algorithm takes an initial solution, the maximum number of neighborhoods that may be explored, and a maximum runtime. To start off, the initial solution is modified by randomly swapping flights between pairings, after which variable neighborhood descent (VND), is used to obtain a new solution. During VND, flights are swapped between pairings, while flights are also removed from some pairings and added to another. Finally, the initial and new solution are compared, and the best of the two is stored as the current best solution. If the new solution does not improve the old one, the random perturbation and VND is repeated until there is a better solution, or the maximum number of explored neighborhoods is reached. This approach showed a significant improvement both in the ability to find solutions, solution quality, and runtime, compared to a previously used metaheuristics for the same problem by the same authors [3][4]. Unfortunately, the research was based on test sets of a maximum of 100 flights. It therefore remains a question whether the proposed approach could actually run realistic instances in reasonable time.

2.3.4. Memetic Algorithms

One of the latest research efforts on metaheuristics by Deveci and Demirel used memetic algorithms, a combination of local search and genetic algorithms, to solve the crew pairing problem [22]. To start off, an initial solution population is generated using a heuristic to reduce the number of deadheads. From the initial population, two parents are chosen, after which two children are created by using one-point crossover on the parent solutions. The child solutions are then mutated by using a random bit-flip operation. To make sure the new solution still satisfies all constraints, the resulting children are repaired by greedily adding the pairing that costs the smallest per flight that it newly covers. To ensure that repairing the solution does not add any unnecessary cost, two local search procedures are used. The resulting solutions are the final child solutions, and replace the two worst solutions in the initial solution set. This process is repeated until a termination criterion is reached. Deveci and Demirel compared their algorithm to metaheuristic algorithms by Beasly and Chu [8] and Zeren and Özkol [53], and found that the memetic algorithm achieved better solutions while runtimes remained similar. Contrary to Agustin et al. [4] the study by Deveci and Demirel uses instances with sizes of up to 700 flights, and therefore resemble real-life problem sizes better.

2.4. Integrated Airline Scheduling

The sequential solving of different airline planning problems might deliver optimal results for each individual problem, but is not likely to produce a solution that is optimal for the entire airline planning problem [46]. This is due to the interdependence of the problems; the output of one problem acts as input for the next. As a result, more and more publications focus on integrated scheduling, where different planning problems are solved simultaneously. The next subsections will describe several approaches to integrate different airline scheduling problems.

2.4.1. Crew Pairing and Crew Rostering

As discussed before, the crew scheduling problem is often solved in two phases due to its complexity. A large advantage of solving the entire crew scheduling problem at once is that it eliminates the need for a separate pairing objective function, which as explained in Section 2.1 can be hard to design to resemble reality. In 2011, Saddoune et al. solved the entire crew scheduling problem using a combination of column generation and dynamic constraint aggregation [42]. Although great cost savings of approximately 4% are reported on real-life instances, the study fails to provide a new way to design crew schedules. The study does not take into account pre-assigned activities, such as safety training, simulator assignments, or holidays. Additionally, crew preference or a fair distribution of workload has not been incorporated into the model, thus resulting more into a calculation of the minimal number of crew members that are theoretically needed to execute a schedule than an actual scheduling tool. In a subsequent study from 2012, Saddoune et al. do include crew preferences for an airline using bidlines [43]. This model resembles reality a lot closer compared to their previous work. Although slightly smaller than in the 2011 study, Saddoune et al. achieve an average cost reduction of 3.37%, while 5.54% fewer pilots were needed to execute the schedule. This integral approach delivered good results, but did require significantly more computing time than previously used sequential approaches. To reduce computing time, other studies have focused on solving these problems using metaheuristic methods. Souai and Teghem, for example, developed a genetic algorithm to solve the pairing and rostering problem simultaneously [47]. Although the algorithm sometimes delivered better solutions than those proposed by the airline, other instances could not be solved because of the resulting integrated problem size. A more promising metaheuristic approach was proposed by Deng et al. in the form of ant colony optimization [18]. By using this method, Deng et al. achieved better results than the genetic algorithm by Ozdemir and Mohan [38] for many real-life instances.

2.4.2. Crew Pairing and Airline Routing

Another interesting approach to integrated scheduling includes combining the crew pairing problem with the aircraft routing model. By determining the crew pairings and aircraft routes at the same time, it is possible to generate new pairings that were illegal due to the extra time and costs associated with an aircraft change mid-pairing [46][13]. Sandhu and Klabjan designed two approaches to integrate the pairing and the aircraft routing problem [46]. The aircraft routing problem is simplified such that there is only a constraint on the maximum number of aircraft to be used, and maintenance requirements are ignored. To solve the integrated problem, both problems are combined and formulated as one large set partitioning problem. The first approach then uses column generation but pairs this with Lagrangian relaxation, while the second approach

uses Benders decomposition. In most cases, the Lagrangian relaxation performs better than the Benders decomposition. Other studies do consider maintenance aspects in the aircraft routing problem, to ensure that an optimal aircraft routing leaves enough time for aircraft maintenance. The solutions to these problems are therefore more likely to be practically feasible [45][23][39]. Diaz-Ramiréz, for example, includes maintenance in the problem, but only proposes a semi-integrated model that first requires the problems to run in a sequential fashion, after which the integrated problem is solved using column generation. Papadakos, on the other hand, provides an overview of how the crew pairing problem can be fully integrated with the maintenance and aircraft routing problems [39]. Because of the large number of constraints, Papadakos applies Benders decomposition and combines this with a column generation strategy that is accelerated by using heuristics. It is shown that a fully integrated approach delivers better results than the semi-integrated approaches, and could result in millions of dollars of cost savings each year for large airlines. A downside of the fully integrated approach is that it takes significantly longer to obtain solutions.

2.4.3. Retiming

Very new and little researched is the incorporation of schedule changes within crew pairing models. When the schedule is fixed, many useful pairings might not be generated because they violate a constraint by a small margin. By allowing changes to the initial flight schedule, it is possible to facilitate the design of more convenient pairings. An example of a small schedule modification is retiming: adjusting the departure and arrival time of a flight by a few minutes [14]. Klabjan et al. studied retiming flights in combination with the pairing and aircraft routing problem [32]. They proposed to solve the problems sequentially, but change the order such that the crew pairing problem with the option of retiming flights is solved first. Being able to retime flights improved the FTC with a factor 2. Cacchiani and Salazar-González do propose an entirely integrated model for the crew pairing, aircraft routing and fleet assignment problems including retiming [14]. The mixed-integer linear programming formulation includes retimed copies for all flight legs, and constraints to make sure only one of the original or copies is chosen for each flight leg. Given the integrated formulation, the relaxed problem is solved first, after which one of four proposed heuristics is used to reach a final solution. Again, it was shown that retiming flights can improve solution quality with at most 7% on large instances. They also conclude that offering retiming by 10 minutes resulted in better solutions than retiming by 5 minutes, while runtimes stayed similar. Dunbar et al. also investigate retiming flights but with an entirely different goal [24]. By integrating the crew pairing and aircraft routing problems with flight retiming, they aim to reduce the costs of propagated delay, and thus increase the robustness of the created schedules. The authors first solve the integrated aircraft routing and crew pairing problem, after which a heuristic method is used to decide which flights will be retimed. This approach resulted in a 14% reduction of the average delay propagation. This shows that flight retiming does not only have a great potential for creating lower cost schedules, but also for improving schedule robustness. To date, no studies have investigated other small flight schedule changes, such as swapping two flights. Seeing the potential of flight retiming, the incorporation of other small schedule changes offers an interesting avenue for further research.

3

Duty Generation

As explained in Section 1.1, crew pairings are made up of working days, called duties. While a pairing needs to start and end in the same base, a duty can start at any airport and end at any airport. Both pairings and duties need to abide by a number of complex rules concerned among others with the number of flight duty hours, number of duty hours, and starting time. For example, a duty that contains more than 13 flight duty hours may contain at most 3 landings, and if a duty begins before 04:30 local time, no more than 4 sequential flights may be assigned [1]. To be able to generate pairings during the solving process using branch-and-price, instead of generating a priori, it is desirable to represent the possible contents of a pairing in a network. As pairings consist of duties, which in their turn consist of flights, it is possible to construct this network of either flights or duties. Seeing that the combination of pairing and duty related rules is very complex to represent in a graph consisting of flights, it has been chosen to first generate duties, that will afterwards be used to construct the graph. For this project, it has been chosen to generate the duties by enumeration over a pruned search-tree, rather than using heuristics. This was done to ensure that the process results in a set of all feasible duties, which is desirable because the crew pairing problem is later formulated as a set partition, where every flight needs to be covered exactly once. Even though all duties are generated by enumeration, the process takes a few seconds at most for the tested instances. This chapter presents the algorithm by which the duties are generated, and gives a few examples of feasible duties.

During the duty generation process, it is the aim to find all possible sequences of flights that can legally be operated by a crew member in one workday. The resulting sequences are called *duties*. To generate these duties, a search tree containing all flights is enumerated in a breadth-first manner. The first layer of the tree consists of all flights in the schedule. At the beginning of the process, a stack that contains all nodes that still need to be explored is created. This entire first layer of nodes is added to the stack. Then, an iterative process takes place that terminates when the stack is empty. This process includes getting the first element, consisting of a path in the search tree, off the stack, and generating a duty including check-in and check-out activities based on the path. If this duty is legal, it will be appended to the list of all legal duties. Additionally, all flights that can be used to further extend the duty will be added as nodes in the tree. These nodes will be children of the last node of the path popped in the first step of the process. The last step of the iterative process is to add the paths from the root to the newly created nodes to the stack of unexplored options. Whenever a duty is not legal, no child nodes will be created, resulting in the branch to be pruned and not expanded upon further. An overview of this process can be seen in Algorithm 1, while specific duty rules that have been followed for this thesis can be obtained from Appendix A.

Algorithm 1 Duty generation

```
t ← new Tree
unexplored ← []
duties ← []
# All flights in the schedule are the first layer of children
for flight in schedule do
    flight_node ← new child node of t.root containing flight
    unexplored.append(flight_node)
end for
# Breadth-first search
while unexplored is not empty do
    node ← unexplored.pop() #Pop the first element of the stack
    duty_sequence ← path from node to root with added check-in and check-out activities
    if duty is legal then
        duties.append(duty)
        adjacent_flights ← list of flights that can be performed after last flight
        for a in adjacent_flights do
            node.add_child(a)
            unexplored.append(a)
        end for
    end if
end while
return duties
```

Below, Tables 3.1, 3.2, and 3.3 give examples of the duties that result from this approach. The columns indicate the start time of each activity, the IATA code of the departure airport, the activity, the IATA code of the arrival airport, the end time of the activity, and if applicable the corresponding aircraft type. Table 3.1 gives an example of a duty that starts and ends in the same base, and consists of four activities: a check-in, a flight from Amsterdam to Hurghada and back, and a check-out. Both flights are operated on a Boeing 737-800 aircraft. In contrast, Table 3.2 shows a duty that starts and ends in two different *outstation* (=non-base) airports. Again, this duty contains a check-in, check-out and two flights. The difference here is that an aircraft change takes place in Amsterdam, where the crew will switch from a Boeing 737-800 to a Boeing 737-700. Lastly, Table 3.3 shows a duty that contains only one flight, while departing from a base, and arriving in an outstation.

Table 3.1: Example of a duty starting and ending in the same base

01/03/2019 13:10	AMS	Check-in	AMS	01/03/2019 14:10	
01/03/2019 14:10	AMS	HV583	HRG	01/03/2019 19:05	B737-800
01/03/2019 20:00	HRG	HV584	AMS	02/03/2019 01:40	B737-800
02/03/2019 01:40	AMS	Check-out	AMS	02/03/2019 02:10	

Table 3.2: Example of a duty starting and ending at outstations, containing an aircraft change for crew

02/03/2019 10:40	LPA	Check-in	LPA	02/03/2019 11:40	
02/03/2019 11:40	LPA	HV5664	AMS	02/03/2019 16:20	B737-800
02/03/2019 18:25	AMS	HV5201	MUC	02/03/2019 19:55	B737-700
02/03/2019 19:55	MUC	Check-out	MUC	02/03/2019 20:25	

Table 3.3: Example of a duty with only one leg

02/03/2019 05:30	AMS	Check-in	AMS	02/03/2019 06:30	
02/03/2019 06:30	AMS	HV6901	DXB	02/03/2019 13:30	B737-800
02/03/2019 13:30	DXB	Check-out	DXB	02/03/2019 14:00	

4

Crew Pairing Optimization

After having described how all feasible duties can be generated from a flight schedule, this chapter describes how the resulting duties can be used to solve the crew pairing problem by presenting two models: a generate-and-test model `CP` and a branch-and-price model `CP_CG`. Both models use a pairing graph, consisting of previously generated duties, to generate the pairings that will be part of a solution to the crew pairing problem. Therefore, the construction of the pairing graph is discussed first, after which both models are explained in detail in later sections.

4.1. Pairing Graph

To generate feasible pairings, both models use a directed acyclic graph consisting of source and sink nodes (bases), nodes representing taxi movements, and nodes representing the duties from the duty generation step. The presence of an edge between nodes indicates that the activities can be performed sequentially. This resulting graph has $O(n)$ nodes and $O(n^2)$ edges. In the graph, a path between the source and sink nodes of the same base then gives a pairing. To later enforce a maximum pairing duration, the edges have a resource cost attribute, which indicates the time needed to perform the activities in a sequence. The edges also have a weight attribute. The weights are assigned based on the time that the crew is on duty, but is not operating a flight. In this way, a connection between two duties without a lot of excessive rest in between is preferred to a connection that contains a lot of idle time. In addition, the use of taxi's is discouraged by penalizing the edge with additional weight. In this way, a pairing that does not contain any positioning activities, is preferred to the same pairing with additional taxi activities. The cumulative weight of the edges in a path from sink to source gives the cost of the pairing to be used in the objective function. An example of a pairing graph with 3 bases and 4 duties is shown in Figure 4.1. The graph is constructed by first creating individual nodes for all duties, all sources and all sinks. Additionally, two taxi nodes are created for each base. One of these nodes indicates the possibility of a pairing starting with a taxi commute from the base, while the other indicates the possibility of ending a pairing with a taxi commute to the base. Similarly, two nodes per base are created representing the combination of a taxi plus overnight stay. Next, all edges are determined. The general steps are described on the following page, where a more precise overview including all edge weights can be found in Appendix B.

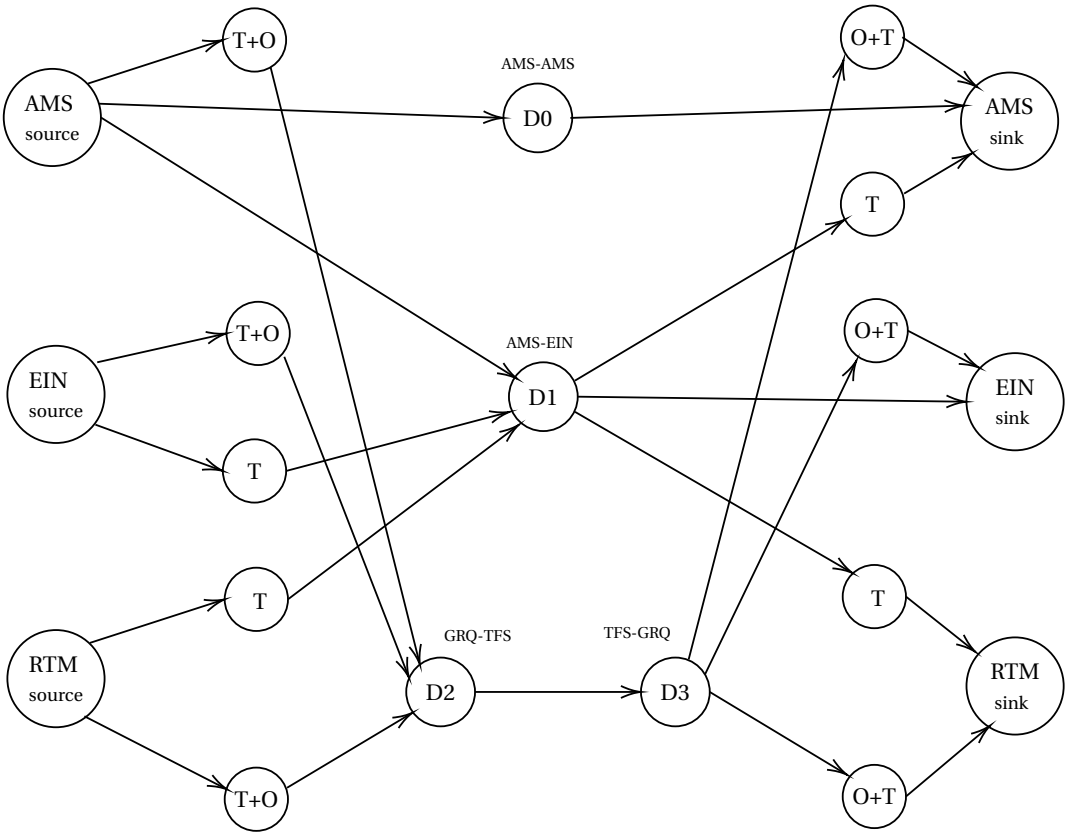


Figure 4.1: Example of a pairing graph containing 4 duties (D0-D3), 3 bases (AMS, RTM, EIN), their respective taxi nodes, indicated with 'T', and taxi+overnight nodes, indicated with 'O+T' and 'T+O'. Unused taxi and taxi + overnight stay nodes are not pictured for image clarity

- All duties starting from a base get an incoming edge from the corresponding base source node, and all duties ending in a base get an outgoing edge to the corresponding base sink node
- All duties starting from a base get an incoming edge from the taxi nodes to all other bases; except for when the duty starts and ends in the same base or the combination does not respect all rules.
- All taxi + overnight nodes corresponding to all other bases get an edge to a duty only if adding a taxi does not respect all rules; except for when the duty starts and ends in the same base or the combination does not respect all rules.
- All duties ending in a base get an outgoing edge to all the taxi nodes corresponding to all other bases; except for when the duty starts and ends in the same base or the combination does not respect all rules
- All duties ending in a base get an outgoing edge to all the taxi + overnight nodes corresponding to all other bases if adding only a taxi does not respect all rules; except for when the duty starts and ends in the same base or the combination does not respect all rules
- All duties that can be performed in sequence with another duties get an edge from the first duty to the following duty.
- All taxi nodes and taxi + overnight nodes get an edge to their corresponding source or sink node

An example of a pairing that can generated by using the graph from Figure 4.1 is given by the path,

$$\text{AMS source} \longrightarrow T+O \longrightarrow D2 \longrightarrow D3 \longrightarrow O+T \longrightarrow \text{AMS sink}$$

and translates to the following pairing:

07/03/2019 17:30	AMS	Taxi Check-in	AMS	07/03/2019 17:45
07/03/2019 17:45	AMS	Taxi	GRQ	07/03/2019 20:00
07/03/2019 20:00	AMS	Hotel	GRQ	08/03/2019 08:30
08/03/2019 08:30	GRQ	Check-in	GRQ	08/03/2019 09:30
08/03/2019 09:30	GRQ	HV5775	TFS	08/03/2019 14:15
08/03/2019 15:00	TFS	HV5776	GRQ	08/03/2019 19:40
08/03/2019 19:40	GRQ	Check-out	GRQ	08/03/2019 20:10
08/03/2019 20:10	AMS	Hotel	GRQ	09/03/2019 09:10
09/03/2019 09:10	GRQ	Taxi	AMS	09/03/2019 11:25

4.2. Generate-and-Test Approach

One way to optimize crew pairings is to first generate all possible pairings, and use these to solve a set partition integer linear program to find the optimal set of crew pairings. This section introduces such a generate-and-test model, named `CP`, after the abbreviation of Crew Pairing. Historically, these kinds of models have been used to solve small instances of the crew pairing problem. In the context of this thesis, this model will be used as a benchmark to compare other solution approaches to.

After the pairing graph has been constructed, the set of all possible pairings is generated by finding all simple paths between the source and sink nodes of the same base in the pairing graph. Only the pairings that do not violate the maximum pairing length are subsequently used as input. Afterwards, the model presented in Section 4.2.1 is solved using CPLEX. An overview of the steps taken during this approach are shown in Figure 4.2.

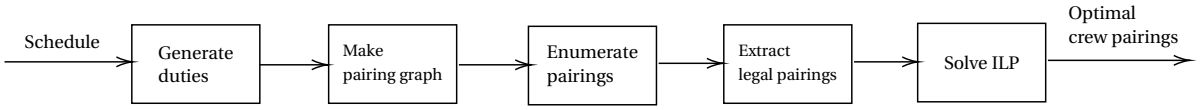


Figure 4.2: Diagram showing the solving process of model `CP`

4.2.1. Model Formulation

The formulation of model CP minimizes the cost of all chosen crew pairings (4.1), such that all flights are covered by exactly one pairing (4.2). Constraints 4.3 ensure the integrality of the problem.

$$\min \sum_{p \in P} c_p x_p \tag{4.1}$$

$$\text{s.t.} \sum_{p \in P} b_f^p x_p = 1 \qquad \forall f \in F \tag{4.2}$$

$$x_p \in \{0,1\} \qquad \forall p \in P \tag{4.3}$$

With:

Decision Variables

x_p	Binary decision variable that indicates whether pairing p is chosen

Sets

P	Set of pairings
F	Set of flights in the schedule

Constants

c_p	Cost of choosing pairing p
b_f^p	Binary constant indicating whether flight f is in pairing p

4.3. Branch-and-Price Approach

Although the CP model presented above returns a set of optimal crew pairings, a lot of memory and time is needed to enumerate all paths of the graph, create the integer linear program using a variable for every created pairing, and solve it. This is especially the case for large problem sizes. Instead of generating all pairings a priori, it is also possible to generate pairings that could possibly improve the solution while solving the problem. This can be done with the column generation approach that is visualized in Figure 4.3. To ensure the problem is always feasible, fake flight variables are used. Section 4.3.1 describes the mathematical formulation of this model, while Section 4.3.2 explains the corresponding pricing problem. Because the model solves a relaxed version of the crew pairing problem, this chapter will conclude with Section 4.3.3, which describes how integer solutions can be obtained by embedding column generation with branch-and-bound in a branch-and-price framework. As this model solves the crew pairing problem using column generation, it will later be referred to by the abbreviation CP_CG.

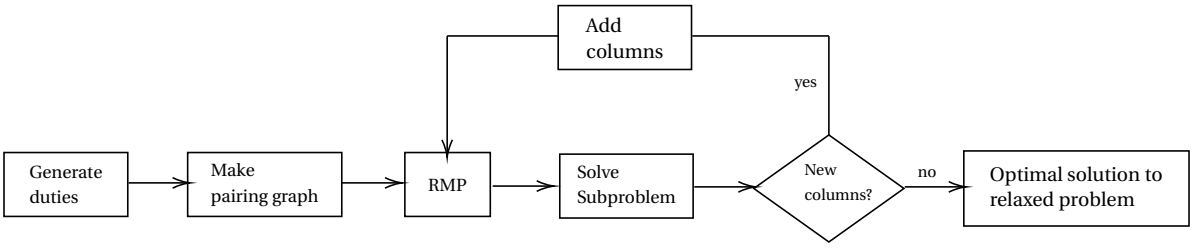


Figure 4.3: Diagram showing the solving process of model CP_CG

4.3.1. Restricted Master Problem

The restricted master problem is similar to the formulation presented in Section 4.2.1 for the CP model. The aim is still to minimize the cumulative cost of all chosen pairings (4.4), while covering all flights by exactly one pairing (4.5). However, the integrality constraints have now been relaxed to enable the use of column generation (4.6).

$$\min \sum_{p \in P} c_p x_p \tag{4.4}$$

$$\text{s.t.} \sum_{p \in P} b_f^p x_p = 1 \qquad \forall f \in F \qquad \alpha_f \tag{4.5}$$

$$x_p \geq 0 \qquad \forall p \in P \tag{4.6}$$

With:

Decision Variables

x_p	Binary decision variable that indicates whether pairing p is chosen

Sets

P	Set of pairings
F	Set of flights in the schedule

Constants

c_p	Cost of choosing pairing p
b_f^p	Binary constant indicating whether flight f is in pairing p

Dual Variables

α_f	Dual variables related to constraints 5.5

4.3.2. Pricing Problem Formulation

The pricing problem for this model is a shortest path problem in an updated pairing graph. The pairing graph is updated by taking a copy of the original pairing graph and using the dual variables α_f resulting from constraints (4.5) to update the edge weights. The following formula gives the reduced pairing costs.

$$\overline{c_p} = c_p - \sum_{f \in F} \alpha_f b_f^p$$

In the pairing graph, this translates to subtracting the dual variable corresponding to the constraint for each flight, from the edges that go into a duty node that contains the flight. Subsequently, a negative reduced cost column can be found by solving the shortest path problem on the updated graph. The source and target nodes for the shortest path problem are represented by source and sink nodes of the same base. This results in as many shortest paths as there are bases, after which the paths with negative reduced cost will be added to the restricted master problem.

The solving methodology for the shortest path problem depends on the number of days that a given flight schedule spans. If this number is smaller or equal to the maximum allowed pairing length, a regular shortest path problem will be solved by using the Bellman-Ford algorithm [9][26]. If the schedule spans more days than the maximum allowed pairing length, the problem will become a Shortest Path Problem with Resource Constraints (SPPRC). For this purpose, the resource cost edge attribute has been defined during the pairing graph generation (see Section 4.1). The sum of these resource costs in a path indicate how much time has passed since the beginning of the pairing. To find a shortest path that respects the maximum pairing duration, the maximum resource that can be consumed in a shortest path is set equal to the maximum pairing length. As the SPPRC is in itself NP-hard, it is first attempted to find negative reduced columns by using a heuristic approach, using a greedy elimination heuristic that eliminates edges that contribute to infeasible resource costs [50]. When no more negative reduced cost columns can be found by using the heuristic, an exact label-setting algorithm will determine whether there are no paths with negative reduced cost left [11]. This ensures that the approach still results in an optimal solution.

4.3.3. Finding an Integral Solution

In order to find an integral solution, the approach described above is combined with branch-and-bound. This combination of column generation and branch-and-bound is called branch-and-price. First, the relaxed problem is solved using column generation. If the variables in this solution are all integers, the solution to the relaxed problem is also the solution to the integer program. If the variables are not all integer, iterative steps of branching, calculating bounds and fathoming subproblems will take place. The current branching strategy is very simple; it finds the first non-integer decision variable, and creates two new problems: one where the variable's value is set to 0, and another where it is set to 1. The newly created problems will again be solved, and their bounds will be calculated. This process continues until all problems have been examined, after which the best integer solution is the optimal solution to the integer program.

5

Integrated Crew Pairing and Aircraft Routing

In order to find better crew pairings, it is also possible to solve the crew pairing problem simultaneously with the aircraft routing problem (see Section 2.4.2). This chapter describes two models where these two problems are integrated. As the input for the generate-and-test model consists of possible pairings and possible aircraft routes, these routes need to be generated from the flight schedule, before solving the model. Section 5.1 discusses how routing graphs can be constructed out of flight nodes, such that a path will represent one possible aircraft route. Next, the concept of crew-aircraft short connections, as an aid to avoid infeasible aircraft changes, is explained. Sections 5.3 and 5.4 thereafter present a generate-and-test and branch-and-price model to solve the integrated crew pairing and aircraft routing problem.

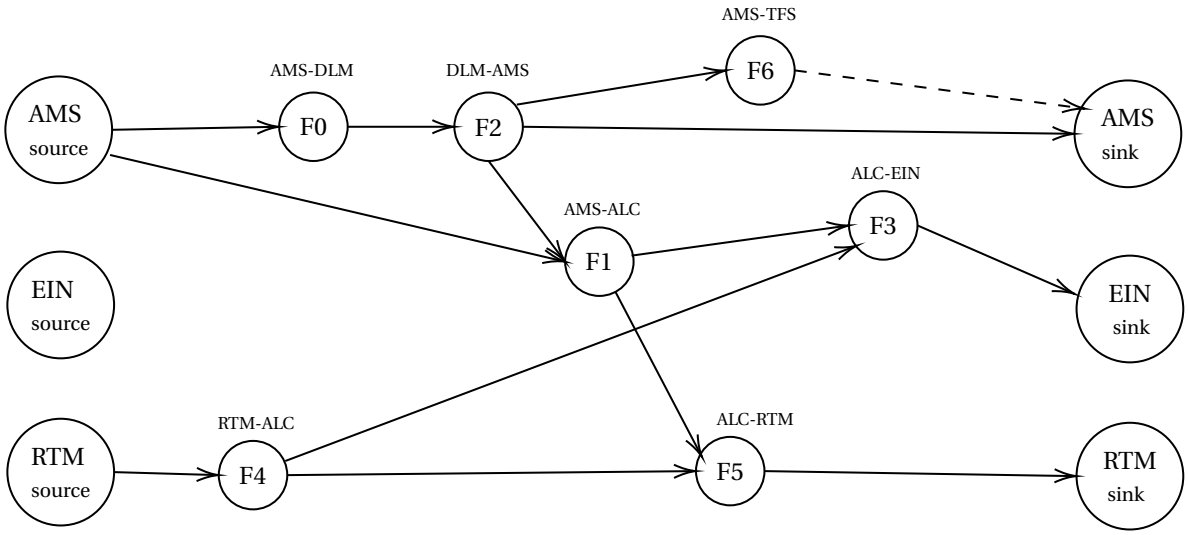


Figure 5.1: An example of a flight graph, with three bases and 7 flights, for one aircraft type

5.1. Routing Graphs

To generate aircraft routes, an approach similar to that used for pairing generation is used. In contrast to the pairing generation, there are fewer rules when generating aircraft routes. The turnaround times should be long enough, and the route should only include flights that can be operated by the same aircraft type, as each route represents one aircraft over (often) multiple days. To accomplish this, a separate directed acyclic graph for each aircraft type is constructed. An example of such a graph can be seen in Figure 5.1. The graph consists of source and sink nodes for each base, and nodes representing a flight. The edges are added as described below. This results in graphs that together have $O(n)$ nodes and $O(n^2)$ edges.

Appendix C gives a full overview of how the routing graphs are constructed, including corresponding edge weights. In the process of determining edge weights, a weight parameter w can be set to increase or decrease the importance of optimizing aircraft routes relative to crew pairings.

- All flights starting in a base, get an incoming edge from the source node of that base
- All flights arriving at a base, get an outgoing edge to the sink node of the corresponding base
- An edge (f_1, f_2) is constructed between two flight nodes f_1 and f_2, if the flight corresponding to node f_2 can legally be performed after flight f_1 by the same aircraft.
- To allow aircraft routes to end in an airport that is not a base, all nodes that represent a flight that does not end in a base get an outgoing edge to the first base in the list of bases. For an example see the edge from F5 to AMS sink in Figure 5.1.

In the graphs, a route is defined as a path from a source to a sink node. Contrary to pairings, aircraft do not have to start and end at the same airport. The cumulative weights of the edges in the path give the cost of the route. One possible route that can result from Figure 5.1 is represented by the following path:

$$\text{AMS source} \longrightarrow F0 \longrightarrow F2 \longrightarrow F1 \longrightarrow F3 \longrightarrow \text{EIN sink}$$

and will thus fly the following route:

$$\text{AMS} \longrightarrow \text{DLM} \longrightarrow \text{AMS} \longrightarrow \text{ALC} \longrightarrow \text{EIN}$$

5.2. Crew-Aircraft Short Connections

The crew pairings and aircraft routes are linked by the principle that crew need extra time when changing aircraft during a duty. In the case of Transavia, this parameter is set to 15 minutes. Before solving the integrated problem, a list of all short-connection flight combinations is made. This list can subsequently be used in the integrated models to enforce that if one of the flight combinations in the list is operated in the same pairing, it should also be operated in the same aircraft route. The list of short connections contains all combinations of two flights f_1 and f_2 for which the following holds:

- f_1 and f_2 are assigned to the same aircraft type
- The arrival airport of f_1 and the departure airport of f_2 are the same
- The time between the arrival of f_1 and the departure of f_2 is larger than the minimum required turn around time, but smaller than the minimum required turn around time + 15 minutes.

5.3. Generate-and-Test Approach

Similar to the process for the `CP` model described in Section 4.2, in this approach to solving the integrated crew pairing and aircraft routing problem, all pairings and routes are generated in advance. The pairings are generated exactly the same as for the `CP` model, by finding all simple paths between source and sink nodes of each base in the pairing graph. Similarly, all aircraft routes are generated by finding all simple paths between the source and sink nodes of each base for all routing graphs. These pairings and routes are all included in the integer linear program described in section 5.3.1. A list of short-connections is used to link the pairings and routes, to ensure that crew only change aircraft when there is enough extra time available.

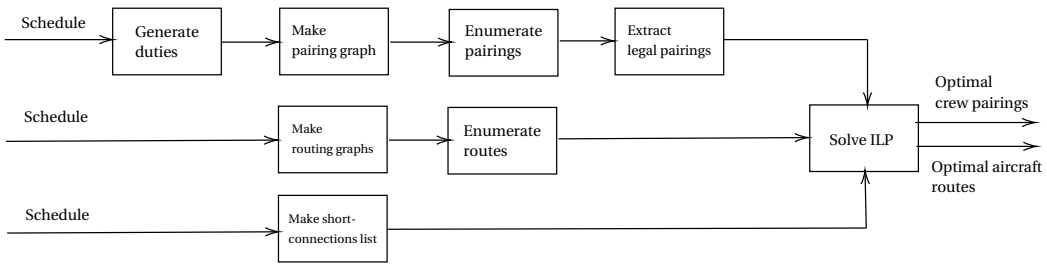


Figure 5.2: Diagram showing the solving process of model `CPIntegrated`

5.3.1. Model Formulation

Equation 5.1 represents the objective function to minimize the combined costs of the chosen pairings and routes. Equations 5.2 and 5.3 ensure that all flights are covered by one pairing and one route respectively. The flight combinations present in the set of short-connections can only be executed on the same pairing if they are also executed by the same aircraft route. This is expressed in equation 5.4, while equation 5.5 ensures that there are no more aircraft used than there are available of each type per base. Lastly, equations 5.6 and 5.7 ensure that the resulting solution is integer.

$$\min \sum_{p \in P} c_p x_p + \sum_{r \in R} c_r x_r \tag{5.1}$$

$$\text{s.t.} \sum_{p \in P} b_f^p x_p = 1 \qquad \forall f \in F \qquad \alpha_f \tag{5.2}$$

$$\sum_{r \in R} b_f^r x_r = 1 \qquad \forall f \in F \qquad \beta_f \tag{5.3}$$

$$\sum_{p \in P} b_{i,j}^p x_p - \sum_{r \in R} b_{i,j}^r x_r \leq 0 \qquad \forall (i,j) \in C \qquad \gamma_{i,j} \tag{5.4}$$

$$\sum_{r \in R} b_r^a b_r^{k^-} x_r \leq maxAC_a^k \qquad \forall k \in K, a \in A \qquad \delta_{a,k} \tag{5.5}$$

$$x_p \in \{0,1\} \qquad \forall p \in P \tag{5.6}$$

$$x_r \in \{0,1\} \qquad \forall r \in R \tag{5.7}$$

With:

Decision Variables

x_p	Binary decision variable that indicates whether pairing p is chosen
x_r	Binary decision variable that indicates whether route r is chosen

Sets

P	Set of pairings
R	Set of routes
K	Set of bases
A	Set of aircraft types
F	Set of flights in the schedule
C	Set of short connections

Constants

c_p	Cost of choosing pairing p
c_r	Cost of choosing route r
$MaxAC_a^k$	Number of available aircraft of type a at base k
b_f^p	Binary constant indicating whether flight f is in pairing p
b_f^r	Binary constant indicating whether flight f is in route r
b_r^a	Binary constant indicating whether route r is operated by aircraft type a
$b_f^{k^-}$	Binary constant indicating whether flight f starts in base a
$b_{i,j}^p$	Binary constant indicating whether flight i and j are in pairing p
$b_{i,j}^r$	Binary constant indicating whether flight i and j are in route r

Dual Variables

α_f	Dual variable corresponding to constraints 5.2
β_f	Dual variable corresponding to constraints 5.3
$\gamma_{i,j}$	Dual variable corresponding to constraints 5.4
$\delta_{a,k}$	Dual variable corresponding to constraints 5.5

5.4. Branch-and-Price Approach

Because generating all possible pairings, and especially all possible routes results in large problem sizes even when a small number of flights is considered, it is desirable to generate pairings and routes when they could potentially contribute to the optimal solution, rather than generating everything in advance. To do so, a branch-and-price approach and corresponding model have been designed. Similarly to the `CP_CG` model described in Section 4.3, the approach includes solving a relaxed version of the previously presented model using column generation, after which an optimal integer solution is found by using branch-and-bound in the same way as described in Section 4.3.3. An overview of the approach can be seen in Figure 5.3. This model will later be referred to as model `CpIntegrated_CG`.

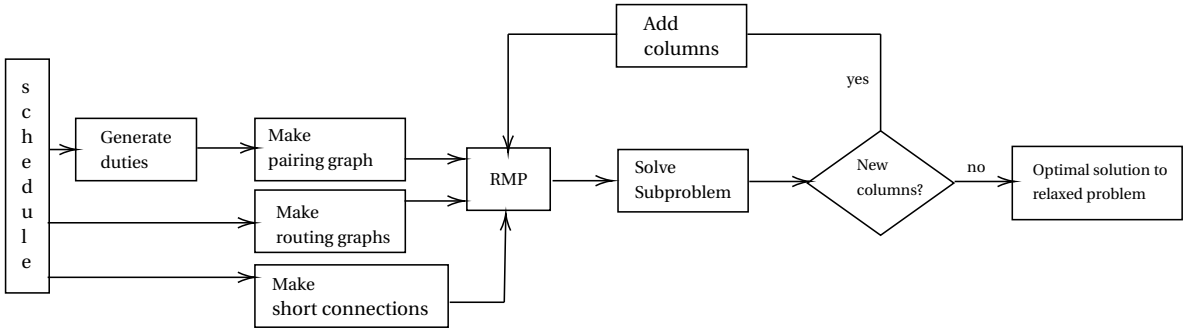


Figure 5.3: Diagram showing the solving process of model `CPIntegrated_CG`

5.4.1. Model Formulation

The mathematical formulation of the `CPIntegrated_CG` model is the relaxed version of the model presented in Section 5.3.1. To this end, equations 5.6 and 5.7 need to be replaced by the following relaxed versions:

$$0 \leq x_p \leq 1 \tag{5.8}$$

$$0 \leq x_r \leq 1 \tag{5.9}$$

5.4.2. Pricing Problem

The pricing problem for this model consists of solving multiple separate problems, namely a shortest path problem in the pairing graph, as well as shortest path problems for every routing graph. Before the shortest path problems can be solved, all graphs first need to be updated by using the values of the dual variables. The following subsections will first show how each graph type can be updated with the dual values, after which the method for finding negative reduced cost columns is explained.

Pairing Graph Updates

- The value of dual variables α_f (constraints 5.2) are subtracted from the weights of the edges incoming to duty nodes that contain flight f
- The value of dual variables $\gamma_{i,j}$ are subtracted from the weights of the edges incoming to duty nodes that contain both flights i and j

Routing Graphs Updates

- The value of dual variables β_f are subtracted from the weight of the incoming edge of the flight node representing flight f
- The value of dual variables $\gamma_{i,j}$ are added to the weight of the edge between the flight nodes representing flight i and j
- The value of dual variables $\delta_{a,k}$ are subtracted from the weights of the outgoing edges of the base source node for base k in the routing graph for aircraft type a

Finding Negative Reduced Cost Columns

After the pairing graph and all routing graphs have been updated using the dual variables, negative reduced cost columns are found for each graph. In the case of the pairing graph, a (resource-constrained) shortest path problem is solved similarly to the method described in Section 4.3.2. This results in a shortest path for each base, of which the ones with negative reduced cost are chosen to be added to the restricted master problem. For the routing graphs, the shortest path problem is solved for each possible base combination in each graph. This results in $|A| * |K| * |K-1|$ shortest paths, of which again the ones with negative reduced cost will be added to the restricted master problem.

6

Integrated Crew Pairing, Aircraft Routing, and Schedule Optimization

From previous studies on retiming and airline scheduling in hub-and-spoke networks, it is known that integrating schedule design with other scheduling problems can also have potential to improve crew pairings and aircraft routes [14][35]. By allowing the model to move flights in the schedule, a larger number of possible crew and aircraft connections can be formed, thus possibly resulting in more efficient pairings. This chapter will describe how flight retiming can be incorporated in the integrated crew pairing and aircraft routing problem, and how the problem can be solved using branch-and-price.

6.1. Flight Retiming

In the planning process, the airline acquires departure and arrival slots at the airport it wants to service. These slots indicate that the airline has the right to depart or arrive the airport within the time bracket corresponding to the slot. Per airport there are only a fixed number of slots available per time of day. For example, in the summer of 2019 at Amsterdam Airport Schiphol, the slot between 08:00 and 08:15 local time, has a capacity of 12 arrivals and 25 departures [37]. These slots are allocated twice a year; once for the summer season, and once for the winter season. Ideally the departure and arrival times stay within the acquired timebrackets after they have been retimed, as rescheduling flights outside of their slots may not be possible, or at least a lot of work. Unfortunately, because the slots and their regulations differ greatly between airports it is at this moment not possible to ensure that flights are always retimed within their slots. However, by allowing flights to move 5 minutes in the schedule, there is a relatively small probability that the flight's departure or arrival time will be outside the slot's timebracket. An example of the potential of retiming a flight by ± 5 minutes is shown in the flight graph in Figure 6.1. Flight 1 is allowed to be retimed with 5 minutes, therefore creating the additional possibilities to connect to flight 3, or flight 0.

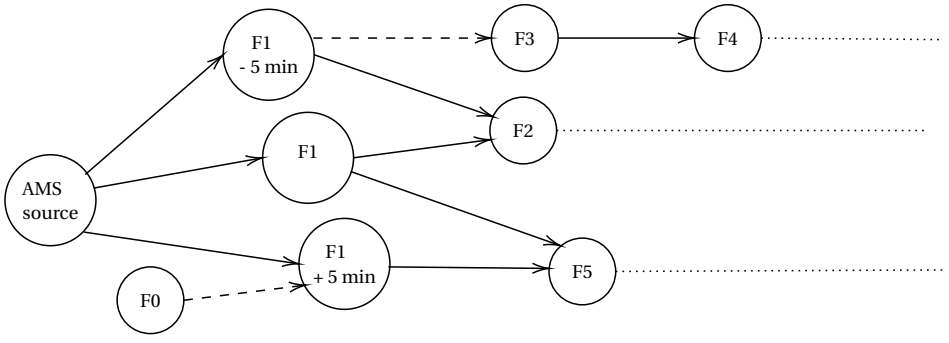


Figure 6.1: Flight graph showing the potential new connections, indicated by dashed edges, when allowing flight F1 to retime with ±5 minutes

To implement this 5 minute change in time, flight copies are created. These 'new' flight variables share the same properties as their original flight, but have a departure and arrival time that is shifted by 5 minutes. The following sections will describe how these new variables can be used in a branch-and-price approach to solve a schedule integrated model that will be referred to as model `CPIntegratedSchedule_CG`.

6.2. Branch-and-Price Approach

The solution approach to the integrated crew pairing, aircraft routing and schedule optimization problem is similar to the process of model `CpIntegrated_CG` presented in Section 5.4. The difference is that first, from the input schedule, all flight copies are created. These flight copies are from there on used to construct the duties, pairing graph, routing graphs and short connections. For each original flight a fake flight variable is created to ensure that the restricted master problem is always feasible. The subproblems are then solved as usual by updating the pairing graph and routing graphs, and solving (resource constrained) shortest path problems on the graphs. The model formulation is described in Section 6.2.1, while the pricing problem is described in Section 6.2.2. A complete overview of how the relaxed problem is solved can be found in Figure 6.2.

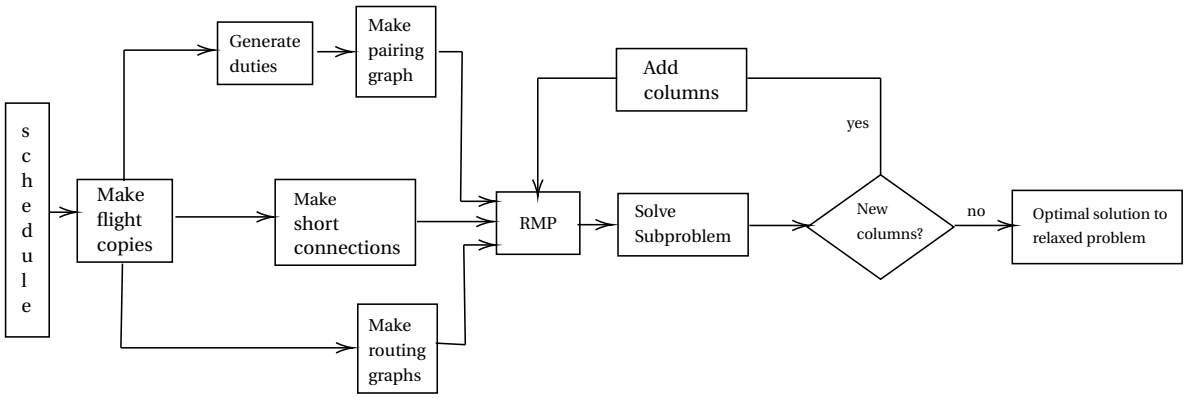


Figure 6.2: Diagram showing the solving procedure for the `CpIntegratedScheduleCG` model

After the relaxed problem is solved, it is checked whether the returned optimal solution consists of all integer values. If this is the case, the optimal solution to the relaxed problem is also the optimal solution to the integer problem. If this is not the case, the problem is embedded in a branch-and-price construction to find the optimal integer solution. This is again done by branching on the first non-integer variable, calculating bounds, and fathoming unpromising branches. During the process of calculating a bound, it is possible that new negative reduced cost columns are found; these are added in the same way as described for the relaxed problem, until no more columns can improve the solution. In the end, when all branches have been explored or fathomed, the integer solution with the lowest objective value is returned as the optimal solution to the integer problem. The final integer solution will, next to the optimal crew pairings and routes, give what flight copies should be executed, and thus how the flights should be retimed.

6.2.1. Model Formulation

The currently presented model slightly differs from the aircraft routing and crew pairing integrated model in Section 5.3.1. Constraints 6.2, 6.3, and 6.4 were adapted to include multiple possible departure times per flight. Constraints 6.5, 6.7, and 6.8 remained the same, while constraints 6.6 were added to ensure that the same departure time for each flight was chosen in both the pairing and routing problem.

$$\min \sum_{p \in P} c_p x_p + \sum_{r \in R} c_r x_r \tag{6.1}$$

$$\text{s.t.} \sum_{t \in D_f} \sum_{p \in P} b^p_{f_t} x_p = 1 \qquad \forall f \in F \qquad \alpha_f \tag{6.2}$$

$$\sum_{t \in D_f} \sum_{r \in R} b^r_{f_t} x_r = 1 \qquad \forall f \in F \qquad \beta_f \tag{6.3}$$

$$\sum_{p \in P} b^p_{i_v j_w} x_p - \sum_{r \in R} b^r_{i_v j_w} x_r \leq 0 \qquad \forall (i,j) \in C, \quad (v,w) \in C_{i,j} \qquad \gamma_{i_v, j_w} \tag{6.4}$$

$$\sum_{r \in R} b^a_r b^{k^-}_r x_r \leq maxAC^k_a \qquad \forall k \in K, a \in A \qquad \delta_{a,k} \tag{6.5}$$

$$\sum_{p \in P} b^p_{f_t} x_p - \sum_{r \in R} b^r_{f_t} x_r = 0 \qquad \forall f \in F, t \in D_f \qquad \epsilon_{f,t} \tag{6.6}$$

$$x_p \in \{0,1\} \qquad \forall p \in P \tag{6.7}$$

$$x_r \in \{0,1\} \qquad \forall r \in R \tag{6.8}$$

With:

Decision Variables

x_p	Binary decision variable that indicates whether pairing p is chosen
x_r	Binary decision variable that indicates whether route r is chosen

Sets

P	Set of pairings
R	Set of routes
K	Set of bases
A	Set of aircraft types
F	Set of flights in the schedule
D_f	Set of departure times for flight f
C	Set of short connections
$C_{i,j}$	Set of departure time combinations (v,w) where v belongs to flight $i \in C$, and w belongs to $j \in C$

Constants

c_p	Cost of choosing pairing p
c_r	Cost of choosing route r
$MaxAC^k_a$	Number of available aircraft of type a at base k
$b^p_{f_t}$	Binary constant indicating whether flight f with departure time t is in pairing p
$b^r_{f_t}$	Binary constant indicating whether flight f with departure time t is in route r
b^a_r	Binary constant indicating whether route r is operated by aircraft type a
$b^{k^-}_r$	Binary constant indicating whether route r starts in base a
$b^p_{i,j}$	Binary constant indicating whether flight i and j are in pairing p
$b^r_{i,j}$	Binary constant indicating whether flight i and j are in route r
$b^p_{i_v j_w}$	Binary constant indicating whether flight i with departure time v and j with departure time w are in pairing p
$b^r_{i_v j_w}$	Binary constant indicating whether flight i with departure time v and j with departure time w are in route r

Dual Variables

α_f	Dual variables for constraints 6.2
β_f	Dual variables for constraints 6.3
γ_{i_v, j_w}	Dual variables for constraints 6.4
δ_a, k	Dual variables for constraints 6.5
$\epsilon_{f,t}$	Dual variables for constraints 6.6

6.2.2. Pricing Problem

The pricing problem for this model is similar to the one defined for the aircraft routing and crew pairing integrated model. All graphs are constructed in the same way, but will contain a larger number of nodes and edges due to the increased number of possible connections. The size of the rotation graphs will now be in the order of $O(|t| * n)$ nodes and $O(|t| * n^2)$ edges, where $|t|$ indicates the number of possible departure times per flight, and n is the number of flights. This is because there exists a node for every flight-departure time combination. The pairing graph will also see an increase in the number of nodes, as more duties can be created with the n different departure times per flight. Similar to the integrated aircraft routing and crew pairing optimization model `CpIntegrated_CG`, the pairing graph and all routing graphs first need to be updated by using dual variables, after which (resource constrained) shortest path problems are solved exactly as described in Section 5.4.2. The process of updating the pairing and routing graphs' weights is described below.

Pairing Graph Updates

- The values of dual variables α_f, corresponding to primal constraints 6.2, are subtracted from the weights of the edges incoming to duty nodes that contain a flight copy of flight f
- The values of dual variables γ_{i_v,j_w}, corresponding to primal constraints 6.4, are subtracted from the weights of the edges incoming to duty nodes that contain both flight i with departure time v and flight j with departure time w
- The values of dual variables $\epsilon_{f,t}$, corresponding to primal constraints 6.6, are subtracted from the weights of the edges incoming to duty nodes that contain flight f with departure time t

Routing Graph Updates

- The values of dual variables β_f, corresponding to primal constraints 6.3, are subtracted from the weights of the edges incoming to the flight node representing a copy of flight f
- The values of dual variables γ_{i_v,j_w}, corresponding to primal constraints 6.4, are added to the weights of the edges between flight nodes representing flight i with departure time v and flight j with departure time w
- The values of dual variables $\delta_{a,k}$, corresponding to primal constraints 6.5, are subtracted from the weight of outgoing edges from the base source node of base k in the routing graph of aircraft type a
- The value of dual variables $\epsilon_{f,t}$, corresponding to primal constraints 6.6, are added to the weights of the edges incoming to flight nodes that represent flight f with departure time t

7

Experiments

To evaluate the models presented in the previous chapters, different experiments are designed. This chapter describes the datasets used to test the models on, after which the following sections will give an overview of the different models and parameters used.

7.1. Data

The input for all models include a flight schedule consisting of legs that need to be supplied with crew and an aircraft. To later be able to compare the outcome of the currently presented crew pairing models and Transavia's pairings, it is important to select a range of flights that enables a comparison that is as fair as possible. To this end, the selection of flights took place by looking at the starting date of Transavia's pairings. For example, for a "one-day" dataset, all flights that were performed in Transavia's pairings starting from that day were chosen. This avoids that only half of the flights of a pairing made by Transavia are in the current dataset, therefore enabling a more fair comparison of results. The set of all datasets can be seen in Table 7.1. The timespan indicates how many pairing starting days were used to get flights from, while the date range indicates between which dates the chosen flights fall.

Table 7.1: Overview of all data sets and their characteristics

Dataset Number	**Dataset Name**	**# Flights**	**Timespan**	**Date Range**	**SPPRC**
1	1_day	142	1 day	1-2 March 2019	×
2	2_day	282	2 days	1-5 March 2019	✓
3	3_day	438	3 days	1-5 March 2019	✓
4	4_day	567	4 days	1-6 March 2019	✓
5	5_day	678	5 days	1-7 March 2019	✓
6	6_day	789	6 days	1-7 March 2019	✓
7	7_day	926	7 days	1-9 March 2019	✓

7.2. Models

The characteristics of all models presented in previous chapters are briefly summarized in Table 7.2. For each model it is described by what name they are referenced, the section that provides a full explanation on their workings, whether they include the crew pairing and aircraft routing problems, and whether it allows retiming. Lastly, it is also indicated whether the model is solved using a branch-and-price approach.

Table 7.2: Overview of all implemented models and their characteristics

Model Number	Model Name	Section	Crew Pairings	Aircraft Routes	Retiming	Branch-and-Price
1	CP	4.2	✓	×	×	×
2	CP_CG	4.3	✓	×	×	✓
3	CP_Integrated	5.3	✓	✓	×	×
4	CP_Integrated_CG	5.4	✓	✓	×	✓
5	CP_Retiming_CG	6.2	✓	✓	✓	✓

7.3. Standard Parameters

Unless mentioned otherwise, all models use a set of standard parameters for the presented experiments. Table 7.3 gives an overview of these parameters.

Table 7.3: Parameters and their standard values used for the presented experiments

Parameter	Explanation	Value
`Check-in time`	Time needed for check-in activity	1 hour
`Check-out time`	Time needed for check-out activity	30 minutes
`Taxi check-in time`	Time needed for check-in before a taxi activity	15 minutes
`Default turn-around time`	Turn-around time used when no time is found for the airport and aircraft combination	40 minutes
`Maximum turn-around time`	Maximum time between two sequential flights in one duty	3 hours
`Extra time aircraft change`	The extra time needed when crew needs to change aircraft	15 minutes
`Maximum pairing length`	The maximum time a pairing covers	4 days
`Maximum layover time`	Maximum time between two duties in a pairing	3 days
`w`	Importance of optimizing aircraft routes relative to pairings. Same importance if `w` = 1	0.1

7.4. Experiments

From the research questions set for this project, and the presented models, three important aspects can be derived: solution quality, practical implications and runtime. The following subsections describe three experiments performed to investigate these aspects.

7.4.1. Solution Quality

To answer the research question posed in Section 1.3, it is important to understand the impact of the different models on the total objective value. Important Key Performance Indicators (KPIs) for this purpose will be the objective value of the integer and linear problems, and the resulting integrality gap. Additionally, to be able to compare the different model outcomes, it is important to look at the objective values. To compare the branch-and-price against the generate-and-test approach for efficiency, it is interesting to compare the number of possible pairings and routes, with the number of generated and used pairings and routes. The hypothesis for the solution quality aspect is as follows:

- The integrated models `CPIntegrated_CG` and `CPIntegrated` will have a higher objective value than the retiming-integrated model `CPIntegratedSchedule_CG`.

7.4.2. Practical

As the aim of this thesis is to achieve an improvement in the crew productivity, it is important to investigate whether the models return more productive solutions than Transavia designed for the given flight schedules. To achieve this, the results of all models on all data sets will be compared to the pairings used by Transavia for the exact set of flights. One important KPI is the average block hours/duty hours per duty over all pairings as this gives a measure for the crew productivity. Additionally, the average number of duties needed to cover all

flights is important because it gives a lower bound on the number of crew needed each day to operate a flight schedule. It will also be interesting to see whether more layovers will be implemented, as this might increase the crew productivity, but is currently not a favorable option because of the costs associated with them. The hypotheses for the practical aspect are as follows:

- It is expected that all models will return an improvement in crew productivity as measured by the average block hours per duty hours ratio, compared to Transavia's results.
- It is expected that the integrated models will return a larger improvement compared to the non-integrated models.

7.4.3. Runtime

Lastly, the runtime is of great importance when deciding whether to use the approaches presented in this thesis. Therefore, the runtime of all models will be examined for all datasets by recording the total runtime and the time needed to find a solution to the linear relaxation. The hypotheses for the solution quality aspect are as follows:

- Although it is expected that the more integrated models will provide better solutions, it is also expected that the more a model is integrated, the longer its runtime will be for the same data set.
- The branch-and-price methodology will be slower than the generate-and-test approach for small data sets due to the large number of steps, but faster for larger data sets.

7.5. Experimental Environment

The models were implemented in Python 3.6, using packages CSPY and Pylgrim to implement the solving strategy of the pricing problems [49] [50]. All experiments were run on a machine with an 8-core 2.40GHz Intel® Xeon® Gold 6148 processor and 32GB of RAM, running on Red Hat Enterprise Linux 7.6.

8

Results

This chapter discusses the results obtained from the runs described in the previous chapter. Section 8.1 will provide the results of the pairing and routing graph generation, while Section 8.2 discusses the problem sizes in terms of the decision variables. Lastly, Section 8.3 provides the results of all experiments.

8.1. Pairing and Routing Graphs

To give an impression of the size of the shortest path problems that are solved for each experiment, Table 8.1 gives an overview of the pairing and routing graph sizes in terms of the number of edges and the number of nodes for each data set.

Table 8.1: Specifications of the pairing and routing graphs created for each dataset. In the case of routing graphs, all numbers are reported as a sum over all routing graphs. † indicates that not enough memory was available to complete the runs, while $*$ indicates that the runs took longer than 24 hours to complete

	Pairing graph			**Routing graphs**			**Pairing graph (retimed)**			**Routing graphs (retimed)**		
Dataset	# nodes	# edges	mean degree	# nodes	# edges	mean degree	# nodes	# edges	mean degree	# nodes	# edges	mean degree
1_day	130	276	2.12	154	599	3.89	3300	9015	2.73	438	3963	9.05
2_day	386	1375	3.56	294	2713	9.23	$*$	$*$	×	$*$	$*$	×
3_day	832	4638	5.57	†	†	×	$*$	$*$	×	$*$	$*$	×
4_day	1329	10535	7.93	†	†	×	$*$	$*$	×	$*$	$*$	×
5_day	1703	15721	9.23	†	†	×	$*$	$*$	×	$*$	$*$	×
6_day	2075	21429	10.33	†	†	×	$*$	$*$	×	$*$	$*$	×
7_day	2488	28332	11.39	†	†	×	$*$	$*$	×	$*$	$*$	×

It can be observed that for each graph the number of nodes and edges increases as the dataset covers more days and flights. Additionally, the retimed pairing graph contains more nodes and edges than the regular pairing graph as a result of the larger set of possible duties due to retiming. It is interesting to see that next to the number of nodes and edges, the mean degree in the retimed pairing graph is also larger than the regular pairing graph for the same data set. This indicates that the retiming results in more possible paths. The same observations can be made for the retimed routing graphs.

8.2. Decision Variables

To investigate the size of the linear problems it is interesting to look at the number of decision variables per model and dataset. This is especially relevant to investigate the difference in problem size between the generate-and-test and branch-and-price approaches, as the aim of column generation is to solve smaller problems and add decision variables only as needed. Table 8.2 shows the average number of decision variables generated during the entire model per model and dataset, over 24 runs per model - dataset combination.

Table 8.2: Average number of decision variables per problem (n = 24). † indicates that the runs did not finish because of not enough memory was available, while ∗ indicates that the run did not finish within 24 hours

Data set	CP	CP_CG	Difference	CP-Integrated	CP-Integrated-CG	Difference	CPIntegrated-Schedule_CG
1_day	298	221	*-25.8%*	1861	704	*-37.8%*	1927
2_day	1607	506	*-68.5%*	330586	4308	*-98.7%*	∗
3_day	11298	921	*-91.8%*	†	∗	×	∗
4_day	63533	1397	*-97.8%*	†	∗	×	∗
5_day	91789	2045	*-97.7%*	†	∗	×	∗
6_day	122099	2697	*-97.8%*	†	∗	×	∗
7_day	150046	3244	*-97.8%*	†	∗	×	∗

It can be seen from Figure 8.1 that the number of decision variables of model CP behaves like a piecewise linear function, consisting of 3 segments: 1_day, 2-3_day, 4-7_day. The structure of this function could be related to the maximum pairing length parameter set to 4 days, and the timespan of the given datasets. Furthermore, as can be observed from Table 8.2, the integrated models use more decision variables, as there is one per crew pairing and one per aircraft route. With regards to the difference between the generate-and-test and branch-and-price approaches, it can be seen that the branch and price approaches use a significantly smaller number of decision variables. For example, CPIntegrated_CG only uses 1.3% of the decision variables compared to CPIntegrated. Generally, the percentage of decision variables that do not have to be generated increase with the size of the dataset. In addition, it can be seen that there is a larger percentual difference for the integrated models.

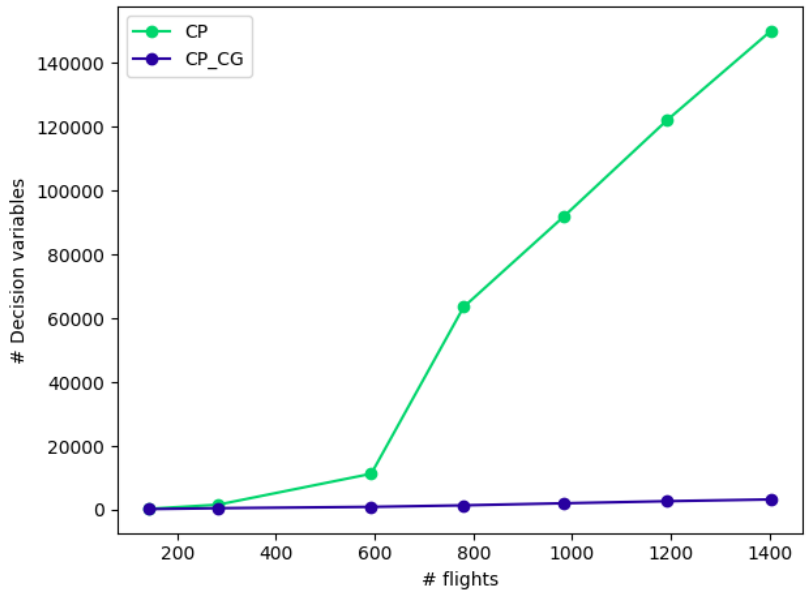


Figure 8.1: Number of decision variables for models CP and CP_CG over problem size

8.3. Experiments

This section describes the results of the experiments presented in Chapter 7. The first section focuses on the solution quality, and how it develops over dataset size and across different models. Afterwards, a comparison will be made between the results obtained from the different models and Transavia's current methodology. Lastly, the runtime performance of all models is discussed in Section 8.3.3

8.3.1. Solution Quality

The solution quality of the presented models on the different datasets is presented in Table 8.3 and visualized in Figure 8.2. It can be seen from the figure that the objective value of models `CP` and `CP_CG`, and models `CPIntegrated` and `CPIntegrated_CG` are equal to each other. This is an indicator of the correctness of the column generation technique in the branch-and-price approaches, as the branch-and-price algorithm should not alter the exact outcome of the problem, but only decrease the number of decision variables. In addition, the `CPIntegrated` and `CPIntegrated_CG` models have a higher objective value than the non-integrated models, due to the inclusion of routing costs. As expected, the retiming model `CPIntegratedSchedule_CG` greatly reduces the objective value compared to the other integrated models. Interesting to note from Table 8.3 is that the solution for the linear problem is equal to the solution value for the integer problem. With the integrated models, this is not the case, but the integrality gap is still high at a lowest value of 0.99.

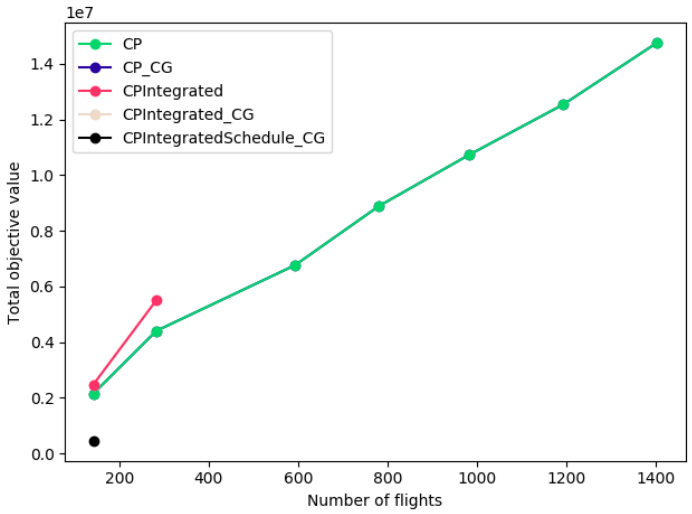


Figure 8.2: Objective values for all models over all data sets

Figure 8.3 displays the value of the objective function over the column generation iterations throughout the branch-and-price process. It can be seen that the objective value greatly decreases in the first 100 iterations, after which the solution to the linear relaxation is found around iteration 175. From the 200th iteration onwards, small spikes in objective value can be observed. These spikes are caused by the branch-and-bound process, where a variable's value is fixed while keeping the existing decision variables. These peaks often get lower because additional columns are added, or the solution is fathomed.

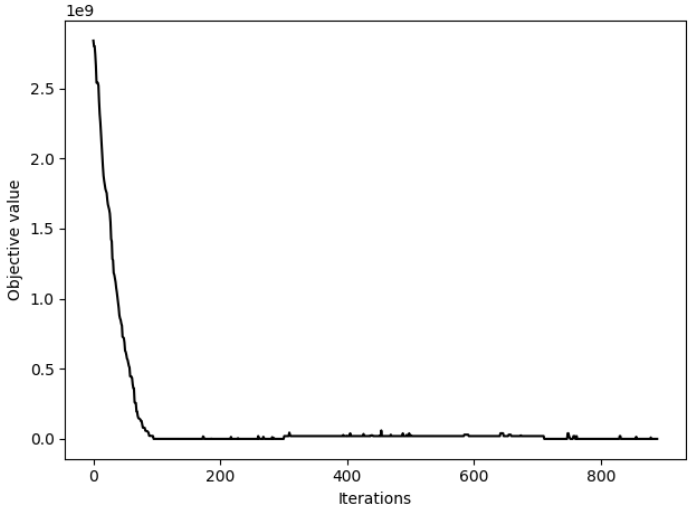


Figure 8.3: Objective value of model `CPIntegratedSchedule_CG` on the 1_day dataset per iteration

Table 8.3: Objective values for the integer and linear problem associated to the different models and data sets, with the average number of column generation iterations (n = 24). † indicates runs were not completed because not enough memory was available, while ∗ indicates that the run did not finish within 24 hours

	CP	CP_CG				CPIntegrated	CPIntegrated_CG				CPIntegratedSchedule_CG			
Data set	IP	LP	IP	Gap	Iterations	IP	LP	IP	Gap	Iterations	LP	IP	Gap	Iterations
1_day	2141700	2141700	2141700	1.0	42	2469180	2469180	2469180	1.0	93	447779.99	447780.00	0.99	401
2_day	4400700	4400700	4400700	1.0	101	5501370	5501369.99	5501370	0.99	926	∗	∗	×	×
3_day	6763200	6763200	6763200	1.0	203	†	∗	∗	×	×	∗	∗	×	×
4_day	8883000	8883000	8883000	1.0	369	†	∗	∗	×	×	∗	∗	×	×
5_day	10729500	10729500	10729500	1.0	647.54	†	∗	∗	×	×	∗	∗	×	×
6_day	12535200	12535200	12535200	1.0	898.875	†	∗	∗	×	×	∗	∗	×	×
7_day	14759700	14759700	14759700	1.0	1114.125	†	∗	∗	×	×	∗	∗	×	×

8.3.2. Practical

This section focuses on the practical value of the presented models. Figure 8.4 shows a visualization of the pairings resulting from running model CP on the 2_day dataset. Each horizontal value represents one pairing, where the green color indicates a flight, blue indicates a check-in or check-out activity, pink is for taxi activities and grey represents a hotel stay. It can be seen that mostly pairings consisting of only one duty are formed; only two pairings with layovers are necessary.

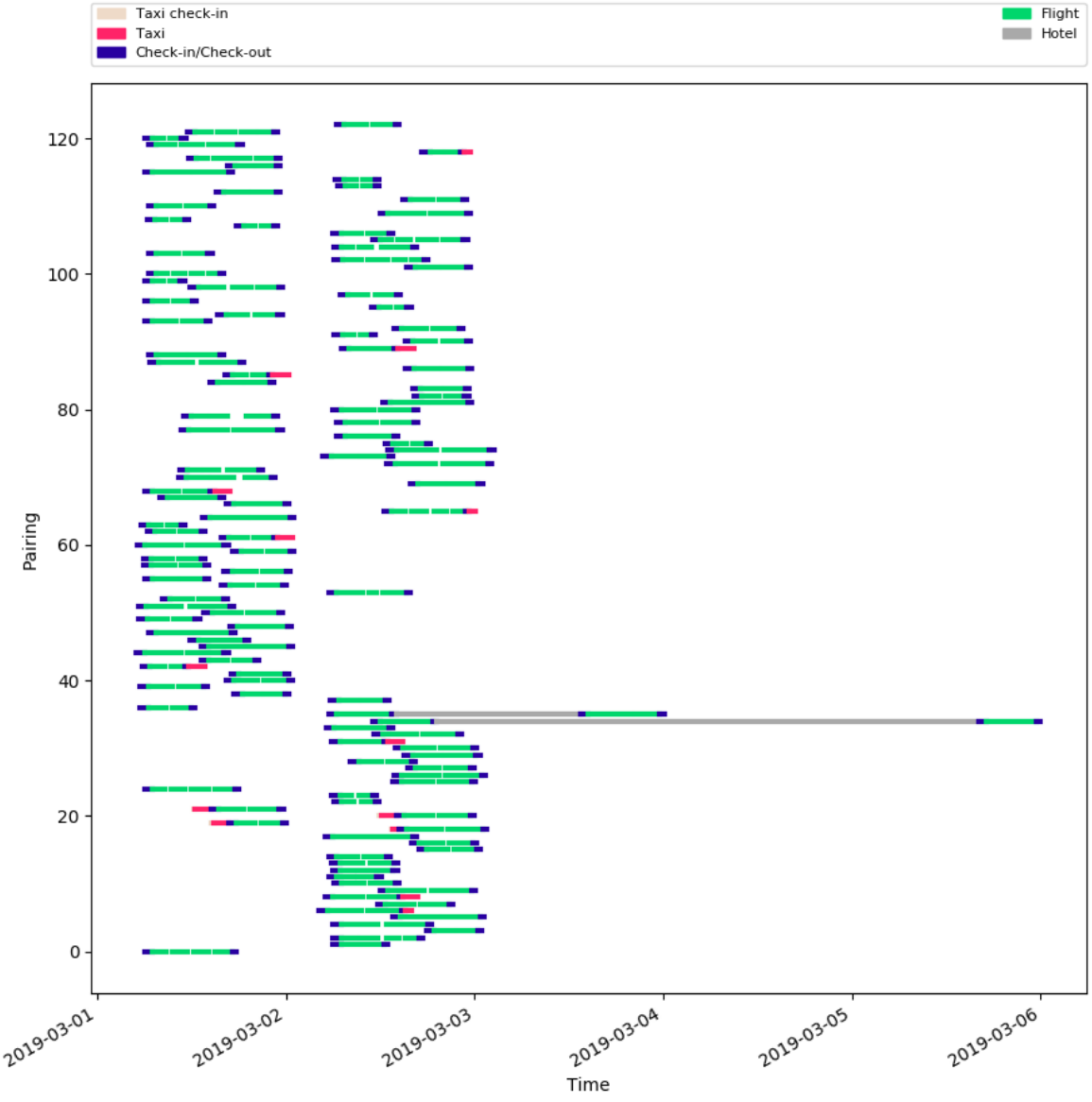


Figure 8.4: Visualization of the pairings resulting from model CP ran on the 2_day data set, where each horizontal line at the y-axis represents a pairing

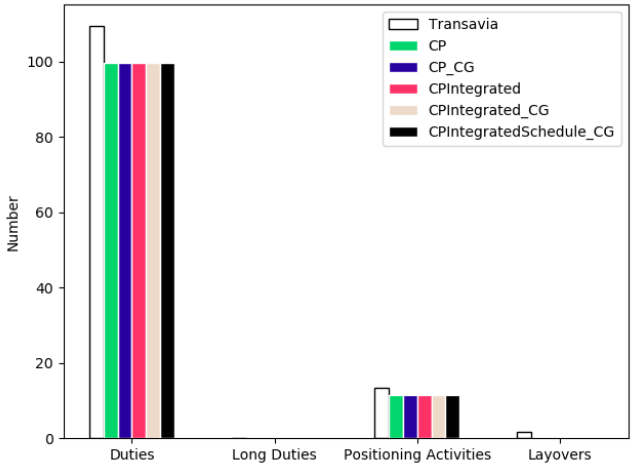
Figure 8.5 shows how the results from the different models compare to Transavia's current pairings, in terms of the number of (long) duties, positioning activities, layovers, and the block hours over duty period (BH/DP). All these statistics have been multiplied by a factor to cover their real values. This has been done to protect the sensitivity of the results, while still being able to show the relationship between them. As can be seen from Figures 8.5a, 8.5b and 8.5c, generally all models over the 1, 2 and 7_day datasets result in a higher number of duties and fewer positioning. Additionally, the results for datasets 1_day and 2_day also contain fewer layovers. This is a positive outcome, as the number of duties gives a lower bound on the number of crew members that are needed to cover a schedule, and positioning and layover activities cost time and money while the crew are at that moment not operating flights.

The hashed statistics corresponding to these observations can be found in Table 8.4. It can be seen that all models have a higher block hour per duty hour ratio.

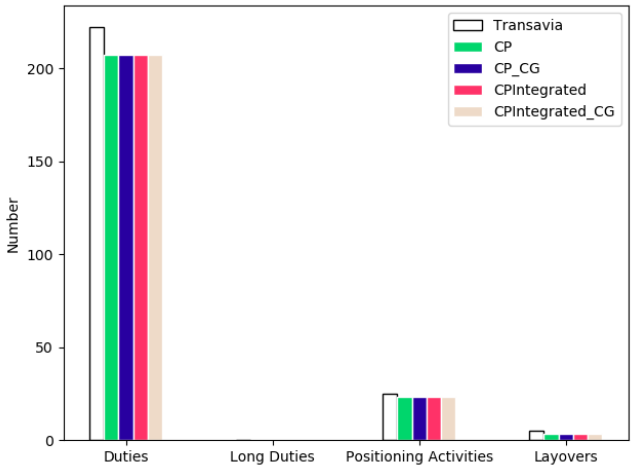
Table 8.4: Hashed practical results of the models over the 1, 2 and 7_day datasets. † indicates that the run was not completed due to too little memory, while ∗ indicates that the run did not complete within 24 hours

Dataset	Metric	Transavia	CP	CP_CG	CP-Integrated	CP-Integrated-CG	CP-Integrated-ScheduleCG
1_day	#duties	109.56	99.6	99.6	99.6	99.6	99.6
	#positioning	13.28	11.62	11.62	11.62	11.62	11.62
	BH/DP	1.133	1.150	1.150	1.148	1.150	1.148
2_day	#duties	222.44	207.5	207.5	207.5	207.5	∗
	#positioning	24.9	23.24	23.24	23.24	23.24	∗
	BH/DP	1.160	1.170	1.172	1.172	1.177	∗
7_day	#duties	722.1	693.88	693.88	†	∗	∗
	#positioning	48.14	46.48	46.48	†	∗	∗
	BH/DP	1.172	1.175	1.179	†	∗	∗

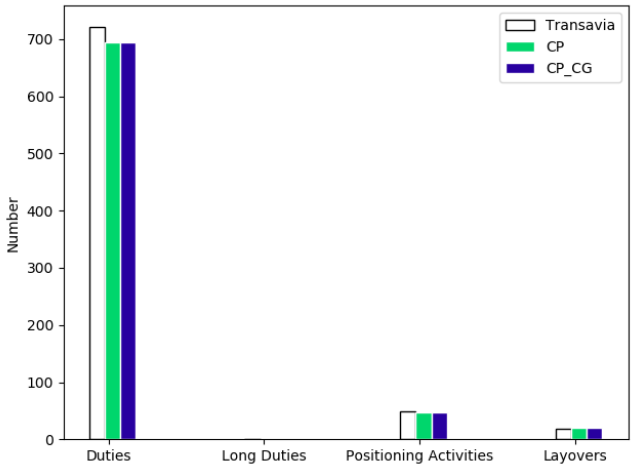
It stands out that significantly fewer (4-10%) pairings are used by the currently presented models. This percentage decreases with problem size. In addition, the resulting crew efficiency measured as the ratio between flight hours and duty hours per duty is 0.6% to 1.5% higher compared to Transavia's pairings. These results decrease with problem size, but generally increase with integration level, as expected.



(a) Number of duties, long duties (>14 hours), duties with positioning, and pairings with layovers on a 1_day dataset



(b) Number of duties, long duties (>14 hours), duties with positioning, and pairings with layovers on a 2_day dataset

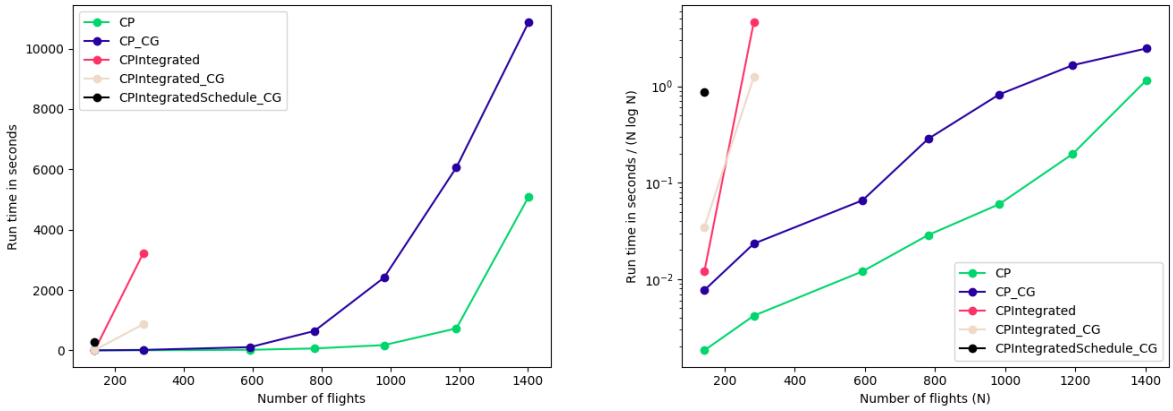


(c) Number of duties, long duties (>14 hours), duties with positioning, and pairings with layovers on a 7_day dataset

Figure 8.5: Barchart comparing the results between Transavia's pairings and the different models' results

8.3.3. Runtime

Lastly, this section expands on the runtime results for all experiments. The runtimes can be observed from Table 8.5, and are visualized in Figure 8.6a. To better show the runtimes that are close together, Figure 8.6b shows the normalized runtime.



(a) Runtime (seconds)

(b) Normalized runtime

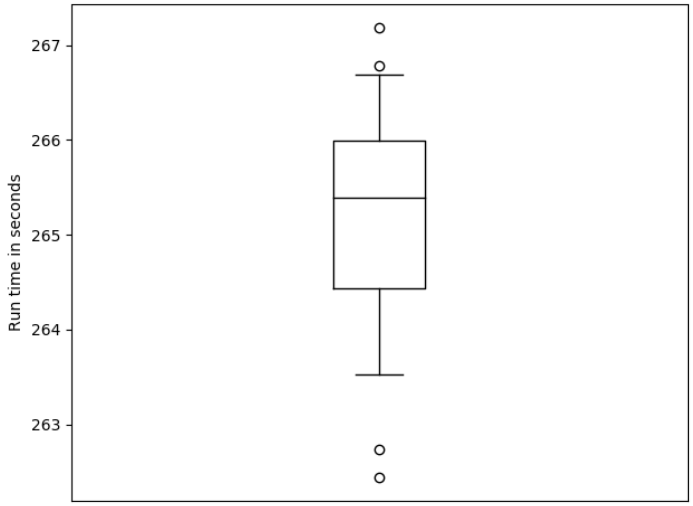
Figure 8.6: Plots showing average (normalized) runtime over problem size for all models (n=24)

As expected, the more integrated a model is, the longer its runtime will be for the same dataset. This can clearly be seen in Figure 8.6a, where there is a large difference between the runtimes of the `CP` and `CPIntegrated` models, as well as between model `CP_CG` and `CPIntegrated_CG`. For the non-integrated models, the runtime of the generate-and-test approach is always smaller than the runtime of the branch-and-price approach. This is most likely due to the relatively small number of decision variables. When looking at the `CPIntegrated` and `CPIntegrated_CG` models, it stands out that the generate-and-test method is faster for the 1_day dataset, while the branch-and-price approach is faster on the larger 2_day dataset. The branch-and-price approach therefore seems to be promising for very large problem sizes; this is conform expectation.

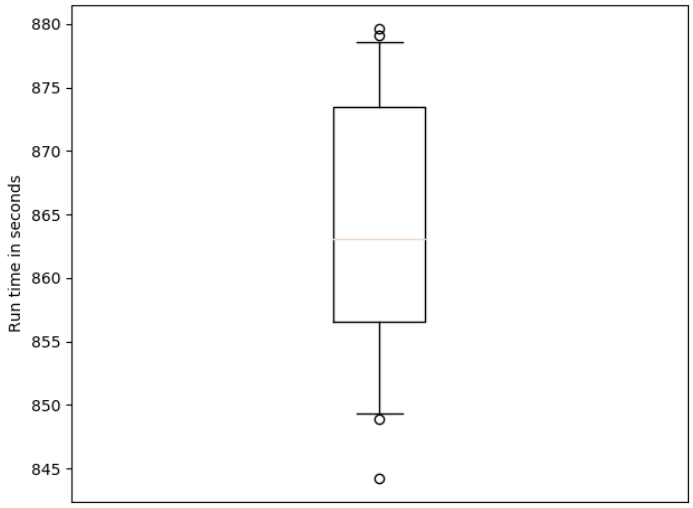
Table 8.5: Average and maximum runtimes per model and data set in seconds. † indicates that the runs did not complete due to too little available memory, while ∗ indicates that the run took longer than 24 hours

	`CP`		`CP_CG`		`CP Integrated`		`CP Integrated _CG`		`CP Integrated Schedule_CG`	
dataset	mean	max	mean	max	mean	max	mean	max	mean	max
1_day	0.56	0.86	2.35	3.64	3.66	5.61	10.54	16.0	265.15	267.19
2_day	2.9	4.23	16.19	24.73	3219.23	3330.85	864.13	879.68	∗	∗
3_day	19.81	24.56	108.05	112.55	†	†	∗	∗	∗	∗
4_day	64.62	65.42	642.67	663.68	†	†	∗	∗	∗	∗
5_day	175.52	177.37	2411.87	3139.14	†	†	∗	∗	∗	∗
6_day	729.23	735.93	6062.17	8528.28	†	†	∗	∗	∗	∗
7_day	5099.19	5603.89	10883.6	14122.46	†	†	∗	∗	∗	∗

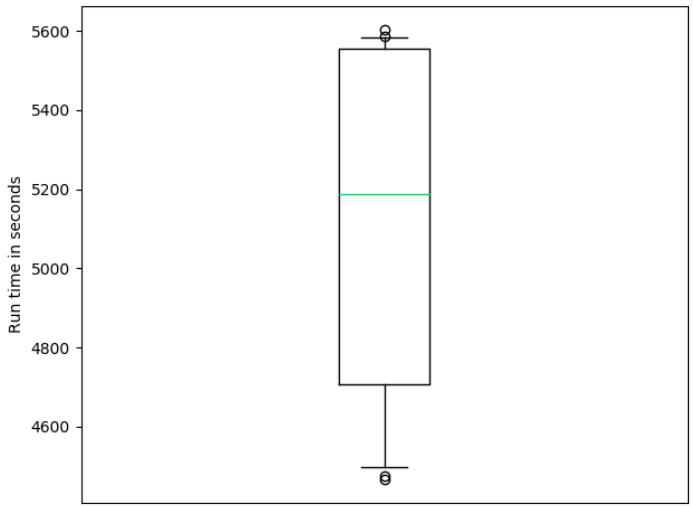
Additionally, it is interesting to look at the mean and maximal runtimes for all models, as presented in Table 8.5 and the boxplots in Figure 8.7. Generally, there is a relatively small difference between the worst-case runtime and the average runtime. This difference is larger for models using the branch-and-price approach. This is as expected, because of the different order in which the columns can be added due to paths having the same weight, and the use of a heuristic for the SPPRC. The spread of the runtimes is also dependent on the problem size, where the larger the problem, the larger the difference between the mean and maximal runtimes becomes. An example hereof is given by the boxplots in Figure 8.7.



(a) `CPIntegratedSchedule_CG` on 1_day data set



(b) `CPIntegrated_CG` on 2_day data set



(c) `CP` on 7_day data set

Figure 8.7: Boxplots of a non-integrated, integrated, and schedule-integrated model showing the runtime results for their largest data sets (n=24). The whiskers indicate the area between the 5th and 95th percentile

8.4. Sensitivity Analysis

This section investigates the influence of the maximum pairing length parameter on the solution quality and runtime results. To this end, 12 additional runs per dataset-model combination have been performed, with the maximum pairing length set to 6 instead of 4 days. All parameters, except the maximum pairing length, were set to the values presented in Section 7.3. Setting the maximum pairing length to a larger value influences which solving method is required for the shortest path problem. Table 8.6 shows the updated dataset characteristics with regards to the SPPRC. It can be seen that the pricing problems of the 2, 3, and 4_day datasets can be now be solved by solving a regular shortest path problem, instead of the SPPRC.

Table 8.6: Datasets and their updated SPPRC attribute when maximum pairing length is set to 6 days

Dataset Number	Dataset Name	SPPRC (length=4 days)	SPPRC (length=6 days)
1	1_day	×	×
2	2_day	✓	×
3	3_day	✓	×
4	4_day	✓	×
5	5_day	✓	✓
6	6_day	✓	✓
7	7_day	✓	✓

As can be seen in Figure 8.8, the number of decision variables increases greatly from the 4_day dataset onwards, as that is the first dataset that has a large number of possible pairings that are longer than 4 days. Interestingly, a similar piecewise linear function as for a maximum pairing length of 4 days can be observed. This time only, a strong increase can be noticed for the CP model from the 4_day dataset onwards, as many more combinations are possible with a maximum pairing length of 6 days. These increased number of decision variables also take up a lot of memory; the CP model with maximum pairing length of 6 days did not finish on the 6_day and 7_day datasets, as the program ran out of memory. From Figure 8.8 it can also be seen that the CP_CG model needs to add a small amount of decision variables before it finds an optimal solution, compared to the 4-day maximum pairing length. However, this increase is far less extreme than can be seen for the CP model.

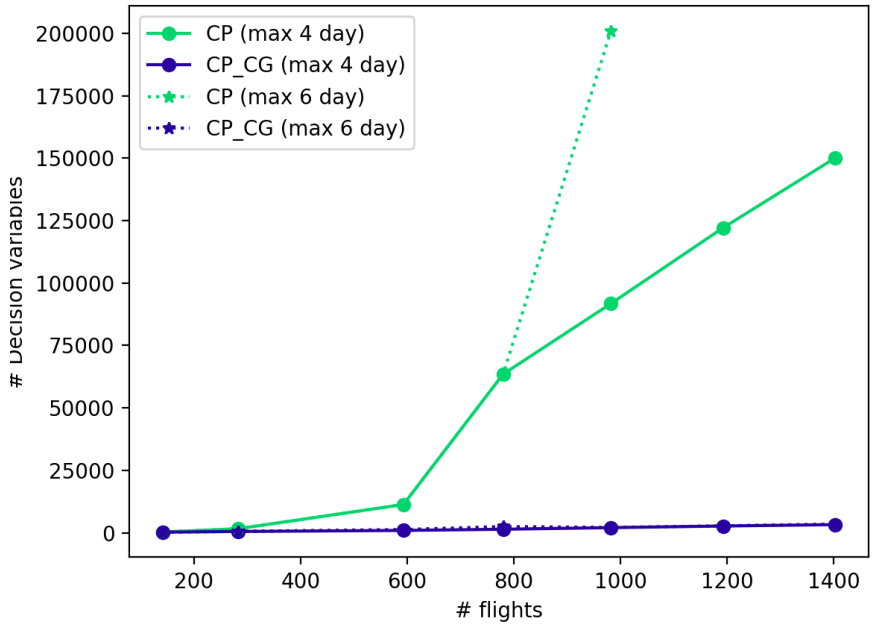


Figure 8.8: Average number of decision variables per dataset for maximum pairing lengths of 4 and 6 days

When looking at the solution quality, a larger maximum pairing length shows to have no effect on the objective value. This can be seen in Figure 8.9, as the dotted and star marked lines for the 6 day maximum pairing length runs are covered by the lines of the 4 day maximum pairing length runs. This can be explained by the high idle time that comes with hotel layovers. Therefore, longer pairings are not favorable over shorter pairings, unless they cover a flight that could not be covered before.

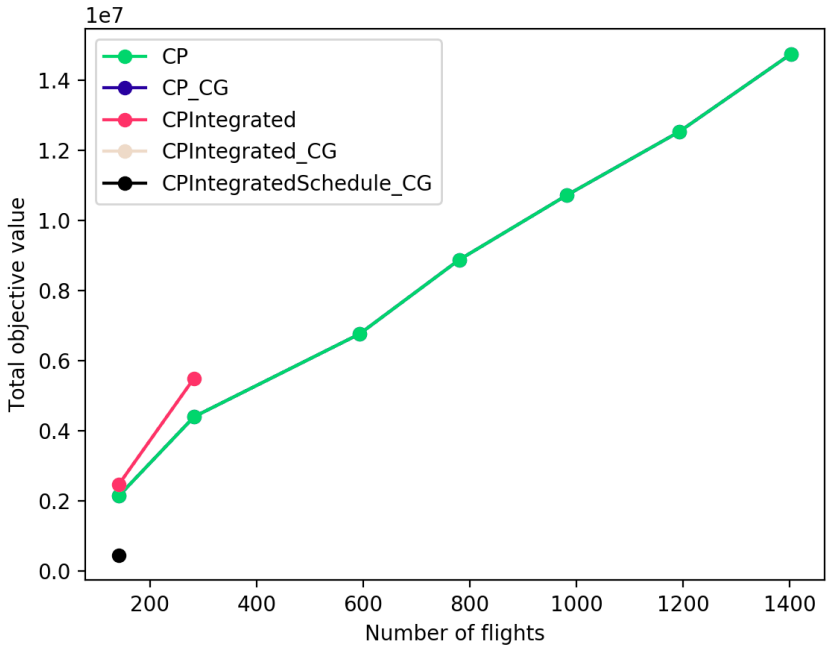
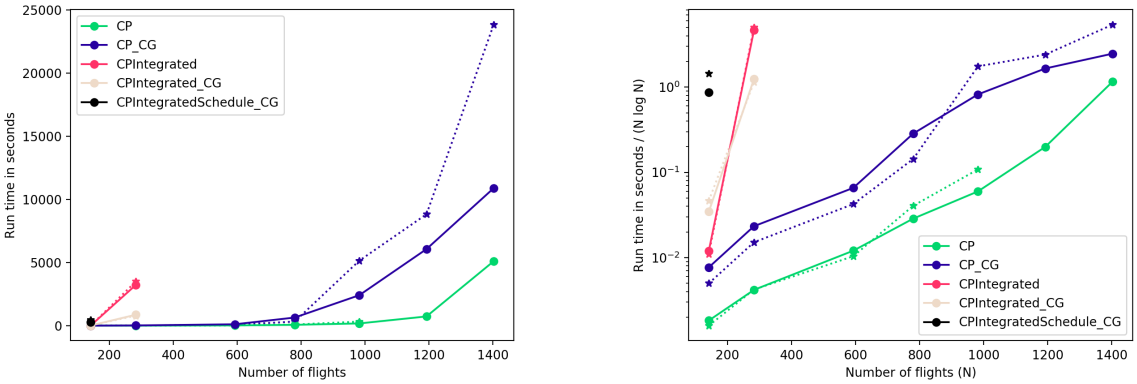


Figure 8.9: Objective value per model and dataset with maximum pairing lengths of 4 and 6 days

With regards to the runtime of the models, Figure 8.9 shows that the runtime for the `CP` and `CP_CG` models is lower when the maximum length is set to 6 days up until the 4_day dataset. For larger datasets, the runtime for the maximum length of 6 days increases above the runtime for the maximum length of 4 days. This can again be attributed to the difference between solving a shortest path problem by using Bellman-Ford or solving a shortest path problem with resource constraints. This difference is smaller for the integrated models.



(a) Runtime (seconds)

(b) Normalized runtime

Figure 8.10: Plots showing average (normalized) runtime over problem size for all models and maximum pairing lengths of 4 and 6 days (n=12). Dotted lines and star markers indicate the use of a 6-day maximum pairing length

9

Conclusion and Discussion

9.1. Conclusion

Crew costs are an airline's second highest expense, being only less expensive than fuel. This research was performed at Transavia, a Dutch low-cost airline, that like many European airlines pays flight crew irregardless of the number of hours flown. Because of this pay structure, it is desirable to assign crew to the flight schedule as efficiently as possible. The two problems associated with making crew schedules are the crew pairing and the crew rostering problem. Currently, airlines solve many different scheduling problems, among which the aircraft routing and crew pairing problem, independently and subsequently. Because the result of the first problem serves as the input for the second, integrally solving them could potentially provide better solutions. The (integrated) crew pairing problem has been studied extensively in the past, but has mostly focused on American network carriers and minimizing pay-and-credit, while European low-cost carriers and optimizing crew efficiency are often overlooked. To fill these gaps, this thesis investigated how the crew pairing problem can be integrated with the aircraft routing problem and schedule retiming as to improve the crew productivity of a low-cost carrier operating a point-to-point network. To this end, the objective function was formulated to reflect crew and aircraft idle time, defined as the duty time that is not used to operate scheduled flights. Further penalties were used to discourage taxi movements and overnight stays. To enable prioritization in the objective function, a parameter was defined that influences the cost of aircraft routes relative to pairing costs. To investigate the impact of integrally optimizing crew and aircraft routes on the quality of the resulting pairings, four different models with different integration levels and solution approaches were defined. Model `CG` solves the crew pairing problem by generating all pairings in advance and solving an integer linear problem, while model `CP_CG` solves the crew pairing model with a branch-and-price approach. The integrated models `CPIntegrated` and `CPIntegrated_CG` solve the integrated crew pairing and aircraft routing problem by using a generate-and-test and branch-and-price approach respectively. Lastly, it was investigated how realistic flight schedule changes can be combined with the integrated crew pairing and aircraft routing problem. To this end, every flight in the schedule was copied twice, and their corresponding departure and arrival times were changed to five minutes earlier for the first copy, and to five minutes later for the second copy. The results show that the pairings created by the currently presented models include fewer positioning and layover activities, and have a higher block hour to duty hour ratio than the pairings provided by Transavia. Additionally, while the more integrated models have longer runtimes, they are also able to decrease the objective value the most. In short, the practical results of up to 10% fewer duties and up to 1.5% higher crew efficiency, show that integrated crew scheduling is also promising for airlines operating a point-to-point network.

9.2. Discussion

Despite these promising outcomes, the practical feasibility of the resulting pairings and aircraft routes could benefit from a few further extensions. Firstly, each aircraft type has different possible configurations, of which, for example, the maximum take-off weight is of great importance. Although two aircraft are of the same type, it might be that one has a lower maximum take-off weight and is therefore unable to fly to all of the destinations in the schedule. Additionally, it is important to mention that the current research did not take required maintenance activities into account. As aircraft need maintenance in a hangar as often as once a week, including this constraint in the integrated problem will increase the applicability of the results

in real life. In addition, the currently presented models can be extended with additional possibilities in the duty generation phase. For example, currently, the models do not produce duties where a positioning activity is located in the middle of a duty, surrounded by flight activities. This is however a possibility that can be explored in the future. With regards to the presented algorithms, it can be seen that the branch-and-bound phase takes a large number of iterations. This can be a motive to look further into promising branching strategies that need fewer iterations to find an integral solution. Also regarding strategy, it could be beneficial for the retiming model to not retime all flights, but a select number chosen by a heuristic, that are expected to yield improvements in the pairings that can be made by allowing retiming. To further decrease the general runtime of all models, it is advisable to implement these and similar models in a different environment and language, e.g. C++, that is more suitable to multiprocessing. This allows the subproblems of the integrated models to be solved in parallel, making the entire solving process a lot faster. These kind of languages are also more performance focused and could greatly speed up the existing models, making them better to work with.

A

Rules

This appendix gives an overview of all the rules that have been included in the by this thesis presented models. These rules have been translated and interpreted from the EASA flight time limitation document, and Transavia's cockpit crew collective labor agreement 2013-2016 [1].

A.1. Rules Related to Duties

1. **Check-in**
   (Article C.17b [1])
   - If one of the activities in a duty has a check-in time before 04:30lt, no more than 4 sequential flights can be assigned.
2. **Maximum Number of Landings**
   (Article D.3c [1])
   - A duty that contains more than 13 flight-duty hours can contain at most 3 landings.
3. **Maximum Duty-Hours**
   (Article D.4 [1])
   - A length of one duty may at most be 16 hours.
4. **Minimum Rest**
   (Bijlage 7 Aanhangsel III [1])
   - The minimum amount of rest after a duty depends on the previous duty period as follows:

Table A.1: Minimum hours of rest required depending on length of previous duty period

Length Previous Duty Period	Minimum Rest
Shorter than 11 hours and 29 minutes	11 hours
11 hours and 30 minutes - 12 hours and 29 minutes	12 hours
12 hours and 30 minutes - 13 hours and 29 minutes	13 hours
13 hours and 30 minutes - 14 hours and 29 minutes	14 hours
14 hours and 30 minutes - 15 hours and 29 minutes	15 hours
Longer than 15 hours and 30 minutes	16 hours

5. **Maximum GVWT**
   (Bijlage 7 Aanhangsel I, Tabel A [1])

   - The maximum number of flight-duty hours allowed depending on check-in time are as follows:

Table A.2: Maximum duty hours depending on check-in time

Check-in time	**Maximum GVWT in Hours**
01:01 – 03:15	10:30
03:16 – 03:30	10:48
03:31 – 03:45	11:06
03:46 – 03:59	11:23
04:00 – 04:15	11:40
04:16 – 04:30	11:50
04:31 – 04:45	12:00
04:46 – 05:00	12:09
05:01 – 05:15	12:18
05:16 – 05:30	12:27
05:31 – 05:45	12:36
05:46 – 06:00	12:45
06:01 – 06:15	12:54
06:16 – 06:30	13:03
06:31 – 06:45	13:12
06:46 – 07:00	13:21
07:01 – 13:00	13:30
13:01 – 13:15	13:18
13:16 – 13:30	13:06
13:31 – 13:45	12:54
13:46 – 14:00	12:42
14:01 – 14:15	12:30
14:16 – 14:30	12:18
14:31 – 14:45	12:06
14:46 – 15:00	11:54
15:01 – 15:15	11:42
15:16 – 15:30	11:30
15:31 – 15:45	11:18
15:46 – 16:00	11:06
16:01 – 16:15	10:54
16:16 – 16:30	10:42
16:31 – 24:00	10:30

6. **Maximum FDP**

   - The maximum number of flight-duty hours allowed depending on check-in time are as follows:

Table A.3: Maximum flight-duty hours depending on check-in time and number of flights in the duty

Check-in time (lt)	**Maximum FDP in hours**								
	1-2 flights	3 flights	4 flights	5 flights	6 flights	7 flights	8 flights	9 flights	10 flights
06:00-13:29	13:00	12:30	12:00	11:30	11:00	10:30	10:00	09:30	09:00
13:30-13:59	12:45	12:15	11:45	11:15	10:45	10:15	09:45	09:15	09:00
14:00-14:29	12:30	12:00	11:30	11:00	10:30	10:00	09:30	09:00	09:00
14:30-14:59	12:15	11:45	11:15	10:45	10:15	09:45	09:15	09:00	09:00
15:00-15:29	12:00	11:30	11:00	10:30	10:00	09:30	09:00	09:00	09:00
15:30-15:59	11:45	11:15	10:45	10:15	09:45	09:15	09:00	09:00	09:00
16:00-16:29	11:30	11:00	10:30	10:00	09:30	09:00	09:00	09:00	09:00
16:30-16:59	11:15	10:45	10:15	09:45	09:15	09:00	09:00	09:00	09:00
17:00-04:59	11:00	10:30	10:00	09:30	09:00	09:00	09:00	09:00	09:00
05:00-05:14	12:00	11:30	11:00	10:30	10:00	09:30	09:00	09:00	09:00
05:15-05:29	12:15	11:45	11:15	10:45	10:15	09:45	09:15	09:00	09:00
05:30-05:44	12:30	12:00	11:30	11:00	10:30	10:00	09:30	09:00	09:00
05:45-05:59	12:45	12:15	11:45	11:15	10:45	10:15	09:45	09:15	09:00

A.2. Rules Related to Pairings

1. **Length of a workweek**
   (Article C.7b [1])

   - A workweek may contain at most 6 duties

2. **Check-in**
   (Article C.16:1-6 [1])

   - After an activity with a check-in time between 00:00lt and 02:29lt, the next check-in time will be at least 27 hours later.
   - After an activity with a check-in time between 02:30lt and 03:59lt, the next check-in time will be at least 26 hours later.
   - After an activity with a check-in time between 04:00lt and 04:59lt, the next check-in time will be at least 24 hours and 15 minutes later.
   - After an activity with a check-in time between 05:00lt and 05:59lt, the next check-in time will be at least 24 hours later.
   - After an activity with a check-in time between 06:00lt and 07:59lt, the next check-in time will be at least 23 hours later.
   - After an activity with a check-in time from 08:00lt, the next check-in time will not be before 05:30lt.

3. **Check-out time and following check-in time**
   (C.17 [1])

   - If the time of check-out is before 02:00lt, 13 hours of rest are required, and the next check-in time may not be between 23:00lt and 06:59lt.
   - If the time of check-out is between 02:01lt and 03:00lt, 13 hours and 30 minutes of rest are required, and the next check-in time may not be between 23:00lt and 07:59lt.
   - If the time of check-out is between 03:01lt and 04:00lt, 14 hours of rest are required, and the next check-in time may not be between 23:00lt and 08:59lt.
   - If the time of check-out is between 04:01lt and 05:00lt, 15 hours of rest are required, and the next check-in time may not be between 23:00lt and 09:59lt.
   - If the time of check-out is after 05:01lt, 29 hours of rest are required.

A.3. Rules Related to the Interaction Between Aircraft and Crew

1. **Aircraft Change**

   - When a crew needs to change aircraft within a duty, 15 minutes of extra turnaround time is needed.

B

Pairing Graph Construction

The pairing graph is an important element in all approaches presented in this thesis. To ensure reproducibility and increase understanding of the models' underlying mechanics, this appendix gives the details to the graph construction explained in Section 4.1.

The pairing graph consists of nodes representing bases (source and sinks), taxi activities, taxi+overnight activities and duties. The edges that are possible between all these nodes are given in Figure B.1. Each edge has an associated weight, representing the amount of idle time: the available duty time not used to operate flights. Each edge also contains a resource cost attribute that records the time needed to execute the pairing. The models that use branch-and-price to find an optimal set of pairings need this attribute to ensure that the created pairings do not violate the maximum pairing length. To construct the pairing graph, all nodes are created first, after which edges and their corresponding weight and resource costs are created. This is done by following the steps presented in Section 4.1.

Because there is only one taxi node per base source node, the edges from the source node to the taxi node have both weight and resource cost 0. The cost of using a taxi is put into the edge attributes of the edges from the taxi node to the first duty. For the weight, these edges get their usual weight as edges going into a duty node, but are increased by the taxi time from the source to the duty's departure airport. This also holds for the resource cost attributes. Similarly, the edges going to a taxi node just include the taxi time as its weight and resource cost. A full overview can be seen in Figure B.1.

To improve the runtime of all models, the pairing graph is kept as small as possible. This is done by iterating over the pairing graph and removing all nodes that have an out-degree or in-degree of 0. By iteratively removing these nodes it is ensured that edges and nodes that cannot be part of a path between a source and a sink node do not remain in the graph and therefore do not slow down the process.

Figure B.1: All possible edges in the pairing graph, with their corresponding weights and resource costs

C

Routing Graph Construction

This appendix explains how the routing graphs are constructed for all models presented in this thesis. Firstly, it is important to remember that each given aircraft type has its own routing graph, to simplify finding a solution. To construct a routing graph for one of the aircraft types, a source and sink node is created for all bases. Additionally, for all flights that are assigned to that aircraft type, a node is created. The process of assigning edges between these nodes is very simple, and only happens when there is enough turn around time between the two flights. Additionally, all flights that depart from a base get an incoming edge from the source node of that base, while all flights that arrive at a base get an outgoing edge to the base's sink node. The edges in the routing graphs only have one attribute: weight. Similar to the pairing graph, the weight is represented as the idle time of the aircraft (in seconds), as a consequence this also minimizes the number of aircraft that are used to cover all flights. This is accomplished by giving outgoing edges from the source node a weight that corresponds to the difference between the departure time of the node's flight and the first flight in the schedule. Similarly, all incoming edges to the sink nodes get a weight that represents the time between the arrival time of the flight and the arrival time of the last flight in the schedule. As explained in Section 5.1, the priority between optimizing pairings and aircraft routes can be shifted by adjusting the w parameter. All edge weights in the routing graphs are multiplied by this parameter. All possible edges and their respective weights can be observed from Figure C.1.

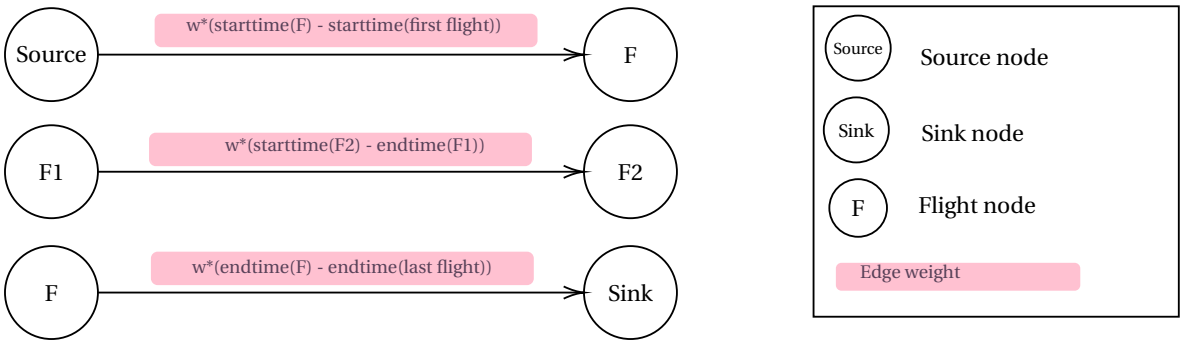


Figure C.1: All possible edges in the routing graphs and their respective weights

An Aircraft and Schedule Integrated Approach to Crew Scheduling for a Point-to-Point Airline

Draft

Johanna P. Korte *, Neil Yorke-Smith

Delft University of Technology, Faculty of Electrical Engineering, Mathematics and Computer Science, Van Mourik Broekmanweg 6, 2628 XE Delft, The Netherlands

ARTICLE INFO

Article History
Draft December 5, 2019

Keywords:
Combinatorial optimization
Integer programming
Large scale optimization
OR in airlines

ABSTRACT

For airlines, crew costs make up the second largest expense, behind fuel costs. Because these costs are very high, there is a large potential gain in improving the crew efficiency within the bounds set by the law and collective labor agreements. The current research presents an integrated model that allows flight retiming, and focuses on obtaining efficient crew pairings for airlines operating point-to-point networks. Solutions are obtained by using a branch-and-price approach, with a shortest path pricing problem. The model is evaluated on data from a Dutch low-cost carrier that operates a point-to-point network.

Introduction

Crew costs are an airline's second highest expense, being only less expensive than fuel. This research was performed on data of a Dutch low-cost airline, that like many European airlines pays cockpit crew irregardless of the number of hours they fly. Because of this pay structure, it is desirable to assign crew to the flight schedule as efficiently as possible. Currently, many airlines solve many different scheduling problems, among which the aircraft routing and crew pairing problem, independently and subsequently. Because the result of the first problem serves as the input for the second, integrally solving them could potentially provide better solutions. The problem concerned with scheduling crew is called the *crew scheduling problem*. As the problem is NP-hard, the problem is often solved in two phases: the *crew pairing problem* and the *crew rostering problem*. In the crew pairing problem, a set of optimal sequences of workdays, called *pairings*, to be executed by the crew are determined such that all flights are covered, while the individual crew schedules are determined in the crew rostering problem by assigning these pairings to crew members. Although individual crew members only get assigned in the crew rostering phase, the cost-determining phase of the crew scheduling problem is the pairing problem [34]. With regards to the problem horizon, a few different variants of the crew pairing problem exist. Due to the hardness of the problem and its rapidly increasing size, many studies solve a daily problem, where the flight schedule that serves as input contains all flights for an airline on 1 day [6][10][36]. Another approach that is used regularly is that of the weekly problem [33]. The daily and weekly problems have as an advantage that a single problem can be solved first, after which the solution is copied onto the next day or week, after which any remaining inconsistencies are solved. From the first studies onwards, a vast majority of research has formulated the crew pairing problem as either a set partitioning or a set cover problem [25][31][41][52]. These formulations enforce all flights to be covered by a pairing, and can easily be extended to capture the scope of the problem by adding constraints. To solve the pairing problem as a set cover or set partition problem, a set of candidate pairings is required. This set can be generated by enumeration or with the use of heuristics. Instead of generating the pairings in advance, it is also possible to generate pairings during the solving process by using column generation. Traditionally, the pairing problem has been solved optimally with the help of column generation and branch-and-bound [28][29][51][52]. Because the crew pairing problem is an example of a binary integer programming problem, column generation is often used to solve the relaxed problem, while branch-and-bound is used afterwards to obtain the optimal integer solution from the relaxed solution. An example of combining column generation and branch-and-bound is given by Yan et al. who solved the pairing problem for cabin crew, while also taking into account different cabin classes, mixed-aircraft types, and home bases [51]. Yan et al. formulated the crew pairing problem using flight networks, similar to Deng and Lin and Ozdemir and Mohan [18][38]. Aiming to get good pairings for realistic problem sizes in less time, studies have also started to apply metaheuristic methods, such as genetic algorithms [38], general variable neighborhood search [4], and ant colony optimization [18]. One of the latest research efforts on metaheuristics by Deveci and Demirel used memetic algorithms, a combination of local search and genetic algorithms, to solve the crew pairing problem [21]. Deveci and Demirel compared their algorithm to metaheuristic algorithms by Beasly and Chu [8] and Zeren and Özkol [53], and found that the memetic algorithm achieved better solutions while runtimes remained similar. Contrary to Agustin et al. [8] the study by Deveci and Demirel uses instances with sizes of up to 700 flights, and therefore resemble

*Corresponding author. Tel.:+31612475472;
E-mail: J.P.Korte@student.tudelft.nl

real-life problem sizes better. The sequential solving of different airline planning problems might deliver optimal results for each individual problem, but is not likely to produce a solution that is optimal for the entire airline planning problem [46]. This is due to the interdependence of the problems; the output of one problem acts as input for the next. As a result, more and more publications focus on integrated scheduling, where different planning problems are solved simultaneously. As discussed before, the crew scheduling problem is often solved in two phases, due to its complexity. A large advantage of solving the entire crew scheduling problem at once is that it eliminates the need for a separate pairing objective function, which can be hard to design to resemble reality. In 2011, Saddoune et al. solved the entire crew scheduling problem using a combination of column generation and dynamic constraint aggregation [42]. A great cost saving of approximately 4% are reported on real-life instances. Unfortunately, the study does not take into account pre-assigned activities, such as safety training, simulator assignments, or holidays. Another interesting approach to integrated scheduling includes combining the crew pairing problem with the aircraft routing problem. By determining the crew pairings and aircraft routes at the same time, it is possible to generate new pairings that were illegal before due to the extra time and costs associated with an aircraft change mid-pairing [13][46]. Papadakos provides an example of how the crew pairing problem can be fully integrated with the maintenance and aircraft routing problems [39]. Because of the large number of constraints, Papadakos applies Benders decomposition and combines this with a column generation strategy that is accelerated by using heuristics. It is shown that a fully integrated approach delivers better results than the semi-integrated approaches, and could result in millions of dollars of cost savings each year for large airlines. A downside of the fully integrated approach is that it takes significantly longer to obtain solutions. Very new and little researched is the incorporation of schedule changes within crew pairing models. When the schedule is fixed, many useful pairings might not be generated because they violate a constraint by a small margin. By allowing changes to the initial flight schedule, it is possible to facilitate the design of more convenient pairings. An example of a small schedule modification is retiming: adjusting the departure and arrival time of a flight by a few minutes [14]. Cacchiani and Salazar-González propose an entirely integrated model for the crew pairing, aircraft routing and fleet assignment problems including retiming [14]. The mixed-integer linear programming formulation includes retimed copies for all flight legs, and constraints to make sure only one of the original or copies is chosen for each flight leg. Given the integrated formulation, the relaxed problem is first solved, after which one of four proposed heuristics is used to reach a final solution. It was shown that retiming flights can improve solution quality with at most 7% on large instances. They also conclude that offering retiming by 10 minutes resulted in better solutions than retiming by 5 minutes, while runtimes stayed similar. Although the (integrated) crew pairing problem has been studied extensively in the past, the focus has mostly been on American network carriers and minimizing pay-and-credit. While European low-cost carriers and optimizing crew efficiency are often overlooked. To fill these gaps, this study investigates how the crew pairing problem can be integrated with the aircraft routing problem and schedule retiming as to improve the crew productivity of a low-cost carrier operating a point-to-point network. The following section will describe the model and corresponding solving approach, while the Results section discusses the outcomes of this model and compares it to other integrated and non-integrated models that solve the crew pairing problem. Lastly, the conclusion discusses all obtained results and provides directions for future research.

Methodology

This section provides a model for the integrated crew pairing and aircraft routing problem. The aim of the model is to maximize the crew efficiency. To this end, the objective function has been formulated to minimize the crew's available duty time that is not spent on flying. The problem is solved by first creating two flight copies for each flight, one that has a departure and arrival time 5 minutes earlier, and one 5 minutes later, than the original flight. Then, a set of all legal duties is created. With these duties, a pairing graph consisting of duties and base nodes is constructed, such that a path from a source base node to a sink base node represents a legal pairing. An example of such a graph can be seen in the figure below.

Figure showing example of a pairing graph

For each aircraft type, a similar graph is constructed to represent aircraft routes. These graphs however consist of base nodes and flight nodes. The edge weights of these graphs are formulated to represent the time that the aircraft is not flying. An example of such a graph can be seen below.

Figure showing example of a routing graph

To ensure that when crew change aircraft, enough extra time is available, a list of crew-aircraft short connections is created. This list can subsequently be used to enforce that if one of the flight combinations in the list is operated in the same pairing, it should also be operated in the same aircraft route. The list of short connections contains all combinations of two flights f_1 and f_2 for which the following holds:

- f_1 and f_2 are assigned to the same aircraft type
- The arrival airport of f_1 and the departure airport of f_2 are the same
- The time between the arrival of f_1 and the departure of f_2 is larger than the minimum required turn around time, but smaller than the minimum required turn around time + 15 minutes.

Subsequently the following integer linear program, based on the set partition is solved using branch and price. For the restricted master problem, the integrality constraints presented by D.7 and D.8 are relaxed.

$$\min \sum_{p\in P} c_p x_p + \sum_{r\in R} c_r x_r \tag{C.1}$$

$$\text{s.t.} \sum_{t\in D_f} \sum_{p\in P} b^p_{f_t} x_p = 1 \qquad \forall f \in F \tag{C.2}$$

$$\sum_{t\in D_f} \sum_{r\in R} b^r_{f_t} x_r = 1 \qquad \forall f \in F \tag{C.3}$$

$$\sum_{p\in P} b^p_{i_v j_w} x_p - \sum_{r\in R} b^r_{i_v j_w} x_r \le 0 \qquad \forall (i,j) \in C, \quad (v,w) \in C_{i,j} \tag{C.4}$$

$$\sum_{r\in R} b^a_r b^{k^-}_r x_r \le maxAC^k_a \qquad \forall k \in K, a \in A \tag{C.5}$$

$$\sum_{p\in P} b^p_{f_t} x_p - \sum_{r\in R} b^r_{f_t} x_r = 0 \qquad \forall f \in F, t \in D_f \tag{C.6}$$

$$x_p \in \{0,1\} \qquad \forall p \in P \tag{C.7}$$

$$x_r \in \{0,1\} \qquad \forall r \in R \tag{C.8}$$

With:
$\boldsymbol{x_p}$: Binary decision variable that indicates whether pairing $\boldsymbol{p}$ is chosen
$\boldsymbol{x_r}$: Binary decision variable that indicates whether route $\boldsymbol{r}$ is chosen
$\boldsymbol{P}$: Set of pairings
$\boldsymbol{R}$: Set of routes
$\boldsymbol{K}$: Set of bases

A: Set of aircraft types
F: Set of flights in the schedule
D_f: Set of departure times for flight f
C: Set of short connections
$C_{i,j}$: Set of departure time combinations (v, w) where v belongs to flight $i \in C$, and w belongs to $j \in C$
c_p: Cost of choosing pairing p
c_r:Cost of choosing route r
$MaxAC_a^k$: Number of available aircraft of type a at base k
$b_{f_t}^p$:Binary constant indicating whether flight f with departure time t is in pairing p
$b_{f_t}^r$: Binary constant indicating whether flight f with departure time t is in route r
b_r^a: Binary constant indicating whether route r is operated by aircraft type a
$b_r^{k^-}$: Binary constant indicating whether route r starts in base a
$b_{i,j}^p$: Binary constant indicating whether flight i and j are in pairing p
$b_{i,j}^r$: Binary constant indicating whether flight i and j are in route r
$b_{i_v j_w}^p$: Binary constant indicating whether flight i with departure time v and j with departure time w are in pairing p
$b_{i_v j_w}^r$: Binary constant indicating whether flight i with departure time v and j with departure time w are in route r

At each iteration of the column generation algorithm, a pricing problem is solved. To solve the pricing problem for this model, it is first needed to update the pairing and routing graphs with the dual variables obtained from solving the restricted master problem. This is done according to the following steps:

Pairing Graph Updates

- The values of dual variables α_f, corresponding to primal Constraints D.2, are subtracted from the weights of the edges incoming to duty nodes that contain a flight copy of flight f
- The values of dual variables γ_{i_v, j_w}, corresponding to primal Constraints D.4, are subtracted from the weights of the edges incoming to duty nodes that contain both flight i with departure time v and flight j with departure time w
- The values of dual variables $\epsilon_{f,t}$, corresponding to primal Constraints D.6, are subtracted from the weights of the edges incoming to duty nodes that contain flight f with departure time t

Routing Graph Updates

- The values of dual variables β_f, corresponding to primal constraints D.3, are subtracted from the weights of the edges incoming to the flight node representing a copy of flight f
- The values of dual variables γ_{i_v, j_w}, corresponding to primal Constraints D.4, are added to the weights of the edges between flight nodes representing flight i with departure time v and flight j with departure time w
- The values of dual variables $\delta_{a,k}$, corresponding to primal constraints 6.5, are subtracted from the weight of outgoing edges from the base source node of base k in the routing graph of aircraft type a
- The value of dual variables $\epsilon_{f,t}$, corresponding to primal Constraints D.6, are added to the weights of the edges incoming to flight nodes that represent flight f with departure time t

After the graphs have been updated, a (resource) shortest path problem is solved for each base combination on each graph. If the cost of a resulting path in the updated graph is smaller than 0, it will be added as a pairing or route column in the restricted master problem. Once a solution has been found to the linear relaxation, an integral solution is found by using branch and bound, while keeping to add any negative reduced columns when they are available.

Results

Here I would provide results of runs on multiple one day datasets. Vary in season and number of flights per day. In addition, metrics on runtime, objective value and practical implications. It is interesting to compare these results to an integrated branch and price model without retiming.

Conclusion

Here I would provide a conclusion and offer directions for further research.

Bibliography

[1] *Collectieve Arbeidsovereenkomst Transavia Vliegers 2013-2016*. URL http://cao.minszw.nl/pdf/174/2017/174_2017_13_238976.pdf.

[2] Divyam Aggarwal, Dhish Kumar Saxena, Michael Emmerich, and Saaju Paulose. On Large-Scale Airline Crew Pairing Generation. *IEEE Symposium Symposium Series on Computational Intelligence SSCI 2018*, (November 2018):593–600, 2018.

[3] Alba Agustin, Aljoscha Gruler, Jesica de Armas, and Angel A. Juan. Optimizing Airline Crew Scheduling Using Biased Randomization: A Case Study. In *Conference of the Spanish Associaton for Artificial Intelligence*, pages 331–340, 2016.

[4] Alba Agustin, Angel Juan, and Eduardo G. Pardo. A variable neighborhood search approach for the crew pairing problem. *Electronic Notes in Discrete Mathematics*, 58:87–94, 2017. ISSN 15710653. doi: 10.1016/j.endm.2017.03.012. URL http://dx.doi.org/10.1016/j.endm.2017.03.012.

[5] Ayyuce Aydemir-Karadag, Berna Dengiz, and Ahmet Bolat. Crew pairing optimization based on hybrid approaches. *Computers and Industrial Engineering*, 65(1):87–96, 2013. ISSN 03608352. doi: 10.1016/j.cie.2011.12.005. URL http://dx.doi.org/10.1016/j.cie.2011.12.005.

[6] A. Azadeh, M. Hosseinabadi Farahani, H. Eivazy, S. Nazari-Shirkouhi, and G. Asadipour. A hybrid meta-heuristic algorithm for optimization of crew scheduling. *Applied Soft Computing Journal*, 13(1):158–164, 2013. ISSN 15684946. doi: 10.1016/j.asoc.2012.08.012. URL http://dx.doi.org/10.1016/j.asoc.2012.08.012.

[7] Cynthia Barnhart, Amy M Cohn, Ellis L Johnson, Diego Klabjan, George L Nemhauser, and Pamela H Vance. Airline Crew Scheduling. In *Handbook of Transportation Science*, chapter 14, pages 517–560. 2003. ISBN 978-0-306-48058-4.

[8] J. E. Beasley and P. C. Chu. A genetic algorithm for the set covering problem. *European Journal of Operational Research*, 94: 392–404, 1996. ISSN 03772217. doi: 10.1016/0377-2217(95)00159-X.

[9] Richard Bellman. On a routing problem. *Quarterly of Applied Mathematics*, pages 87–90, 1958.

[10] Mohamed Ben Ahmed, Farah Zeghal Mansour, and Mohamed Haouari. Robust integrated maintenance aircraft routing and crew pairing. *Journal of Air Transport Management*, 73:15–31, 2018. ISSN 09696997. doi: 10.1016/j.jairtraman.2018.07.007. URL https://doi.org/10.1016/j.jairtraman.2018.07.007.

[11] Natashia Boland, John Dethridge, and Irina Dumitrescu. Accelerated label setting algorithms for the elementary resource constrained shortest path problem. *Operations Research Letters*, 34(1):58 – 68, 2006. ISSN 0167-6377. doi: https://doi.org/10.1016/j.orl.2004.11.011. URL http://www.sciencedirect.com/science/article/pii/S0167637705000040.

[12] E Rod Butchers, Paul R Day, Andrew P Goldie, Stephen Miller, Jeff A Meyer, David M Ryan, Amanda C Scott, Chris A Wallace, E Rod Butchers, Paul R Day, Andrew P Goldie, Stephen Miller, Jeff A Meyer, David M Ryan, Amanda C Scott, A Chris, Paul R Day, Andrew P Goldie, Jeff A Meyer, David M Ryan, Amanda C Scott, and Chris A Wallace. INFORMS Journal on Applied Analytics. *INFORMS Journal on Applied Analytics*, 31(1):30–56, 2001.

[13] Valentina Cacchiani and Juan José Salazar-González. A heuristic approach for an integrated fleet-assignment, aircraft-routing and crew-pairing problem. *Electronic Notes in Discrete Mathematics*, 41:391–398, 2013. ISSN 15710653. doi: 10.1016/j.endm.2013.05.117.

[14] Valentina Cacchiani and Juan José Salazar-González. Heuristic approaches for flight retiming in an integrated airline scheduling problem of a regional carrier. *Omega (United Kingdom)*, Advance On, 2019. ISSN 03050483. doi: 10.1016/j.omega.2019.01.006. URL https://doi.org/10.1016/j.omega.2019.01.006.

[15] Hai D. Chu, Eric Gelman, and Ellis L. Johnson. Solving Large Scale Crew Pairing Problems. *European Journal of Operational Research*, 97:260–268, 1997.

[16] Gerald Cook and Jeremy Goodwin. Airline Networks: A Comparison of Hub-and-Spoke and Point-to-Point Systems. *Journal of Aviation/Aerospace Education & Research*, 17(2), 2008. doi: 10.15394/jaaer.2008.1443.

[17] Jean-François Cordeau, Goran Stojković, François Soumis, and Jacques Desrosiers. Benders Decomposition for Simultaneous Aircraft Routing and Crew Scheduling. *Transportation Science*, 35(4):375–388, 2001. ISSN 0041-1655. doi: 10.1287/trsc.35.4.375.10432.

[18] Guang-Feng Deng and Woo-Tsong Lin. Ant colony optimization-based algorithm for airline crew scheduling problem. *Expert Systems with Applications*, 38:5787–5793, 2011. ISSN 09574174. doi: 10.1016/j.eswa.2010.10.053. URL http://dx.doi.org/10.1016/j.eswa.2010.10.053.

[19] G. Desaulniers, J. Desrosiers, Y. Dumas, S. Marc, B. Rioux, M. M. Solomon, and F. Soumis. Crew pairing at Air France. *European Journal of Operational Research*, 97(1):245–259, 1997. ISSN 03772217. doi: 10.1016/S0377-2217(96)00195-6.

[20] G. Desaulniers, J. Desrosiers, and M. M. Solomon. *Column Generation*. Springer US, 1st edition, 2005. doi: 10.1007/b135457.

[21] Muhammet Deveci and Nihan Çetin Demirel. A survey of the literature on airline crew scheduling. *Engineering Applications of Artificial Intelligence*, 74:54–69, 2018. ISSN 09521976. doi: 10.1016/j.engappai.2018.05.008. URL https://doi.org/10.1016/j.engappai.2018.05.008.

[22] Muhammet Deveci and Nihan Çetin Demirel. Evolutionary algorithms for solving the airline crew pairing problem. *Computers and Industrial Engineering*, 115:389–406, 2018. ISSN 03608352. doi: 10.1016/j.cie.2017.11.022. URL https://doi.org/10.1016/j.cie.2017.11.022.

[23] Jenny Díaz-Ramírez, José Ignacio Huertas, and Federico Trigos. Aircraft maintenance, routing, and crew scheduling planning for airlines with a single fleet and a single maintenance and crew base. *Computers and Industrial Engineering*, 75(1):68–78, 2014. ISSN 03608352. doi: 10.1016/j.cie.2014.05.027.

[24] Michelle Dunbar, Gary Froyland, and Cheng-Lung Wu. An integrated scenario-based approach for robust aircraft routing, crew pairing and re-timing. *Computers and Operations Research*, 45:68–86, 2014. ISSN 03050548. doi: 10.1016/j.cor.2013.12.003. URL http://dx.doi.org/10.1016/j.cor.2013.12.003.

[25] Günea Erdogan, Mohamed Haouari, Melda Örmeci Matoglu, and Okan Örsan Özener. Solving a large-scale crew pairing problem. *Journal of the Operational Research Society*, 66:1742–1754, 2015. ISSN 14769360. doi: 10.1057/jors.2015.2.

[26] Lester R. Ford. Network flow theory. *RAND Corporation*, 1956. URL https://www.rand.org/pubs/papers/P923.html.

[27] Christos Goumopoulos and Efthymios Housos. Efficient trip generation with a rule modeling system for crew scheduling problems. *Journal of Systems and Software*, 69:43–56, 2004. ISSN 01641212. doi: 10.1016/S0164-1212(03)00048-7.

[28] Karla L. Hoffman and Manfred Padberg. Solving Airline Crew Scheduling Problems by Branch-and-Cut. *Management Science*, 39 (6):657–682, 1993. ISSN 0025-1909. doi: 10.1287/mnsc.39.6.657.

[29] Atoosa Kasirzadeh, Mohammed Saddoune, and François Soumis. Airline crew scheduling: models, algorithms, and data sets. *EURO Journal on Transportation and Logistics*, 6:111–137, 2017. ISSN 2192-4376. doi: 10.1007/s13676-015-0080-x.

[30] Nabil Kenan, Aida Jebali, and Ali Diabat. The integrated aircraft routing problem with optional flights and delay considerations. *Transportation Research Part E: Logistics and Transportation Review*, 118:355–375, 2018. ISSN 13665545. doi: 10.1016/j.tre.2018.08.002. URL https://doi.org/10.1016/j.tre.2018.08.002.

[31] Diego Klabjan, Ellis L. Johnson, George L. Nemhauser, Eric Gelman, and Srini Ramaswamy. Solving large airline crew scheduling problems: Random pairing generation and strong branching. *Computational Optimization and Applications*, 20(1):73–91, 2001. ISSN 09266003. doi: 10.1023/A:1011223523191.

[32] Diego Klabjan, Ellis L. Johnson, George L. Nemhauser, Eric Gelman, and Srini Ramaswamy. Airline Crew Scheduling with Time Windows and Plane-Count Constraints. *Transportation Science*, 36(3):337–348, 2002. ISSN 0041-1655. doi: 10.1287/trsc.36.3.337.7831.

[33] Diego Klabjan, Ellis L. Johnson, George L. Nemhauser, Eric Gelman, and Srini Ramaswamy. Airline Crew Scheduling with Regularity. *Transportation Science*, 35(4):359–374, 2003. ISSN 0041-1655. doi: 10.1287/trsc.35.4.359.10437.

[34] Niklas Kohl and Stefan E. Karisch. Airline crew rostering: Problem types, modeling, and optimization. *Annals of Operations Research*, 127:223–257, 2004. ISSN 15729338. doi: 10.1023/B:ANOR.0000019091.54417.ca.

[35] Anne Mercier and François Soumis. An integrated aircraft routing, crew scheduling and flight retiming model. *Computers and Operations Research*, 34:2251–2265, 2007. ISSN 03050548. doi: 10.1016/j.cor.2005.09.001.

[36] Anne Mercier, Jean François Cordeau, and François Soumis. A computational study of Benders decomposition for the integrated aircraft routing and crew scheduling problem. *Computers and Operations Research*, 32:1451–1476, 2005. ISSN 03050548. doi: 10.1016/j.cor.2003.11.013.

[37] Airport Coordination Netherlands. Declared capacity ams summer 2019. URL https://slotcoordination.nl/slot-allocation/declared-capacity/.

[38] H. Timucin Ozdemir and Chilukuri K. Mohan. Flight graph based genetic algorithm for crew scheduling in airlines. *Information Sciences*, 133:165–173, 2001. ISSN 00200255. doi: 10.1016/S0020-0255(01)00083-4.

[39] Nikolaos Papadakos. Integrated airline scheduling. *Computers & Operations Research*, 36:176–195, 2009. ISSN 1860949X. doi: 10.1016/j.cor.2007.08.002.

[40] Frédéric Quesnel, Guy Desaulniers, and François Soumis. A new heuristic branching scheme for the crew pairing problem with base constraints. *Computers and Operations Research*, 80:159–172, 2017. ISSN 03050548. doi: 10.1016/j.cor.2016.11.020.

[41] Jerrold Rubin. A Technique for the Solution of Massive Set Covering Problems, with Application to Airline Crew Scheduling. *Transportation Science*, 7(1):34–48, 1973. ISSN 0041-1655. doi: 10.1287/trsc.7.1.34.

[42] Mohammed Saddoune, Guy Desaulniers, Issmail Elhallaoui, and Franois Soumis. Integrated airline crew scheduling: A bi-dynamic constraint aggregation method using neighborhoods. *European Journal of Operational Research*, 212:445–454, 2011. ISSN 03772217. doi: 10.1016/j.ejor.2011.02.009. URL http://dx.doi.org/10.1016/j.ejor.2011.02.009.

[43] Mohammed Saddoune, Guy Desaulniers, Issmail Elhallaoui, and François Soumis. Integrated Airline Crew Pairing and Crew Assignment by Dynamic Constraint Aggregation. *Transportation Science*, 46(1):39–55, 2012. ISSN 0041-1655. doi: 10.1287/trsc.1110.0379.

[44] Mohammed Saddoune, Guy Desaulniers, and François Soumis. Aircrew pairings with possible repetitions of the same flight number. *Computers and Operations Research*, 40:805–814, 2013. ISSN 03050548. doi: 10.1016/j.cor.2010.11.003.

[45] Juan José Salazar-González. Approaches to solve the fleet-assignment, aircraft-routing, crew-pairing and crew-rostering problems of a regional carrier. *Omega (United Kingdom)*, 43:71–82, 2014. ISSN 03050483. doi: 10.1016/j.omega.2013.06.006. URL `http://dx.doi.org/10.1016/j.omega.2013.06.006`.

[46] Rivi Sandhu and Diego Klabjan. Integrated Airline Fleeting and Crew-Pairing Decisions. *Operations Research*, 55(3):439–456, 2007. ISSN 0030-364X. doi: 10.1287/opre.1070.0395.

[47] Nadia Souai and Jacques Teghem. Genetic algorithm based approach for the integrated airline crew-pairing and rostering problem. *European Journal of Operational Research*, 199:674–683, 2009. ISSN 03772217. doi: 10.1016/j.ejor.2007.10.065. URL `http://dx.doi.org/10.1016/j.ejor.2007.10.065`.

[48] Bernard W. Taylor III. Module C. Integer Programming: The Branch and Bound Method. In *Introduction to Management Science*. Pearson, 11th edition, 2012.

[49] Toon Weyens and Daan van Vugt. `Pylgrim`– Elementary Shortest Path Problem with or without Resource Constraint.

[50] David Torres Sanchez. `cspy`– A Python package with a collection of algorithms for the (Resource) Constrained Shortest Path problem, 2019. URL `https://github.com/torressa/cspy`.

[51] Shangyao Yan, Tun-Tai Tung, and Yu-Ping Tu. Optimal construction of airline individual crew pairings. *Computers and Operations Research*, 29:341–363, 2002. ISSN 03050548. doi: 10.1016/S0305-0548(00)00070-8.

[52] Bahadir Zeren and Ibrahim Özkol. A novel column generation strategy for large scale airline crew pairing problems. *Expert Systems with Applications*, 55:133–144, 2016. ISSN 09574174. doi: 10.1016/j.eswa.2016.01.045.

[53] Bahadir Zeren and Äbrahim Özkol. An Improved Genetic Algorithm for Crew Pairing Optimization. *Journal of Intelligent Learning Systems and Applications*, 04(01):70–80, 2012. ISSN 2150-8402. doi: 10.4236/jilsa.2012.41007.