Thesis for the degree Master of Science in Business Administration and Engineering

An Adaptive Large Neighborhood Search Algorithm for the Tail Assignment Problem of Airlines

Andreas Hottenrott

RWTH Aachen University
School of Business and Economics
Chair of Operations Research

Aachen, June 2015

Author
Andreas Hottenrott
Student ID: 288287

Professor:
Prof. Dr. Marco Lübbecke

External supervisors:
Dr. Ulrich Dorndorf
Dr. Sebastian Brand

Date of graduation:
Aachen, 16 June 2015

Abstract

Given a set of aircraft and a set of scheduled flights, the tail assignment problem determines the sequences of flights, ground periods and maintenance activities for each individual aircraft. The routings have to comply with various operational requirements stipulated by airlines, airports and authorities. Of particular importance are restrictions on the maximum accumulated flying time, number of take offs and calendar time between two consecutive maintenance checks.

In the literature and also in practice, fleeting and routing are usually solved individually and sequentially. Tail assignment is often treated as a pure feasibility problem as the associated costs are relatively low compared to other planning stages. In doing so, airlines give up a great potential to improve their operational efficiency. Our approach is different as it minimizes tail assignment costs while providing the possibility to readjust fleeting decisions.

We introduce an adaptive large neighborhood search algorithm for the tail assignment problem of airlines. Throughout the algorithm, we take advantage of the benefits of two different formulation techniques of how to map a flight schedule in a mathematical model. To quickly obtain an initial solution, we embed an inexpressive problem formulation in an iteratively solved row generation framework. In the subsequent improvement iterations, we employ a more expressive problem formulation. The model is able to capture detailed maintenance requirements and all other relevant operational restrictions, such as minimum turn times, curfews and maintenance capacities.

We show how a math-heuristic framework of the adaptive large neighborhood search can be designed to obtain good solutions to the tail assignment problem in acceptable time frames. To reduce problem sizes and increase solving speed, we develop and evaluate several preprocessing methods. The performance of our approach is demonstrated using real-world data from two major international carriers.

Keywords: *airline scheduling, tail assignment, aircraft routing, large neighborhood search, adaptive large neighborhood search, math-heuristic*

Declaration of Authorship

I confirm that this Master's thesis is my own work and I have documented all sources and material used. This thesis was not previously presented to another examination board and has not been published.

Aachen, 16 June 2015

Andreas Hottenrott

Acknowledgments

The work presented in this thesis has been developed in collaboration with the Chair of Operations Research at RWTH Aachen University and INFORM GmbH in Aachen. INFORM GmbH is a leading IT company specialized in intelligent planning and logistics decision-making software based on mathematical optimization algorithms. At this point, I would like to take the opportunity to express my gratitute to those people who helped me realize this work and supported me in my research.

My special thanks go to Prof. Dr. Marco Lübbecke from the Chair of Operations Research at RWTH Aachen University for the academic supervision of my thesis. Furthermore, I wish to thank Dr. Ulrich Dorndorf and Dr. Sebastian Brand from INFORM GmbH for the sympathic collaboration. I expressly thank Dr. Dorndorf for the given opportunity and his confidence. My gratitude also goes to Dr. Brand for his encouragement and generous support during my research. With his advice and ideas as well as his knowledge and experience, he decisively contributed to the success of this research. In addition, I thank all other employees at INFORM GmbH who have helped me during the preparation of this thesis. Finally, I wish to express my thanks to my family for their endless support in all situations of my life.

Andreas Hottenrott
Aachen, June 2015

Contents

1 Introduction **1**
- 1.1 Overview . . . 1
- 1.2 Motivation . . . 2
- 1.3 Outline . . . 4

2 Airline operations planning **5**
- 2.1 Market environment . . . 5
- 2.2 Planning process . . . 7

3 Tail assignment **13**
- 3.1 Problem description . . . 13
- 3.2 Objectives in tail assignment . . . 15
- 3.3 Restrictions in tail assignment . . . 16

4 Literature review **21**
- 4.1 Modeling techniques and solution methodologies . . . 21
- 4.2 Aircraft maintenance routing literature . . . 27
- 4.3 Tail assignment literature . . . 29
- 4.4 Integration with other planning stages . . . 31
- 4.5 Contribution . . . 32

5 Large neighborhood search **33**
- 5.1 Overview . . . 33
- 5.2 Process of a large neighborhood search . . . 34
- 5.3 Adaptive large neighborhood search . . . 36
- 5.4 Benefits compared to other solution approaches . . . 38

6 Mathematical model **39**
- 6.1 Choice of modeling technique . . . 39
- 6.2 Main model . . . 40
- 6.3 Model enhancements . . . 44
- 6.4 Maintenance capacity . . . 47

7 Solution approach **49**
- 7.1 Initial solution stage . . . 49
- 7.2 Improvement stage . . . 71

8 Analysis and evaluation **79**
8.1 Generation of test instances 79
8.2 Computational results 81

9 Conclusion **87**
9.1 Summary 87
9.2 Future work 89

Bibliography **93**

List of Figures

1.1 Outline of this thesis 4

2.1 Trends in the airline industry 7
2.2 Traditional planning process of an airline 9

3.1 Classification of maintenance constraints 18

4.1 Example of a time-space network 23
4.2 Example of a connection network 24

5.1 Process of a large neighborhood search 34

6.1 Enhancements of connection networks 46

7.1 Structure of Benders framework 50
7.2 Change of problem characteristics depending on arc definition 52
7.3 Relation between solving times and schedule size for Carrier One . . . 53
7.4 Example schedule 55
7.5 Event dates in time-space network 59
7.6 Minimal forbidden sequence 61
7.7 Illustration of model enhancements 63
7.8 Overnight destructor [44] 73
7.9 Route swap destructor [44] 74
7.10 Time-space destructor 74

8.1 Performance of ALNS algorithm across 208 instances 83
8.2 Destructor benchmarking 84
8.3 Solving times of ALNS algorithm and exact approach 85
8.4 Fair comparison of ALNS and exact approach 86
8.5 Progress of ALNS and exact search on instance with 50 activities . . 86

List of Tables

7.1 Example instances 50
7.2 Benchmarking: main model as feasibility problem 51
7.3 Pseudo-code for FIFO heuristic 54
7.4 Pseudo-code for priority heuristic 57
7.5 Evaluation of aircraft aggregation enhancement 64
7.6 Evaluation of node aggregation enhancement 65
7.7 Evaluation of island isolation enhancement 66
7.8 Evaluation of activity aggregation enhancement 67
7.9 Evaluation of possible objective functions 69
7.10 Evaluation of strategies to generate Benders cuts 70
7.11 Candidate values for the ALNS parameters 77
7.12 Solving times and objective values given different reaction factors r . 78
7.13 Aggregated deviation for both user profiles 78
7.14 Optimized parameter values for ALNS framework 78

8.1 ALNS solution of example instances 82

Nomenclature

ALNS	adaptive large neighborhood search
APU	auxiliary power unit
EASA	European Aviation Safety Agency
ETOPS	Extended-range Twin-engine Operational Performance Standards
FAA	Federal Aviation Administration
FIFO	first-in, first-out
GRASP	greedy randomized adaptive search procedure
ISC	Industry Steering Committee
LNS	large neighborhood search
LP	linear programming
MIP	mixed-integer program
MRB	Maintenance Review Board
MRBR	Maintenance Review Board Report
OR	operations research
RMP	restricted master problem
WG	Working Groups

1 Introduction

The airline industry is a dynamic and highly competitive business sector. Airlines continuously need to adapt to various influencing market forces. New players and low-cost carriers have created a market environment characterized by excess capacities and low fares. To remain profitable, the efficient management of resources, especially aircraft and crews, is inevitable. In this context, airlines rely on the methods and tools of operations research. Major airlines are confronted with huge amount of data, which is impossible to handle manually. One of our airline partners offers nearly 3,500 daily flights to 275 destinations in 50 countries using more than 700 aircraft of twelve different fleet types.

1.1 Overview

The planning tasks of an airline include schedule design, fleet assignment, aircraft scheduling and crew scheduling. Traditionally, these planning stages are solved individually and sequentially. After developing the flight schedule and assigning a fleet type to every flight, the airline seeks to create routings for its aircraft and work schedules for its crews. This thesis deals with the tail assignment problem, which is part of the aircraft scheduling stage. The name of the problem results from the fact that aircraft are identified by their registration number, which is indicated on the rear end of the aircraft. The problem consists of creating flight routes for individual aircraft such that all flights in a schedule are covered and diverse operational restrictions fulfilled. The most crucial restrictions are the maintenance requirements stipulated by the aviation authorities. The planner has to make sure that the routes of the aircraft comply with these requirements to avoid aircraft from being grounded. Other operational restrictions arise from minimum turn time requirements, limited maintenance capacity, multi-leg flights as well as preassigned and forbidden activities for individual aircraft.

The purpose of this thesis is to develop an optimizing tool for the tail assignment problem of airlines. The tool should be able to create valid aircraft routings while optimizing maintenance planning. Furthermore, it should provide the possibility to consider fleeting and routing decisions simultaneously. The generated routing and maintenance plans have to be suitable for real-world implementation. The solution therefore has to comply with all relevant restrictions of the real-world problem, especially the complex, interdependent maintenance requirements. The planning horizon

should cover a fixed period of approximately one week. To derive valid routings, the initial position and maintenance state of the aircraft have to be considered.

An explicit requirement is to create a flexible and technically unspecialized optimizing system. During the development of a prototype for a software, massive changes of customer requirements are probable. The system therefore has to be designed in a way that allows fast and easy adjustments to future customer requests. Also, the tool should take into account different preferences of the user. It should be able to quickly generate solutions of acceptable quality, as well as solution of higher quality when the user is willing to wait longer.

Throughout this thesis, we develop a comprehensive mathematical representation of the real-world tail assignment problem. Our model is distinctive as it is able to capture various operational restrictions and allows multiple fleets to be treated simultaneously. We then present a math-heuristic to obtain problem solutions. Our algorithm is based on an adaptive large neighborhood search. The underlying idea is to iteratively improve an initial solution by alternately destroying and repairing different parts of the assignments. We implement multiple competing destructor methods in an adaptive framework, enabling the algorithm to adapt to the instance at hand and state of the search.

We prove our concept on 220 real-world instances from two major international carriers. The instances are heterogeneous. They cover planning periods from three days to one month. The largest instances involve up to 180 aircraft and 2,500 flights.

1.2 Motivation

The current market situation poses a big challenge for airlines. Fierce competition, cost pressure and changes in customer requirements put many airlines in difficult financial situations. Losses, bankruptcies and low profit margins are omnipresent. In order to survive, airlines are forced to increase the efficiency of their operations. Improvements in aircraft routing and maintenance planning are promising candidates. Compared to other stages of the airline planning process, they still offer a great potential for cost reductions.

The majority of existing studies on tail assignment is incapable of capturing all relevant objectives and restrictions of the real-world problem. Therefore, these studies are unsuitable for operational implementation, not to mention for optimization of maintenance planning. Tail assignment is often treated as a pure feasibility problem, because the incurred costs are relatively low compared to other planning stages, such as fleet assignment and crew scheduling. In neglecting the financial impact of unnecessary and untimely maintenance, airlines disregard a great potential to reduce their operational costs. We present an approach in which detailed maintenance requirements and many other operational restrictions are considered while incurred costs are minimized. Our research supports airlines to increase the utilization of their

expensive aircraft resources and improve their competitive position. The resulting routing and maintenance plans are suitable for direct implementation.

Most airlines take a sequential approach when planning their operations. The sequential approach is justified by organizational and computational reasons, but contains major drawbacks. Decisions taken in early planning stages limit the solution space in succeeding stages and might result in a final solution that is far away from the global optimum. Many planning stages are carried out weeks or even months before the day of operation, using historic data and unreliable forecasts as key inputs. Often the assumptions turn out to be inaccurate, requiring costly adjustments close to the day of operation. For example, the actual passenger demand on a flight is unknown during fleet assignment. Airlines counteract by incorporating a late refleeting stage in their planning process to reoptimize their fleeting decisions close to the day of operation. They aim to improve the generated profit by better matching offered capacity with actual demand. However, changes in fleet assignment require additional adjustment loops in aircraft and crew scheduling.

In this thesis, we present an alternative paradigm. We propose a tail assignment model that considers several fleets simultaneously and includes fleeting decisions. We thereby provide the opportunity for a late refleeting while avoiding costly iteration loops in aircraft scheduling. By optimizing fleeting and routing decisions simultaneously in an integrated approach, we expect to obtain solutions that are closer to the global optimum. We thus again contribute to increase profitability of the airline.

The operations of an airline are subject to many influencing factors. The frequent occurrence of unplanned maintenance needs, schedule disruptions, delays and cancellations force airlines to continuously revise their planning and adapt the routings of their aircraft to new information. Our proposed solution approach generates solutions of high quality in acceptable time frames satisfying the airlines' need for frequent replanning.

By its nature, tail assignment is a combinatorial optimization problem that is computationally hard in practice. To handle the huge amount of decision variables and constraints appearing in real-world instances, efficient model formulations and solution methodologies have to be applied. A deficit of exact approaches, like column generation, is that they are not aimed at finding solutions quickly. Heuristic approaches, on the other hand, are very promising for finding good suboptimal solutions while applying a reasonable computational effort. These characteristics make them ideally suited for the dynamic operational planning of an airline.

Our solution algorithm is based on a comprehensive mathematical model in conjunction with an adaptive large neighborhood search. The proposed math-heuristic fulfils the requirement to be flexible and adjustable. Future customer requirements can be easily included in the model without compromising the efficiency of our solution approach.

1.3 Outline

The structure of this thesis is organized in nine chapters as illustrated in Fig. 1.1. Subsequent to this introduction, we analyze the current market situation in the airline industry and introduce the planning process of an airline (Chap. 2). In Chap. 3, we provide a detailed overview on the tail assignment problem, its objectives and constraints. Chap. 4 discusses modeling techniques as well as solution methodologies for the tail assignment problem and reviews the relevant literature. In Chap. 5, we provide a theoretical background on the large neighborhood search, our solution methodology. In Chap. 6, we introduce a comprehensive mathematical model for the tail assignment problem. Our solution algorithm is described in Chap. 7. We present computational results and evaluate the performance of our solution tool on real-world instances in Chap. 8. Chap. 9 provides a summary of our work and identifies future research possibilities.

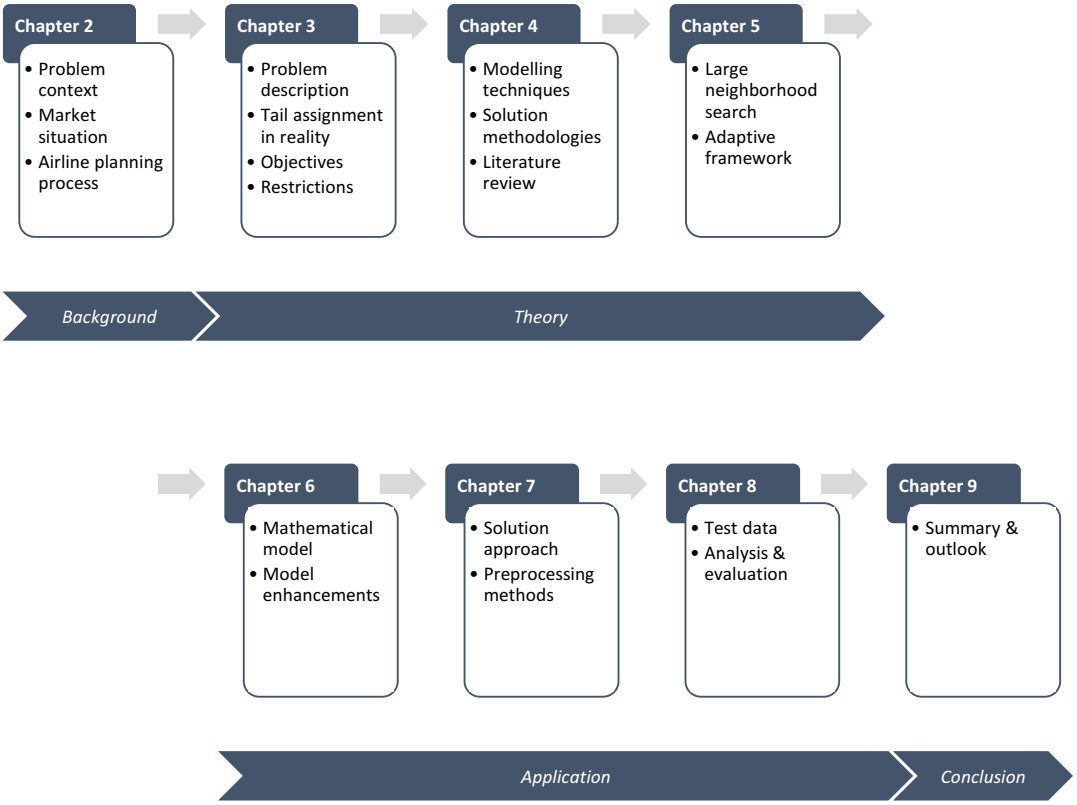


Figure 1.1: Outline of this thesis

2 Airline operations planning

In this chapter, we provide an overview on the planning tasks of an airline. To point out the need for efficient planning, we start by taking a look at the development and current state of the airline market (Sec. 2.1). Afterwards, we describe the traditional planning process of an airline (Sec. 2.2).

2.1 Market environment

The commercial aviation has developed impressively since the first passenger flight 100 years ago [53]. Since the 1970s, the aviation sector is one of the strongest growing traffic sectors. The average annual growth rate of passenger demand in the past 30 years was five percent[1] [41]. External factors such as the events of 9/11, economic downturns, or the SARS epidemic only led to a temporary slowdown of growth [6]. A significant event in the development of the airline industry was the deregulation of the American aviation sector in 1978. After the commencement of the *United States Airline Deregulation Act*, American airlines got the right to decide independently on their flight schedules, networks and ticket prices. At the same time, governmental protection disappeared, which led to the formations of many new competitors. As consequences, traffic and network growth increased rapidly, but also market dynamics and uncertainties [13, 18].

In 2013, the global passenger traffic grew by 5.2 percent [41]. A total of five trillion passenger kilometers were flown [75]. The total profit of all airlines worldwide amounted to $7.6 billion [40]. Also, the future development is predicted to be promising. Boeing forecasts an annual growth of passenger and cargo traffic of approximately five percent in the next 20 years. For the same period, Boeing estimates that 36,770 new airplanes are needed: 15,500 to replace old ones and 21,270 to meet the growing demand [18]. Especially, the demand for single-aisle aircraft will be high due to the fast growth of low-cost carriers.

In spite of those promising figures, many airlines operate at the edge of profitability or struggle with losses. In seven out of the nine years from 2000 to 2008, the U.S. airline industry spent more money than they earned. In 2008 alone, U.S.-based airlines generated a combined operating loss of $5.6 billion [71]. This trend

[1] All quoted growth rates are based on either revenue passenger kilometers to measure actual passenger traffic or revenue tonne kilometers to measure actual cargo traffic.

changed after 2008. In 2012, American airline companies generated a combined profit of $5 billion [71]. Nevertheless, profit margins are marginal. Out of every dollar generated through revenue, an airline typically makes less than one cent profit. Compared to other industries, this key figure is extremely low [75]. Apple Inc., for example, had a profit margin of 25 percent in 2012 [59]. There are different reasons for this situation. One reason is the fierce competition among the airlines. At the same time, the supplying aircraft manufacturers face little competition. Only two companies, namely Airbus and Boeing, share most of the market and thus possess a strong competitive position. Airports and air traffic control are also in a monopoly position towards airlines [75]. To survive in this environment, many airlines tried to increase their market share by adding additional capacities to their network routes. Air fares were cut due to a surplus of supply over demand. Only in the past few years, airlines started to incorporate a strict capacity discipline to their strategies and air fares started to rise again [71]. Another reason for the low profit margins are volatile, unpredictable cost parameters. Especially, fuel prices emerged to play a dominant role. Accounting for approximately 35 percent of total costs, fuel nowadays represents the largest cost factor of an airline, followed by crew expenses [29, 49, 59].

We can identify several trends that will affect the business models of airlines in the future. A major trend is the rise of low-cost carriers in the past two decades. Those airlines operate with 20 to 40 percent lower operational costs than flag carriers. At the same time, a consolidation process among the flag carriers is noticeable. Many former renowned airlines go bankrupt or merge with other financially stricken competitors. They struggle with old fleets, large networks, unionized staff and large pension liabilities [47, 75]. The bankruptcy of American Airlines in 2011 and the merger of Air France-KLM in 2004 are two prominent examples.

Another challenge for many flag carriers, especially European, is the fast growth of state-subsidized airlines from the Persian Gulf. Those airlines grow a lot faster than the global average [18]. Tim Clark, CEO of the state-owned Emirates airline from Dubai, targets a continuous profit growth of eight to ten percent per year due to modern aircraft and new routes [63]. According to him, Emirates will expand its fleet to 250 aircraft and serve 70 million annual passengers by 2020. Emirates will then be the world's largest airline by international passenger traffic [55].

Besides the Gulf region, there are other emerging markets which will play a significant role in future airline strategies. Above all, this will be China and the Asia Pacific region. With increasing income and economic wealth, more travelers will choose an air flight instead of slower, less comfortable modes of transportation. The Chinese domestic market is growing more rapidly than any other region in the world [6]. China already starts to produce own airplanes and will soon be challenging Airbus and Boeing on the global market [68]. Similar developments can be observed in India, Brazil and Russia. Another major trend is environmental awareness. Some countries have already implemented additional environmental charges while others plan to do so in the near future. Fig. 2.1 summarizes the identified trends in the airline industry.

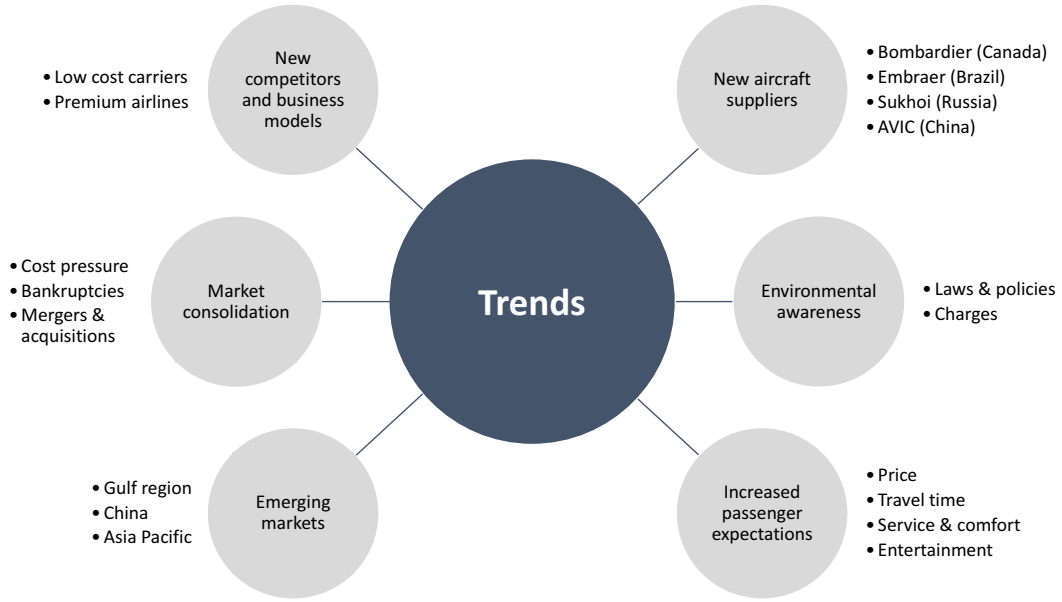


Figure 2.1: Trends in the airline industry

In order to survive in this increasingly competitive, dynamic and uncertain market environment, airlines need to optimize the utilization of their resources, in particular aircraft and crews [13, 29]. The decisions must take into account all influencing market forces as well as the desires of the customers. The complexity of the planning problems requires adequate tools to support decision-making.

Airlines have a long history of applying methods and techniques of operations research (OR) to their planning problems. Due to the advances in computer technology and optimization models in the past years and decades, OR tools are able to solve larger problems with increased complexity in shorter time. Major airlines have established in-house OR departments and seek a close cooperation with correspondent research institutes [13, 37, 65]. By optimizing their planning, airlines can improve their profitability, which allows them to invest additional resources for future growth. Those investments again are needed to enhance the offered services and operational efficiency. Airlines seek to achieve a maximum of operational efficiency on all levels of their planning process [18].

2.2 Planning process

The operations planning of an airline is a continuous process, which starts up to five years in advance of the day of operation [42]. It involves various decisions on different levels of management. In the early stages, strategic decisions are made based on long planning horizons. These decisions involve the selection of flight destinations and flight times. With advancing time and approaching the day of operation, the

responsibility for planning shifts to lower management levels, who assign aircraft and crews to every flight. In the end, operational decisions are made with daily or even hourly planning horizons. In this phase, the planners define countermeasures to react to disruptions and other unpredictable events during operation.

The decisions to be made do not only differ in planning horizon, but also in their scope, objectives and group of involved stakeholders. The decision-making is limited by various restrictions. Airport, fleet and aircraft characteristics are important restrictions when working out the flight schedule and assigning aircraft to flights. In addition, maintenance requirements limit the leeway for aircraft assignment decisions. During crew planning, requirements of the Human Resources Department have to be considered. Applicable laws and policies are important restrictions for all decisions in the planning process.

Understandably, mapping the entire planning process in a single OR model would result in an overall optimized solution and the biggest benefit [11]. However, due to the complexity and heterogeneity of the decisions, this is hardly possible. According to Haouri et al., the size of such a model would be too large to be manageable by standard solution methodologies, even for a moderately sized airline [33]. To optimize their operations, airlines divide the planning process into several stages. Traditionally, these stages are solved separately and sequentially. The outputs of upstream stages constitute the inputs for downstream stages. For many stages, OR models have been successful in supporting the decision-making of airlines [11, 50].

2.2.1 Traditional planning approach

The airline planning process is usually divided into five stages: flight scheduling, fleet assignment, aircraft scheduling, crew scheduling and disruption management. The chronology and interdependencies of these stages are illustrated in Fig. 2.2.

Flight scheduling

Starting point of the planning process is the construction of the flight timetable, also called flight schedule. In this strategic stage, the planners decide on served destinations, flight times and flight frequencies [13]. The initial schedule is usually designed more than a year in advance of the operating day. Decisions on introducing new destinations are made even earlier. The scheduling decisions are critical, because the choice of flight destinations and travel times affect the number of attracted passengers and the potential flight fares [42]. Airlines seek to create a schedule that best matches passenger travel demand. To predict the demand on various origin-destination relations, extensive and reliable forecasts are required [76].

Many airlines, especially flag carriers, have established *hub-and-spoke networks* to serve a maximum number of origin-destination relations with a minimum number of

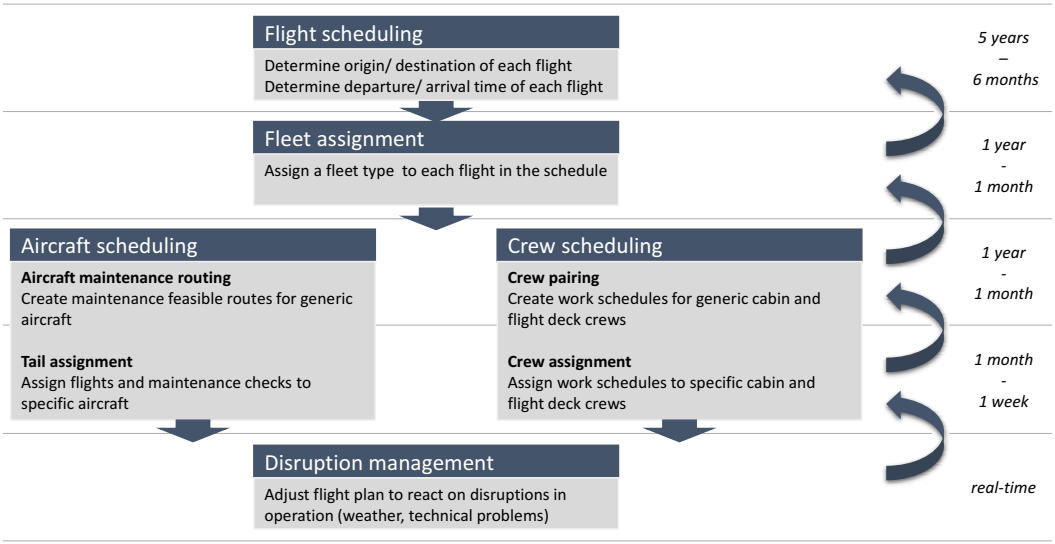


Figure 2.2: Traditional planning process of an airline

flights. An immanent disadvantage of hub-and-spoke networks compared to *point-to-point networks* is that passengers need to transfer flights at the airline's hub. For hub-and-spoke operators, the flight time decisions are especially important to provide convenient and fast connections to onward flights [42].

The design of the schedule is restricted by airport characteristics, e.g., the availability of slots and night flight bans. International flights are subject to bilateral agreements and government allocations [11]. Also, airline alliances are a factor to consider [29]. The flight schedule defines the core product sold by the airline and forms the basis for all subsequent planning steps.

Fleet assignment

In the tactical fleet assignment, the flights in the schedule are assigned to an aircraft type (fleet). To decide on an assignment, the planners trade the expected revenues against the operational costs of using a particular fleet on a flight [50]. Fleets with a large seating capacity generate more revenue when assigned to high-demand flights. On the other hand, smaller aircraft have lower operational costs, especially on short-haul routes [11, 70]. The fleet assignment is restricted by the technical characteristics of the fleet, e.g., the maximum range and required runway length. Besides the financial and technical aspects, the fleet assignment is also driven by marketing aspects. Marketing departments try to enter new markets or increase shares in important markets by assigning new, modern fleets to corresponding routes.

The fleet assignment is carried out one month to one year prior to the departure. It has to be performed early for two reasons. First, it checks on a very high level if the flight schedule is operable. That is, it ensures that the aircraft flow is balanced

and sufficient turn time is available to perform the connections between two flights. Second, the fleet assignment is a major input for the subsequent planning stages. Depending on the assigned fleet, different operational requirements apply for aircraft and crew scheduling. As the solution of the fleet assignment decomposes the flight schedule into flight schedules for separate fleets, the following planning steps can be solved on a homogeneous set of aircraft [13, 29, 76].

Aircraft scheduling

In the aircraft scheduling stage, the flights in the schedule are assigned to individual aircraft. While the fleet assignment only checks the operability of the schedule on a very high level, detailed operational restrictions, especially maintenance intervals, are incorporated in this stage. The planners have to consider different levels of maintenance checks, which vary in scope, frequency, duration and cost [13, 22]. The aircraft scheduling is often divided into two separate stages, the aircraft maintenance routing and the tail assignment stage [42].

The assignment of flights to specific aircraft is done in the tail assignment stage and is subject to the current location and maintenance condition of the aircraft. Due to disruptions and replanning, this information is difficult to forecast. It becomes increasingly more accurate when approaching the day of operation. Therefore, tail assignment is usually done only a few days or weeks before the departure of a flight [28]. The disadvantage is that deficits in the schedule concerning maintainability and other operational requirements are detected very late. In response, many airlines insert an additional planning step in-between the fleet and tail assignment. This is referred to as aircraft maintenance routing and can be seen as a tail anonymous version of the tail assignment problem [28]. The problem consists of creating generic routes such that enough maintenance opportunities are provided for all aircraft. The routes are to be flown by a single, yet unspecified, aircraft [62].

In general, the aircraft maintenance routing can only take into account the most frequent maintenance requirements, since the planning of less frequent checks depends on the flying and maintenance history of the specific aircraft [12, 62]. To guarantee valid assignments in the tail assignment stage, the planners have to readjust the generic routes [62]. Due to marketing or technical issues, some aircraft may be restricted from performing certain flights. Similarly, some activities may be preassigned to particular aircraft. The outputs of the tail assignment are the routings of the aircraft as well as a detailed plan of maintenance activities.

Crew scheduling

The process of crew scheduling is similar to aircraft scheduling. In the crew pairing stage, anonymous work schedules are generated. In the subsequent crew assignment stage, the work schedules are assigned to specific crew members [11].

The objective in the crew pairing stage is to find a set of work schedules that minimizes total crew cost. The work schedules are called pairings and consist of duty and rest periods that are performed by a single, yet anonymous, crew member. Crew pairings are conceptually the same as aircraft routings, as both describe a sequence of flights and ground times assigned to a single resource [44]. The planners have to make sure that the required number of crew members is assigned to the flights and that the work schedules comply with applicable laws and policies dictated by authorities and labor unions [11, 29].

In the crew assignment stage, the planners create extended work schedules for all employees for a period of typically one month. They need to consider vacation requests and fixed training sessions [11]. Most airlines also try to consider the employees' preferences during the assignment. There are two different approaches. In Europe, rostering is commonly used. Rostering means that the requests of the crew members are explicitly considered when designing their individual work schedules. American carriers, in contrast, commonly use a preferential bidding system. In this approach, anonymous extended work schedules are created and the employees set individual priorities on the available schedules. The assignment of schedules to employees is then usually based on seniority [11].

Disruption management

Once the tail and the crew assignment stages are finalized, the operations plan is finished. However, disruptions are unpreventable occurrences in the daily operations of an airline. They occur when equipment failures make flying unsafe, when bad weather shuts down airports, or when required flight crews are unavailable [8]. The job of the operations controller is to react to any disruption and define countermeasures to run operations as close to plan as possible while minimizing delays, cancellations and customer inconvenience [11]. When delays occur, successive flights can be deliberately delayed to make sure that connecting passengers and crews reach their onward flights. In addition, the cruise speed may be increased to compensate for delays. When a flight is cancelled, successive flights either need to be cancelled or a substitute aircraft and crew have to be provided. The decisions in this stage are made in real-time [11].

2.2.2 Disadvantages of the traditional approach

The hierarchical, sequential approach in planning the operations of an airline is common industry practice. Due to the breakdown of the entire planning, the subproblems become tractable and solvable. However, the approach involves a number of substantial drawbacks. The optimization of subproblems does not necessarily lead to an overall optimized solution. The decisions are taken by different departments on different management levels which are driven by different key figures. As the

outcomes of precedent stages restrict decision-making in subsequent stages, it is not guaranteed that fixed decisions allow good or even feasible solutions in subsequent stages. For example, the optimal fleet assignment might incur high maintenance costs in the subsequent aircraft scheduling stage. When no feasible solution exist, iteration loops are inevitable (see Fig. 2.2).

Other drawbacks are the long lead times between planning and execution. Many decisions are made months or even years before the day of operation, using inaccurate and unreliable forecasts as key input. During the long time between planning and execution, substantial changes of the general conditions may occur which influence the benefits of taken decisions [62].

2.2.3 Trends

Recently, researchers started to investigate the integration of multiple planning stages. There are multiple reasons for this trend. First, it is obvious that solving integrated problems results in solutions that are closer to the global optimum. Second, the increased computational performance in the past years enables examining larger problem sizes with smaller time requirements. Third, the development and enhancement of efficient solution methodologies, like column generation and intelligent heuristics, allow the handling of more complex problems. Finally, due to major progresses in data availability, data analysis and forecasting, decisions of different time horizons can be optimized simultaneously with satisfactory results [42].

According to Grönkvist, an integration can be achieved in three ways [29]. The first option is to fully integrate different planning stages, which guarantees to find the best solution for the integrated problem. However, the problem size and complexity increase drastically when solving multiple stages simultaneously. The second option is to partially integrate different planning steps. For example, additional constraints could be included in the fleet assignment model to consider some restrictions of the aircraft scheduling stage. The third option is to iteratively solve different planning stages. In this approach, the planning steps communicate with each other providing feedback from a subsequent stage to an earlier stage which is then readjusted.

Another trend is to incorporate a late refleeting stage in the planning process. As information becomes significantly more accurate, it would be favorable to adjust fleeting and routing decisions close to the day of operation. According to Jiang and Barnhart, only half of the flight tickets are sold more than three weeks prior to the departure [43]. Refleeting would allow airlines to better match seat capacity with the demand on a flight, and thereby increase profit. Of course, the cost to readjust aircraft routes and crew schedules should not be higher than the profit growth through refleeting. Warburg et al. estimate that a profit growth of up to 3.47 percent can be achieved when refleeting is done 24 days prior to the day of operation and up to 7.46 percent when done within the last week prior to departure [74].

3 Tail assignment

This thesis aims to optimize the tail assignment process of airlines. In this chapter, we take a closer look at how the tail assignment problem is treated in reality and explain the underlying objectives and restrictions. At the same time, we describe the framework and assumptions of our optimizing approach.

3.1 Problem description

Tail assignment is the problem of creating operational feasible routes for individual aircraft covering a set of scheduled flights. A route represents a sequence of flights and maintenance activities that are performed consecutively by a specific aircraft. The construction of the route for an aircraft is subject to its current location, maintenance and flying history, and possibly tail-dependent restrictions [29, 62].

Most airlines solve the tail assignment a few days before the departure of a flight, when reliable information about the current state of the aircraft is available [62]. For other reasons, it is motivating to enlarge the planning horizon to several weeks. One reason is that there are maintenance checks that an aircraft does not need to perform frequently. In order to maximize the utilization before the next check, it is beneficial to look at a longer planning period. Another reason is the interdependency with crew assignment. It is desirable for crews to follow the routes of the aircraft. Such a solution reduces crew costs and is more robust regarding disruptions. As the work schedules for crews are usually designed for a one-month period, tail assignment should cover the same amount of time [21, 29, 62].

The choice of the appropriate period length is a tradeoff between different factors. In general, the longer the planning horizon, the more beneficial for maintenance and crew planning. On the other hand, the reliability of data forecasts decreases with a longer planning horizon. Due to the high frequency of stochastic events in the airline business, long planned routings are likely to be disturbed. Also, larger instances are more difficult and time-consuming to solve. In this study, we focus on solving the tail assignment for a planning period of one week. Nevertheless, our problem formulation is independent of the horizon length and capable of considering periods of several weeks.

Airlines solve the tail assignment for a specific time period considering restrictions for individual aircraft. Therefore, we set up a dated problem formulation based

on a finite planning horizon. This approach is common for the tail assignment problem [12, 29, 30, 62, 65]. In contrast, the aircraft maintenance routing is often modeled as a cyclic problem based on an infinite planning horizon [9, 27, 33, 44, 50, 51]. In a cyclic formulation, the researchers assume that the same daily/ weekly schedule is carried out every day/ week [50]. This formulation is reasonable for the aircraft maintenance routing stage where it is assumed that the fleets consist of generic aircraft. In the tail assignment stage, we want to consider individual characteristics of the aircraft, especially their flying and maintenance history. This implies that we need to have information about the position and maintenance state of the aircraft at the beginning of the planning horizon. It is also imaginable that some aircraft are preassigned or restricted to perform certain activities. We use the term activity to refer to flights and maintenance. The terms airport and station are used interchangeably throughout this thesis. A maintenance station is an airport at which the airline operates a maintenance base.

There are two different application scenarios for a tail assignment optimizer. In the first scenario, the airline requires an optimizer to derive the initial routes for the aircraft during a selected future planning period. The objective is to assign all activities in the schedule to the available aircraft. We can assume that the initial positions as well as the maintenance state of the aircraft are according to plan and allow all activities to be performed. The second scenario is to use an optimizer to recover from disruptions. That is, when the real operations deviate from the predefined plan due to bad weather or delays. In this scenario, the disruption manager needs a tool that readjusts the routes according to the new information in real-time. The objective is to return to the original plan as quickly as possible. It is likely that the conditions of the aircraft do not allow all activities to be performed. Our research is focused on the first scenario, the standard tail assignment procedure, and not on disruption recovery. We assume that all scheduled activities have to be performed. At the end of this thesis, we provide an outlook on how our model and solution process would need to be adjusted to allow activities to be left unassigned.

We have seen in the previous chapter that the fleet assignment decomposes the schedule into sets of activities for every fleet. Therefore, many airlines solve the tail assignment individually for every fleet [13, 76]. We have decided to develop a fleet-independent problem formulation allowing us to look at several fleets simultaneously. We see several advantages of our approach. First and most importantly, as the fleet assignment does not include detailed operational requirements, solving the tail assignment individually for every fleet might be infeasible [29]. Second, airlines tend to update their fleet assignment close to the day of operation. In allowing several fleets to be considered simultaneously, we support the refleeting purposes of airlines [62]. Finally, different fleets might share common resources, e.g., hangar space. Only when considering those fleets simultaneously, we can guarantee that the available resource capacity is respected.

We assume that the problems we solve are deterministic. This means that all information is available before we solve the problem and does not change during the

planning horizon. In reality, tail assignment is a stochastic problem based on unsure information and subject to disruptions during the planning horizon. We take account of the stochastic characteristics of the real-world problem by applying our approach within a rolling time horizon scheme. We propose to regularly resolve the tail assignment with updated information about the position and state of the aircraft.

Our optimizer has to be able to find good solutions within acceptable computation time. Many optimization systems are focused on pure optimization and only able to generate a single optimal solution at the end of every run. A shortcoming of such systems is that the user does not know in advance how long the run time will be. Finding the optimal solution can require disproportional longer run times than finding a solution that is close to the optimum. We create a framework that constantly generates feasible solutions while running. The longer the run time, the better the quality of the solutions. The user can decide when an acceptable solution quality has been achieved and stop the optimization process. Our approach follows the satisficing concept introduced by Nobel Laureate in economics Herbert Simon. The term satisficing is a coinage of satisfactory and optimizing. Simon argues that managers tend to seek solutions that are "good enough" rather than spending much effort in finding a solution that is presumably optimal regarding various desirable objectives [37].

3.2 Objectives in tail assignment

For the tail assignment problem, different objectives are imaginable. As the incurred costs are relatively low compared to fleet assignment and crew scheduling, some airlines treat tail assignment as a pure feasibility problem [27, 44, 51, 70]. Other airlines seek to minimize some cost function, representing real or fictitious costs.

A popular approach is to reward short and long ground times while penalizing medium length connections in the routes of the aircraft [28, 33, 50, 51]. During a medium length connection, i.e., around two or three hours, the aircraft cannot be used for other activities and may be causing high parking cost at a gate. Long ground times are less unattractive, because the aircraft may be used as standby aircraft in case of disruptions [29]. Extremely short connections, making it difficult for crews to change aircraft, are either forbidden or penalized as they are likely to cause disruptions [28].

Another approach is to maximize through revenues [13, 20, 33, 50, 51]. Through revenues are measured by the number of passengers that stay on the same aircraft between two consecutive flights. Especially when operating a hub-and-spoke network, maximizing through revenues can be a reasonable approach to alleviate the disadvantage of longer flight times compared to point-to-point operators [13].

Some airlines try to increase the robustness of the tail assignment solution by maximizing maintenance opportunities. They seek to design their aircrafts' routes such

that every aircraft frequently spends some time, e.g., a night, at a maintenance base. During maintenance opportunities, an aircraft does not necessarily undergo a maintenance check, but in case unpredictable maintenance activities are required, they can be incorporated easily in the route of the aircraft [13].

For our model, we choose a different objective. As we seek to solve a deterministic and dated problem, we focus on the real occurring maintenance effort. Our objective is to minimize the number of maintenance activities and penalize untimely maintenance, that is, whenever a check is performed before the allowed interval limits are reached. Early maintenance is undesirable, because flight capacity is wasted and parts are replaced before reaching their permitted lifetime [12]. In addition, we allow the user to set individual penalties on undesirable connections.

3.3 Restrictions in tail assignment

After analyzing the different objectives in tail assignment, we now summarize the relevant constraints. The airline business is an industry with strict safety and operational regulations. There are many restrictions that an airline has to consider when deciding on an aircraft to perform a flight. Especially, maintenance requirements limit the leeway in decision-making. But also characteristics of the aircraft and fleets impose assignment restrictions.

3.3.1 Maintenance

An aircraft in service has to undergo periodic maintenance inspections to prove its airworthiness. The content of the inspections as well as the inspection intervals differ from fleet to fleet. In the United States, maintenance regulations are specified by the *Federal Aviation Administration* (FAA), in Europe by the *European Aviation Safety Agency* (EASA).

Before entering into service, the minimum scheduled maintenance requirements for a new aircraft type and its power plants are set. This is the responsibility of the *Maintenance Review Board* (MRB), a panel of representatives of the civil aviation authorities (e.g., FAA, EASA). The MRB gets technical support from the *Industry Steering Committee* (ISC). In this committee, representatives of the aircraft and engine manufacturers, major suppliers as well as airlines and maintenance providers are convened. The MRB and ISC call together *Working Groups* (WG) which work out proposals on the maintenance requirements. At least three different WG for the fields of structure, systems and zonal dependencies are formed. Based on the proposals of the WG, the MRB works out a *Maintenance Review Board Report* (MRBR) in which the minimum scheduled maintenance requirements for the aircraft type and its power plants are approved. The MRBR it is not necessarily fixed for the whole life cycle as revisions are possible due to lessons learnt in operation [24, 38].

Airlines develop their individual maintenance programs in accordance with the MRBR [24]. The programs are often more strict than the regulations of the authorities to increase the robustness in case of disruptions [29].

The MRBR defines in detail which maintenance tasks have to be performed at which intervals. If an aircraft in service exceeds the intervals, it is grounded until the necessary maintenance tasks have been performed. A grounded aircraft at an airport which is not a maintenance base of the airline provokes high costs as external maintenance providers need to be used. Lufthansa Technik, for example, charges 275 euros per man-hour for non contracted customers. Additional costs incur for taxiing, materials and hangar space [52]. Unplanned groundings also lead to disruptions as flights may be delayed or cancelled. Therefore, airlines pay high attention that their aircraft operate within the allowed maintenance interval limits.

The interval limits are subject to the specific system or structural component. As most systems and components are exposed to a usage-dependent wear, the intervals are usually defined in terms of flight hours. For systems or components that are only stressed at specific events, the interval limits are defined in terms of flight cycles. For example, the landing gear and braking system are mainly stressed during landing. Other systems and components are exposed to a time-dependent wear. For example, the inflation pressure of the tires may change even when the aircraft is not used. The corresponding intervals are defined in terms of calendar time [15, 16, 17, 69].

The maintenance requirements are the most important constraints in tail assignment. We thus seek to map them as close to reality as possible in our model formulation. We can distinguish regular from irregular maintenance activities. While regular activities derive from the requirements of the authorities, irregular activities originate from unpredictable defects during operation. Irregular activities are always defined for a specific aircraft. An example of an irregular maintenance activity is the repair of a broken seat. Regular maintenance activities can be further divided in minor and major maintenance activities. Minor activities do not require long ground times but occur regularly in short intervals, e.g., Daily Check, A-Check. They are usually done at the ramp of the airport or overnight at a hangar [13]. Major activities are longer but occur seldom, e.g., C-Check, D-Check. They require special tools, spare parts, staff and hangar space.

Based on this classification, we define two categories of maintenance constraints (see Fig. 3.1). The first category applies for minor maintenance activities. We assume that minor maintenance activities are not fixed in time and not preassigned to aircraft. Their planning is the result of the tail assignment process. The tail assignment planner has to make sure to respect the interval limits between two consecutive checks, which can be defined in terms of accumulated flight time, flight cycles or calendar time. The second category of maintenance constraints applies for major and irregular maintenance activities. It is reasonable to assume that the location and time these activities take place are specified by a maintenance planning department. In tail assignment, they can be treated as preassigned activities. The

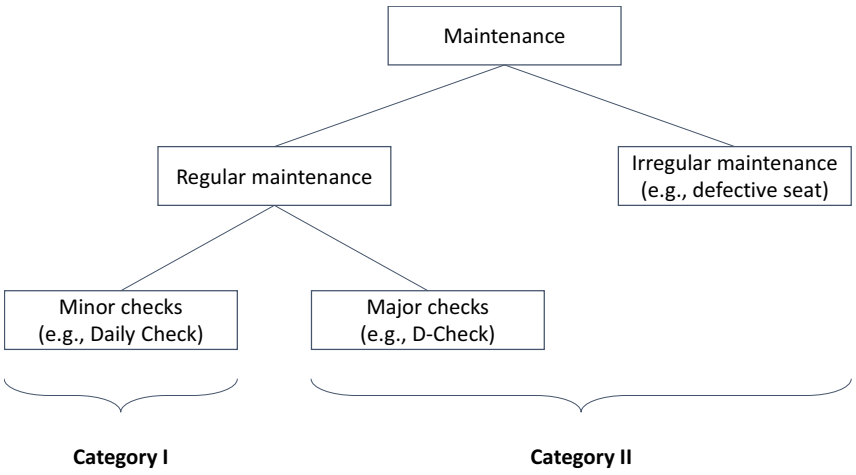


Figure 3.1: Classification of maintenance constraints

tail assignment planner solely has to make sure that the particular aircraft is on ground at the specified maintenance base during the specified time [29].

It should be noted that the second category of maintenance constraints only affects a small group of aircraft. Major maintenance checks occur very seldom and irregular maintenance stops should be an exception that applies only for a small percentage of the aircraft. We estimate that no more than 20 percent of the aircraft have prescheduled maintenance stops in a planning horizon of one week. The first category of constraints, in turn, affects all aircraft.

Some airlines require their aircraft to return to a maintenance base after a certain amount of days, regardless of how much flight time they have accumulated. In doing so, they create periodic maintenance possibilities for every aircraft. If a minor defect occurs, it might be possible to continue the planned operation of the aircraft until the next maintenance possibility without rerouting other aircraft [29]. In our formulation, these robustness constraints can be modeled the same way as regular minor maintenance activities.

The different types of regular maintenance activities are interrelated and cannot be planned individually. For example, it is not necessary to perform a Daily Check on days an A-Check is performed. We model the relation in terms of a check hierarchy. We assume that the check types can be explicitly ranked. A higher ranked check includes all lower ranked checks, which means a higher ranked check covers all tasks required in lower ranked checks plus additional tasks.

Maintenance activities cannot take place at every station, because certified staff and tools are required. The training of staff and purchasing of certified tools is expensive. Therefore, airlines concentrate their maintenance activities at a few maintenance bases [13]. Each base can only perform a limited number of checks at the same time, as maintenance resources, like hangar space and staff, are finite.

3.3.2 Other operational restrictions in tail assignment

Besides maintenance, there are other operational restrictions to consider in tail assignment. The most natural ones are the turn time constraints. The routes of the aircraft have to respect the minimum ground time that is required to perform the turnaround between two flights. During this time, the plane is disembarked, unloaded, cleaned, refueled, loaded, boarded and eventually de-iced. The turn time depends on fleet and airport and is typically 30 to 60 minutes. If the minimum ground time is not respected, the risk of disruptions rises [30].

Other common restrictions are so called curfews forbidding individual aircraft to perform certain activities. There can be numerous reasons for curfews. One reason is lacking aircraft equipment. For example, some aircraft in a fleet may be lacking an auxiliary power unit (APU), a small gas turbine located in the tail of the aircraft, which is needed to generate power during ground times. If an aircraft is not equipped with an APU, it is restricted from flying to airports where external ground power is unavailable. Aircraft without crew rest compartment are not allowed to perform long-distance flights on which two different crews are required. If a twin-engine fleet is used on flights where the distance to the next suitable airport exceeds certain limits, only aircraft fulfilling the *Extended-range Twin-engine Operational Performance Standards* (ETOPS) can be assigned.

Other reasons for curfews are minor technical issues. A prominent example is a defective reverse thrust [28]. This failure does not necessarily force an aircraft to immediately undergo repair. Instead, it is allowed to continue its operations, but restricted from landing at airports having too short runways. The tail assignment planner can try to route the aircraft so that restricted airports are avoided and the repair can be done in a cost- and time-effective manner during the next maintenance visit at an own base. Marketing aspects can also be reasons for curfews. If an aircraft is not equipped with an in-flight entertainment system, it is preferentially used on domestic routes and not on long international flights [29]. Even though some curfews apply for the entire fleet while others only apply for subfleets or individual aircraft, they all can be modeled as forbidding individual aircraft to perform certain activities.

Aircraft are sometimes preassigned to a list of activities. As stated above, pre-assigned activities are mainly irregular and major maintenance checks. But also flights can be preassigned to an aircraft. For example, Lufthansa used one of its new Boeing 747-8i equipped with a special livery to pick up the German national team from the World Cup 2014 in Brazil. Special treatment is also required for multi-leg flights. Multi-leg flights are flights consisting of several legs under a single flight number. The idea of multi-leg flights is that passengers do not need to change aircraft during a stop-over. These types of flights are especially important for hub-and-spoke operators. When smartly designed, they reduce the number of passengers having to change aircraft between two flights and thus increase passenger comfort. During tail assignment, the planner has to make sure that all legs of a multi-leg flight are assigned to the same aircraft.

Operational restrictions also arise due to interdependencies with other planning stages, e.g., crew scheduling. The routes of the aircraft have to be designed in a way that allows valid crew assignments. If the route of an aircraft includes flight sequences longer than the working hours limit without providing the possibility to swap crews in-between, it will be impossible to find valid crew assignments [21, 29]. As the scope of this thesis is focused on tail assignment, we currently do not take account of interdependency constraints with other planning stages.

Airlines sometimes require that all aircraft are exposed to an equal wear. In a cyclic formulation, this requirement can be achieved by a big-cycle constraint forcing all aircraft in a fleet to perform the same sequence of activities with a time shift. In a dated formulation, it is more difficult to incorporate such a constraint, because it involves the entire fleet and not a single tail. It can be approximated by requiring that the utilization of an aircraft must be within a specified interval of flying time relative to calendar time [29]. As such a formulation runs the risk to provoke infeasibilities, we do not incorporate equal utilization constraints in our model formulation.

The features and assumptions described in this chapter form the basis for our model formulation in Chap. 6. Before presenting our model, we first summarize existing research studies (Chap. 4) and provide a theoretical overview on our solution methodology (Chap. 5).

4 Literature review

The tail assignment problem of airlines is comparable to scheduling problems in other transportation sectors. Shipping companies have to assign vessels to their journeys [7, 60], railway companies schedule trains [19, 26, 36], and public transportation companies decide on busses to perform the offered trips [25, 46, 57]. While the general structures of these problems are similar to tail assignment, the objectives and restrictions are different. For bus scheduling, the focus is to minimize the number of required vehicles and deadhead trips while satisfying depot capacity. The critical bottlenecks in train scheduling are the limited number of platforms, tracks and overtaking possibilities. In seaborne shipping, the main challenge arises from the individual characteristics and cost structure of each vessel. None of these transportation sectors are confronted with such strict maintenance regulations as the airline industry. Therefore, the research is only partially transferable to the commercial aviation.

In this literature review, we summarize existing studies on the tail assignment problem and discuss the contributions of our work. We first provide an overview on modeling techniques and solution methodologies that are commonly used for solving this type of large-scale problem in the airline industry (Sec. 4.1). In practice, the assignment of aircraft to flights is performed in two stages: a tactical aircraft maintenance routing stage and an operational tail assignment stage[1]. In Sec. 4.2, we provide a comprehensive review of research papers on the aircraft maintenance routing problem. Existing research on the tail assignment problem is presented in Sec. 4.3. In Sec. 4.4, we review studies that seek to integrate the aircraft maintenance routing or tail assignment stage with other stages of the airline planning process. At the end of this chapter (Sec. 4.5), we summarize the benefits of our approach as opposed to the presented studies and outline our research contribution.

4.1 Modeling techniques and solution methodologies

Like many other airline planning problems, the aircraft maintenance routing and tail assignment problems are characterized by a large size and complexity. To be able to find solutions within a reasonable time frame, efficient formulations of the underlying

[1] It should be noted again that we use the term aircraft maintenance routing to refer to the problem of determining maintenance feasible routings for generic aircraft, while the tail assignment problem deals with specific aircraft.

mathematical models are required. At the same time, standard solution methodologies, like branch-and-bound, are often incapable of dealing with the characteristics of airline problems. As a consequence, more intelligent solution methodologies have to be adapted or newly invented. In this section, we give a résumé on used modeling techniques and solution methodologies for the aircraft maintenance routing and tail assignment problems in the literature.

4.1.1 Modeling techniques

A major challenge for all planning stages succeeding the flight scheduling is to map the constructed schedule in a mathematical model. Concerning the planning horizon, the existing research approaches can be split up in those assuming a finite and those assuming an infinite planning horizon. Approaches assuming a finite planning horizon are usually applied in a rolling horizon framework to reduce undesirable end-of-horizon effects. When an infinite planning horizon is used, researchers either assume that the same schedule is repeated every day or that a weekly schedule is repeated every week. While the weekly schedule is more realistic, especially for international operating airlines, it increases the complexity considerably [50]. In general, three different formulations are used to transform the flight schedule in a mathematical model:

- Time-space network
- Connection network
- Flight string formulation

The three formulations can be seen as a progression. In a time-space network each flight is treated individually, a connection network looks at pairs of flights and a flight string formulation considers sequences of flights. While the first two formulations originate from graph theory and model the flight schedule as a directed flow network, the flight string formulation results in a combinatorial set partitioning approach.

Time-space network

The time-space network is not only used in the context of aircraft maintenance routing and tail assignment, but also to model other airline planning problems, such as the fleet assignment and crew pairing. It was first proposed by Hane et al. to model the fleet assignment problem [31]. The schedule is modeled as a directed graph consisting of a set of vertices, called nodes, and a set of connecting edges, called arcs. Each destination in the schedule is represented by a time line. The time line of an airport consists of a series of nodes that denote either departure or arrival events. To correctly order the nodes on the time line, event dates are needed. For departure events, the departure time of the corresponding flight is used. For arrival events, the minimum turn time is added on the actual arrival time of the flight.

There are different types of arcs in the time-space network. Flight arcs connect the departure node with the arrival node of a flight. Ground arcs are required to allow aircraft to stay on ground for a certain period of time, e.g., waiting for the next departure [51, 64]. When an infinite planning horizon is assumed, additional wraparound arcs are required to connect the last with the first events in the planning horizon. When using a finite planning horizon, researchers usually implement dummy start events that represent the time and place an aircraft becomes available. Fig. 4.1 shows the time-space network for a schedule that consists of two aircraft, three airports and eight flights assuming a finite day-long planning horizon. We will use this schedule throughout this thesis to illustrate our proceedings.

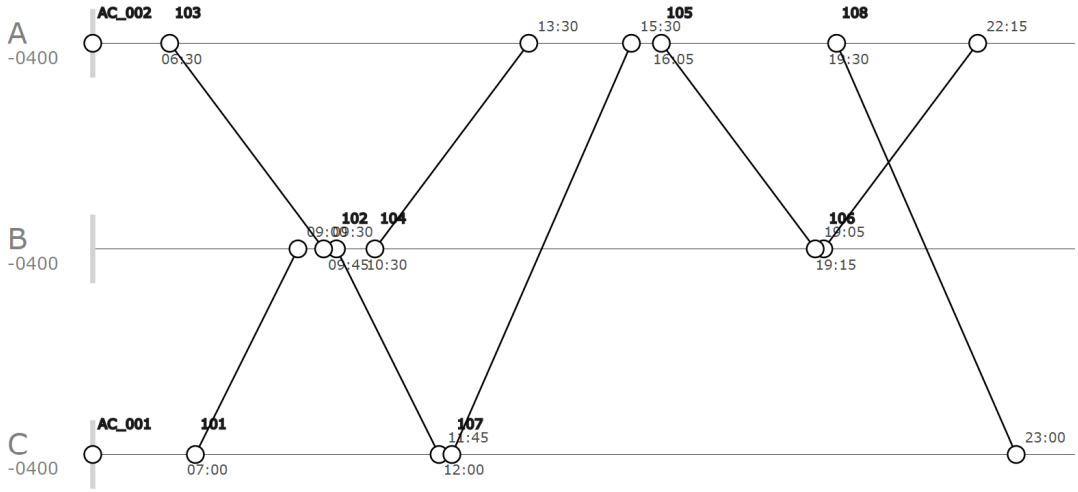


Figure 4.1: Example of a time-space network

Connection network

The connection network is also based on a directed graph. In contrast to a time-space network, the node set is used to denote the flights in the schedule. The arc set represents connections between two flights. Two flight nodes are connected by an arc if the departure airport of the succeeding flight equals the arrival airport of the preceding flight and the departure time of the succeeding flight is later than the arrival time of the preceding flight plus the minimum turn time [65]. Besides direct connection arcs, additional maintenance arcs are incorporated in the network. The prerequisite for a maintenance arc is that the airport at which the preceding flight arrives at and the succeeding flight departs from is a maintenance base of the airline, and that the available ground time is longer than the required time to perform maintenance [9]. If a finite planning horizon is used, additional source and sink nodes representing dummy start and end activities have to be implemented [32, 65]. The connection network is acyclic as only connections to future flights are possible [29]. Fig. 4.2 illustrates our example schedule using a connection network.

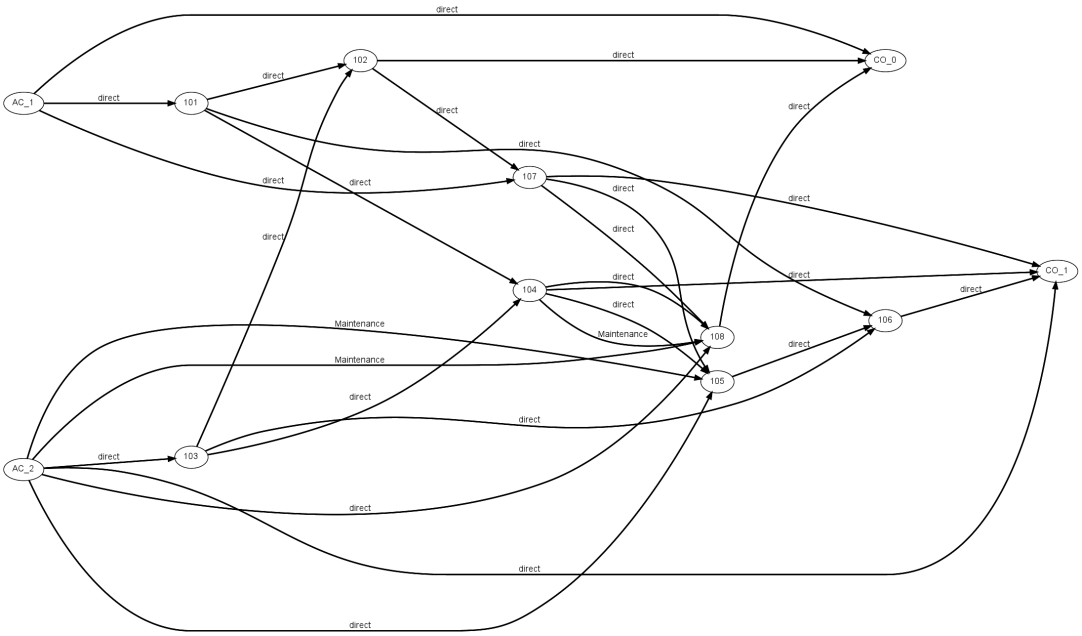


Figure 4.2: Example of a connection network

Flight string formulation

We find a third formulation technique to map the schedule of an airline in a mathematical model. In a flight string formulation, the problem is formulated in terms of flight routes, called flight strings. A flight string is defined as a sequence of flights, ground times and maintenance activities that are consecutively performed by an individual aircraft. A flight string satisfies flow balance, thus each flight in the sequence departs from the airport its predecessor arrived at. The sequence is feasible with respect to all operational requirements, e.g., maintenance standards of the aviation authorities [9, 56, 62]. A model based on the flight string formulation has a binary decision variable for each flight string through the schedule [29].

4.1.2 Solution methodologies

The second challenge is to develop a procedure to derive solutions from the model. For the aircraft maintenance routing and tail assignment problems, different solution methodologies are proposed in the literature. According to Klabjan they can be classified in five categories [45]:

- Branch-and-price (column generation)
- Benders decomposition
- Lagrangian relaxation
- Constraint programming
- Heuristic approaches

Branch-and-price (column generation)

Branch-and-price is a branch-and-bound algorithm where a linear programming (LP) relaxation is solved at every node of the branch-and-bound tree using column generation. Column generation is ideally suited when the constraint matrix has too many variables (columns) to handle efficiently. As the values of most variables equal zero in an optimal solution, the idea is to only consider a subset of columns in every iteration. The problem consisting of the chosen subset of columns is called the restricted master problem (RMP). At the start of each iteration, the LP relaxation of the RMP is solved and a set of optimal dual values determined. Based on the dual values, the reduced cost of nonbasic variables can be assessed. A subproblem, called pricing problem, is created to check the optimality of the obtained solution. The pricing problem seeks to identify nonbasic columns which may improve the solution. In a minimization problem, promising columns are columns with negative reduced cost. In case promising columns are identified they are added to the RMP and the next iteration begins. Appending columns to the RMP is referred to as column generation. Similarly, columns with large reduced cost are removed from the RMP when the size of the constraints matrix becomes too large. The repeated iterations are stopped and an optimal solution is found when the pricing problem cannot identify columns with negative reduced cost [9, 10, 45].

The performance of column generation largely depends on the pricing problem. It should be able to identify columns with negative reduced cost without examining all variables [28]. Therefore, it must be tailored to the problem under consideration.

Benders decomposition

Benders decomposition follows a strategy of "learning from one's mistakes" [39]. The original problem is decomposed in a master problem and a subproblem which are solved iteratively. By solving the master problem, trial values for a subset of decision variables are generated. The subproblem consists of finding the optimal solution to the residual problem after fixing the trial values.

Traditionally, both problems are formulated as a linear programming problem. The master problem includes a single continuous variable that provides a bound on the optimal solution. When infeasibilities occur, the dual vector to the subproblem is used to define a Benders cut which is added to the master problem before it is resolved. The added cuts restrict the assignment of trial values for the fixed variables in the next iterations. Finally, only optimal values remain and the algorithm terminates after enumerating only a few possible values for the fixed variables [39, 45].

Logic-based Benders decomposition can be seen as a generalization of the classical Benders decomposition. It extends the generation of Benders cuts to an arbitrary class of subproblems. Instead of using the linear programming dual, an inference dual is derived from the constraint set. The inference dual is a logical formulated proof of optimality which provides a bound on the optimal value [39].

Lagrangian relaxation

Lagrangian relaxation works under the assumption that the constraint matrix can be subdivided into "easy" and "difficult" constraints. The difficult constraints are accountable for raising complexity. When they are removed, the problem is solvable without difficulty. Lagrangian relaxation takes advantage of this observation. The difficult constraints are removed from the set of constraints and instead added to the objective function. A linear positive Lagrangian multiplier is associated with every constraint that has been moved to the objective function to impose a penalty for violating it. The resulting problem is called Lagrangian relaxation and is less challenging to solve than the original problem. The solution of the Lagrangian relaxation is not necessarily feasible for the original problem, but it can be used to formulate a bound on its optimal solution. The problem of finding the set of Lagrangian multipliers which yields the tightest bound is referred to as Lagrangian multiplier problem. The Lagrangian multiplier problem is a nonlinear optimization problem, which is usually solved by variants of the subgradient algorithm [5, 37, 45].

A major advantage of Lagrangian relaxation is that the generated bounds are much tighter than those provided by a LP relaxation. Also, it requires minor implementation effort and is able to handle complex constraints. The main drawback is that the approach does not guarantee to find feasible solutions, as they have to be created heuristically in the subgradient algorithm [5, 37, 45].

Constraint programming

In the recent years, constraint programming found its way into solving large-scale planning problems in the airline industry. Constraint programming is a form of declarative programming that was developed in the mid-1980s. In contrast to standard mathematical programming methods, the focus of the analysis is not the objective function, but the variables, their domain and the constraints. As constraint programming mainly addresses feasibility, rather than optimality, it is suitable when the user is interested in finding any solution to a problem [28, 73].

Heuristic approaches

Heuristic approaches are preferentially used when the problem is too complicated to be solved exactly within a reasonable time frame. As opposed to exact approaches, heuristics do not guarantee optimality. The goal of using heuristics is to find good suboptimal solutions while applying a reasonable computational effort. Heuristics are often based on intuitive ideas of how to search for a good solution. Many approaches employ some sort of priority rule to obtain initial solutions, e.g., greediness, or seek to iteratively improve existing solutions by screening their neighborhood. The main drawback of these local search methods is the risk to get trapped

in local optima, which they are unable to escape from. Also, heuristics are usually problem-dependent and need to be tailored specifically to the problem under consideration. In recent years, the development of meta-heuristics allows to circumvent these shortcomings [37, 73, 77].

A meta-heuristic is an adaptive approach that provides a general framework in which problem-specific heuristics and exact approaches can be integrated. It can be described as an "iterative master process that guides and modifies the operations of subordinate methods to efficiently produce high quality solutions" [72]. In contrast to standard heuristics, a meta-heuristic does not stop when reaching a local optimum. Instead it guides and modifies the subordinate methods to continue the search. The basic concepts of meta-heuristics originate from psychology, biology, physics and neurology. Among the most prominent meta-heuristics are: large neighborhood search, simulated annealing, tabu search, evolutionary algorithms and swarm algorithms [73]. Math-heuristics introduce mathematical programming methods within a meta-heuristic framework. An essential feature is the utilization of mathematical programming techniques in some part of the algorithm [37, 73, 77].

4.2 Aircraft maintenance routing literature

The aircraft maintenance routing problem has received much academic attention in the past years. The studies mostly deal with cyclic problems and ignore initial positions and conditions of the aircraft.

Liang et al. [51] propose a mixed-integer programming (MIP) formulation for the daily aircraft maintenance routing problem. They create a compact representation of the time-space network which by its structure forces the aircraft to spend a night at a maintenance base after a maximum number of operating days. Instead of wraparound arcs, the authors introduce capacitated maintenance arcs at maintenance stations. The objective function maximizes through values and penalizes undesirable short connections. The computational results show that the formulation performs very well and optimal solutions can be found in a few seconds, even for instances with up to 70 aircraft and 350 daily flights. A deficit is the assumption that the maintenance requirements can be represented by a single check that only depends on the number of operating days. Furthermore, they assume that maintenance is only performed at night. This assumption might be valid for most carriers operating a domestic flight schedule, as those flights are usually performed during daytime. For international operating carriers, this assumption seldom applies.

Later, Liang and Chaovalitwongse [50] extend the analysis to a weekly schedule. To do that, they further adjust the time-space network. The computational results are again noticeable. For test scenarios with up to 4000 weekly flights and more than 250 aircraft, the problem can be solved in less than 150 seconds. The points of critic are similar to those mentioned above.

Haouari et al. [33] propose a connection formulation for the daily aircraft maintenance routing problem. They assume that the maintenance requirements depend on three criteria: the accumulated number of flight hours and flight cycles as well as the calendar time since the last check. However, they only consider a single check type. The compact polynomial-sized formulation avoids the need for complicated algorithmic implementations. Test cases of up to 350 daily flights and 140 aircraft can be solved to optimality in less than ten seconds. A weakness of the proposed formulation is the arc definition. Two flights are connected by a single arc type. Haouari et al. assume that the aircraft undergo maintenance whenever it is possible. This formulation is valid when we look at maintenance opportunities, but not when we want to minimize maintenance effort. Due to the exclusive arc definition, the amount of created arcs remains relatively small. By allowing multiple arc types to connect the same two flights, e.g., multiple maintenance arcs and direct connection arcs, the number of arcs would increase drastically.

Barnhart et al. [9] discuss a flight string formulation for the daily aircraft maintenance routing problem. In their model, they implement a big-cycle constraint ensuring an equal utilization of all aircraft. As solution approach, the authors design a branch-and-price framework based on an alternating column generation and constraint generation algorithm. They solve test scenarios with up to 190 daily flights. Their algorithm performs quite well on small instances with less than 80 daily flights. For larger instances, the solving time rises significantly. This is due to the long time requirement for generating maintenance feasible flight strings, since their number increases exponentially with the problem size.

A different approach is taken by Clarke et al. [20]. They model the daily aircraft maintenance routing problem as an asymmetric traveling salesman problem with side constraints. As solution approach, they use Lagrangian relaxation and subgradient optimization. Clarke et al. maximize through values while respecting a big-cycle constraint and the requirements of two maintenance types that both depend on the accumulated number of calendar days. They provide results of eleven test instances, but do not mention how large the instances are.

An early approach to optimize aircraft routings is presented by Kabbani and Patty [44]. They propose a set-partitioning model to create maintenance feasible routings at American Airlines. For a route to be maintenance feasible, it has to incorporate an overnight stay at a maintenance base every three days. The authors develop a two-step solution approach. Since maintenance is only done overnight, they first determine over-the-day routings. In the second step, they connect the daily routings to develop tours that satisfy the three-day maintenance requirement.

The same two-stage process is taken on by Gopalan and Talluri [27]. They propose a polynomial-time algorithm to create routes when maintenance must be performed overnight after three days of flying. Besides, the three-day maintenance check, they also consider a less frequent balance check. Since the two-stage process of fixing over-the-day routings before creating maintenance feasible tours can easily

lead to infeasibilities, they develop an iterative framework. If infeasibilities occur, the authors use heuristics to readjust the over-the-day routings and start a new iteration. Gopalan and Talluri present two different models. First, they analyze a model based on an infinite planning horizon in which the same over-the-day routings are repeated every day. Second, they analyze a model based on a finite planning horizon in which they permit varying daily routings.

Talluri [70] extends the analysis to longer maintenance intervals. He assumes that the same over-the-day routings are repeated over an infinite planning horizon. The author shows that the problem is NP-complete for any maintenance interval longer than three days, but provides a polynomial-time algorithm for the special situation of a single maintenance station and a four-day maintenance interval. He also shows that the four-day routing problem can be solved in polynomial time for multiple maintenance stations when neglecting the balance check requirement.

4.3 Tail assignment literature

Compared to aircraft maintenance routing, the research on tail assignment is limited and mostly recent [12, 62]. The studies at this level deal with dated problems that explicitly take into consideration the initial state of the aircraft. Operational restrictions are more detailed than in the aircraft maintenance routing stage, because the solutions serve as plans to be followed in real operation [62].

Grönkvist [29] develops a connection network and a flight string formulation for the tail assignment problem. He assumes a finite planning horizon of several weeks. As constraints, he considers detailed maintenance restrictions, preassigned activities and aircraft curfews. He argues that the string-based model is more suitable to incorporate the complicated maintenance constraints and thus focuses his research on it. To solve the problem, he proposes a combined column generation and constraint programming approach. Instead of embedding the column generator in a branch-and-price framework, he proposes a local search heuristic. Constraint programming is mainly used during preprocessing to reduce problem sizes and to generate initial solutions to start from.

A tail assignment approach for the short-term planning of an airline is presented by Sarac et al. [65]. They aim to create routings such that all flights over a one-day planning horizon are covered and aircraft needing maintenance end their route at an appropriate maintenance station in the evening. Maintenance is performed in the following night and is subject to available capacity. Sarac et al. propose a set-partitioning model, which they expect to be solved every day. As solution approach, they propose a branch-and-price algorithm. The authors present computational results of ten randomly generated test instances, which are rather small-sized. They assume that maintenance activities require six to eight hours and exclusively depend on the accumulated flight hours. In their study, they neglect the impact of their short-term route changes on crew planning.

Afsar et al. [2] propose a two-step heuristic based on a longest path method that maximizes aircraft utilization before maintenance checks and balances flight loads of the aircraft. In the first step, they focus on critical aircraft which are in need of maintenance during the next week. In the second step, they seek to ensure that all remaining flights are assigned to the noncritical aircraft. Their approach is applied in a rolling horizon framework, solving one week planning horizons at a time. A shortcoming is that only one type of check depending on the accumulated flight hours is considered. Their heuristic allows flights to be left unassigned, which unfortunately appears to happen quite often in the final solutions.

In a following study, the same authors apply different heuristics, including a simulated annealing approach, to the identical problem [3]. In both studies, all maintenance checks are prescheduled.

A heuristic for a similar problem is proposed by Basdere and Bilge [12]. They aim to maximize the utilization of the remaining flying time of the aircraft before a maintenance check. The approach is embedded in a rolling horizon framework with a weekly planning period. The underlying model is based on a connection network and takes into account maintenance capacity considerations. The authors propose a heuristic solution approach based on compressed annealing. For benchmarking, they compare the performance with an exact branch-and-bound approach on a small-sized instance. A shortcoming in their formulation is that at most one maintenance activity is planned for each tail in the planning horizon. Besides, they assume that predefined maintenance slots need to be used, but not necessarily by the preassigned aircraft. This assumption is rather unrealistic, as aircraft specific parts need to be provided to perform certain maintenance.

Argüello et al. [8] focus their research on reoptimizing the tail assignment in case of disruptions. The objective is to minimize flight cancellation and delay costs associated with the schedule recovery. They propose a greedy randomized adaptive search procedure (GRASP) to generate feasible routings. GRASP is based on a local search method that iteratively generates new solutions and randomly explores their neighborhood. To generate new solutions a greedy approach is used.

Lapp and Wikenhauser [49] reformulate the tail assignment problem to consider fuel consumption and emissions. Due to different engine equipment, modifications and age, aircraft of the same fleet have different fuel efficiencies. The goal in this study is to assign more efficient aircraft to fuel-intensive flights. The authors assume that an initial routing plan for the planning period already exists. However, they neglect any maintenance considerations. They only try to reduce distortions by limiting the number of aircraft and route changes compared to the initial plan. Also, the result may lead to a severe scattering of aircraft utilization as more efficient aircraft tend to be used more.

4.4 Integration with other planning stages

In recent years, the integrated analysis of different planning stages has attracted more and more academic interest. The aircraft maintenance routing stage is preferentially integrated with the fleet assignment stage [9, 32, 34, 50]. One of the earliest attempts to integrate these two planning stages has been made by Barnhart et al. [9]. The authors introduce a string-based model which is solved using a branch-and-price framework. They consider a single maintenance check which is due after four days. They were able to solve a weekly schedule consisting of nine fleets, 89 aircraft and 1124 flights in five and a half hours.

Liang and Chaovalitwongse [50] examine the same problem. They propose a model based on an adjusted time-space network and present computational results for eight different test cases of four and eight fleets with up to 2000 weekly flights and more than 100 aircraft.

Haouari et al. [32] present heuristic approaches to solve the integrated fleet assignment and aircraft maintenance routing problem based on a connection network. In addition, they develop two exact approaches to solve the same problem [34]. The first approach is based on column generation, the second on Benders decomposition. In their analysis, the authors conclude that column generation is more capable of finding optimal solutions, the Benders approach of quickly finding good solutions.

A different integration is proposed by Cohn and Barnhart [21]. They seek to partially integrate the crew pairing and aircraft maintenance routing stage by incorporating key maintenance routing decisions in an extended crew pairing model. The authors propose a string-based model embedded in a branch-and-price approach. As this approach might require long solving times, they also propose an alternative idea. There, they start with a crew pairing problem in which the maintenance constraints are neglected. If the obtained solution is maintenance infeasible, cuts excluding the current solution are added to the problem before it is resolved.

An attempt to integrate the entire tactical planning of an airline is proposed by Papadakos [56]. He uses a flight string formulation for solving the fleet assignment, aircraft maintenance routing and crew pairing problem simultaneously. As solution approach, he combines an enhanced Benders decomposition method with accelerated column generation.

Cordeau et al. [22] focus their research on integrating the tail and crew assignment stage. They solve a dated set-partitioning model using a Benders decomposition approach. Tail assignment is handled in the master problem and crew assignment in the subproblem. Both problems are solved using column generation. A heuristic branch-and-bound method is used to compute integer solutions. They present results to nine test instances with one to three days planning periods and up to 525 flights, 35 aircraft and 67 crews. Deficits are the solving times of up to five hours which are too long for an operational implementation.

Another approach for the integrated tail and crew assignment is presented by Ruther et al. [62]. They propose a planning approach in which the aircrafts' routes and crew assignments are reoptimized close to the day of operation, allowing a late re-fleeting. Their model is based on a flight string formulation. As solution approach, Ruther et al. propose a branch-and-price framework with a pricing problem for each aircraft and group of crews having the same working period and work base. A shortcoming is that crews would only be noticed a few days in advance when and where they will fly. Before the assignment, they would only be informed about their periods of operation. This is hardly realizable at airlines with powerful labor unions.

4.5 Contribution

The literature review shows that the tail assignment of airlines is not well optimized yet. Most related studies focus on the tactical aircraft maintenance routing instead of the operational tail assignment stage [12, 62]. The majority of studies is unable to capture the complicated maintenance requirements. It is common to translate the detailed restrictions in simply requiring the aircraft to overnight at a maintenance base after a certain amount of days. Also, most studies assume a single check type, neglecting the hierarchical interdependencies between different checks in reality. Our approach is distinctive as it maps maintenance restrictions very close to reality. Besides, we incorporate many other operational restrictions. Only few other studies model the tail assignment problem as close to reality as we do, e.g., Grönkvist [29].

We present a model based on a connection network. In contrast to flight string formulations, which imply exponentially many variables, our formulation makes do with quadratically many variables. Our objective is not only the maximization of through values or the minimization of undesired connections. We also aim to minimize the occurring maintenance effort while maximizing aircraft utilization. Maximizing aircraft utilization decreases the loss of flight potential, which can be defined as the unused flight time, flight cycles and calendar time before a check. We therefore contribute to reduce operational costs as unnecessary maintenance checks are avoided. A weekly planning horizon applied in a rolling horizon framework appears to be well balanced. On one hand, it is long enough to maximize aircraft utilization before the next check and provide operational benefits for maintenance and crew planning. On the other hand, it guarantees sufficient data quality and avoids simplifying assumptions regarding maintenance needs.

Many publications can be found that successfully apply LNS to a wide variety of optimization problems [1, 14, 35, 48, 54, 61]. However, the application to the tail assignment problem is yet unexplored. The study of Grönkvist [29] leads in a similar direction, but his research is mainly focused on column generation and constraint programming. Our results show that LNS is capable of generating high quality solutions in acceptable time frames making it ideally suited for the dynamic environment of the airline industry.

5 Large neighborhood search

The aim of this work is to develop a large neighborhood search (LNS) algorithm for the tail assignment problem of airlines. In this chapter, we provide a theoretical foundation of the chosen solution methodology. We first give a brief introduction to the LNS method (Sec. 5.1) and outline the process to derive a problem solution (Sec. 5.2). In Sec. 5.3, we describe the enhanced adaptive large neighborhood search (ALNS) framework which we seek to apply in our research. We conclude this chapter (Sec. 5.4) by justifying our choice of solution methodology.

5.1 Overview

The LNS method is a meta-heuristic, which was first proposed by Shaw in the context of vehicle routing [67]. The idea is to gradually improve an initial solution by alternately destroying and repairing different parts of the solution [14, 58]. It is similar to the ruin and recreate principle introduced by Schrimpf et al. [66]. An alternative view on LNS is to see it as a sequence of fix-and-optimize iterations. In the beginning of each iteration, the values of some variables in the current solution are fixed. In the succeeding optimization step the algorithm seeks to reoptimize the problem while taking into account the values of the fixed variables [61].

LNS belongs to the class of local search methods which explore the neighborhood of a solution for improving alternatives. Traditionally, local search methods are defined by exploring a very limited neighborhood in every iteration. In doing so, a huge number of solutions is analyzed in a short time. However, there are only minor changes of a solution in every iteration and the algorithms sometimes show difficulties exploring different promising areas of the solution space. Enlarging the searched neighborhood is favorable to facilitate moving around the solution space. LNS provides a framework that supports exploring a large neighborhood in an efficient manner [4, 61].

Solution algorithms based on LNS have recently become very popular. They are used in various transportation and scheduling problems and persuade with an outstanding ratio of solution quality to time effort [58]. Researchers report that LNS has excellent capabilities, especially when solving large, complex problems [66]. Generic solvers are often incapable of handling these problems explicitly in the aggregate. The LNS procedure avoids this obstacle as only a small-sized decomposed problem is investigated in each iteration.

5.2 Process of a large neighborhood search

The LNS process is divided into two stages:

1. Initial solution stage
2. Improvement stage

The structure of the LNS process is illustrated in Fig. 5.1. Starting point of every LNS algorithm is the construction of an initial solution. The subsequent improvement stage consists of three steps that are iteratively repeated. In the first step, the incumbent solution is partially destroyed. Afterwards, a repair procedure generates a new solution for the decomposed problem. In the last step, a decision tool analyzes the newly obtained solution. Based on decision rules, the new solution is either accepted to become the new incumbent or rejected. The iteration sequence is repeated until a stopping criterion, e.g., an upper time limit, is met. The final result is the best solution that has been found during the entire search.

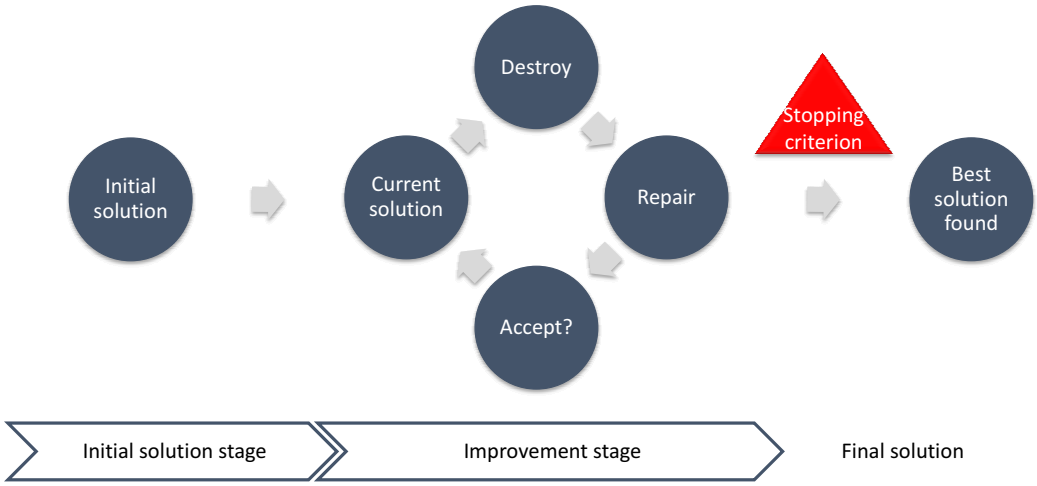


Figure 5.1: Process of a large neighborhood search

5.2.1 Initial solution stage

All LNS algorithms share the characteristic that they require an initial solution to start the destroy-and-repair iterations in the improvement stage. The initial solution must be feasible regarding all operational and functional constraints of the underlying problem [14]. However, it must not be notably good in terms of objective value. The required time to construct the initial solution should be low to allow a quick changeover to the subsequent improvement stage.

For tractable problems, finding a feasible initial solution does not require much computational effort. Intractable problems, on the other hand, are computationally hard to solve, even when only looking for a feasible solution.

5.2.2 Improvement stage

As stated above, the iterations in the improvement stage consists of three steps. In this section, we take a detailed look at each step.

Step 1: Destructor method

In the first step, the incumbent solution is partially destroyed by using a destructor method. This means that the values of some variables are released while other variables are fixed at their current values. The ratio of released variables defines the size of the searched neighborhood. When all variables are fixed, then no search is performed. When all variables are released from their current values, the entire problem is subject for resolving [61]. On one hand, the larger the neighborhood, the higher is the quality of the locally optimal solutions. On the other hand, severe destruction results in longer solving times to explore the neighborhood. Shaw proposes an approach in which the degree of destruction is gradually increased. He argues that LNS works best when in each iteration only the minimum set of assignments of the current solution is released that yields an improvement [67]. Ropke and Pisinger, in contrast, randomly vary the degree of destruction in each iteration [61]. The destructor method should also incorporate some randomness to assure that different parts of the solution are destroyed in each iteration [58].

Step 2: Repair method

The implemented repair method affects the efficiency of the LNS algorithm as well. The repair method can be designed as to find the optimal solution to the decomposed problem. In case a mathematical programming technique is used to search the neighborhood for improving solutions, LNS evolves to a math-heuristic. Alternatively, heuristic methods may be implemented, which yield faster but not necessarily as good solutions as exact methods. Pisinger and Ropke favor heuristic repair methods. They argue that exact procedures only lead to improving or identical solutions and therefore are at risk to get stuck in local optima [58]. The repair method can be hand-coded or implemented by use of a solver tool.

Step 3: Acceptance rule

The acceptance rule determines if a new solution is accepted as new incumbent or rejected. If the new solution is rejected, the next iteration starts from the previous incumbent solution. To avoid getting stuck in local optima, it might be unfavorable to only accept improved solutions. By allowing some deterioration in iterations, better solutions might be generated in the long run. Schrimpf et al. propose five different acceptance rules [66]. The *random walk rule* accepts every new solution.

The *greedy rule* only accepts new solutions which are better than the incumbent solution. If a *simulated annealing rule* is implemented, improving solutions are accepted, but also, with some probability, worse solutions. The probability that a worse solution is accepted is high in the beginning and decreases gradually with increasing number of iterations [58]. The *threshold rule* accepts new solutions which are not worse than a certain threshold. The threshold is defined in reference to the incumbent solution, e.g., a maximum deterioration of ten percent in objective value. Finally, the *great deluge rule* rejects new solutions below a certain quality level, e.g., all solutions with an objective value of more than ten.

Stopping criterion

The three-step iterative procedure is repeated until a stopping criterion is met. Different definitions of an appropriate stopping criterion can be found in the literature. Ropke and Pisinger propose to stop after a specified number of iterations [61]. Alternatively, a time limit can be set as stopping criterion [14]. Dueck recommends to stop if the solution quality has not improved for a long time [23].

5.3 Adaptive large neighborhood search

The adaptive large neighborhood search (ALNS) heuristic is an advancement of the LNS method. It was first proposed by Ropke and Pisinger in the context of pickup and delivery problems [61]. In the standard LNS method, a single destructor and repair method is chosen to be used throughout all iterations of the search. However, it is unknown in advance which methods are best suited for the instance under consideration. It is also imaginable that the performance of a method varies during the search. For example, some methods might be well suited in the first iterations of the search while others lead to better results in later iterations. Ropke and Pisinger argue that the overall robustness of the search can be increased by alternating between different destructor and repair methods [61].

ALNS provides a framework that allows multiple methods to be used within the same search. The destructor and repair methods are selected randomly in each iteration. The probability that a particular method is selected is based on its performance during the past iterations. At fixed points during the search the probabilities are readjusted. The ALNS heuristic thus possesses a certain intelligence as promising methods are selected more often. The dynamic readjustment takes into account that the suitability of a method may change during the progress of the search. The ALNS heuristic enables the search to autonomously "adapt to the instance at hand and to the state of the search" [58].

An important element in the design of an ALNS algorithm is the selection procedure for the destructor and repair methods used in the next iteration. Ropke and

Pisinger propose a roulette-wheel selection principle [61]. Given n competing methods with weights w_i, the probability that method x is selected in the next iteration is:

$$P_x = \frac{w_x}{\sum_{n \epsilon N} w_n} \tag{5.1}$$

Another critical decision is how and at what time to readjust the probabilities. In order to consider a changing suitability of methods during the search, Ropke and Pisinger divide the search into segments. A segment is defined as a fixed number of iterations. At the beginning of the search, all methods have an equal weight. When reaching the end of a segment, the weights are readjusted. Ropke and Pisinger introduce a score variable to track the performance of a method. The score represents how well the method has performed in the recent iterations. Successful methods are characterized by a higher score. At the beginning of each segment the score variables are set to zero. In each iteration a combination of the destructor and repair methods is randomly selected. As it is impossible to state whether the destructor or the repair method is accountable for a success, the scores of both methods are updated equally at the end of the iteration. Depending on the quality of the new found solution, the scores of the used destructor and repair methods are increased by the parameters $\sigma_i (i = 1, 2, 3, \sigma_1 > \sigma_2 > \sigma_3)$. If the new found solution is a new global best solution, both methods are rewarded with a high σ_1. If the new solution is better than the incumbent solution and has not been visited before, σ_2 is added to the scores of the corresponding methods. Ropke and Pisinger also reward non-improving solutions that have not been visited before. In this case, the small reward σ_3 is added to the scores of the methods. It is interesting to note that Ropke and Pisinger only reward methods that find unvisited solutions. In doing so, they want to increase the utilization of methods that diversify the search. At the end of a segment, the new weights are calculated using formula 5.2. Let w_x^* denote the new weight of method x. π_x denotes the final score of method x achieved in the last segment. Θ_x represents the number of iterations during the last segment in which method x was selected [61].

$$w_x^* = w_x(1 - r) + r\frac{\pi_x}{\theta_x} \tag{5.2}$$

The reaction factor $r \in [0, 1]$ determines how quickly a change in performance is applied to the weights. The higher r the more the weights are based on the recent performance. If r is set to zero, the recent performance is ignored and the previous weights are used again during the next segment [61].

5.4 Benefits compared to other solution approaches

There are several reasons why we choose LNS as our solution methodology. The LNS framework is an intuitive concept that can be implemented quickly without much effort. At the same time, LNS provides a large freedom of design in configuration. The three steps of the improvement stage, the stopping criterion as well as the ALNS structure can be designed to any complexity. The efficiency of LNS largely depends on the design of these elements. Compared to traditional local search methods, the risk of getting stuck in local optima is minimized, because a large neighborhood is explored in every iteration. Another benefit is that the solution methodology can be tuned according to the user preferences. LNS is capable of providing solutions of different quality levels depending on the provided time frame. The user himself can choose if he wants an acceptable solution quickly, or if he is willing to wait longer for a better solution.

In contrast to other algorithms, LNS does not focus on optimality, but instead on satisficing. It is promising to find very good solutions for the tail assignment problem in an acceptable time frame.

6 Mathematical model

In this chapter, we present a comprehensive mathematical model for the tail assignment problem. We will refer to this model as *main model,* because it provides a formal description of the detailed problem. The main model is independent of the chosen solution methodology. In our approach, it will serve as basis for our ALNS math-heuristic.

6.1 Choice of modeling technique

In Chap. 4, we provided an overview on modeling techniques for the tail assignment problem. We identified three different formulations that are used in existing studies, namely time-space networks, connection networks and flight string formulation. Our main model is based on a connection network. In this section, we analyze the advantages and disadvantages of the three formulations and justify our choice.

The main advantage of time-space networks is that their size increases linearly with the size of the schedule [50]. Models based on a time-space network are very compact and reported to be solvable in reasonable time frames, even for large instances. A major shortcoming is the limited possibility to incorporate operational restrictions, such as maintenance requirements, in the model formulation. Since the connections between two flights are not explicitly defined, we can solely incorporate restrictions that are independent of the flight route. As described in Sec. 3.3, the maintenance intervals of an aircraft depend on three different criteria, i.e., flight hours, flight cycles and calendar time. Out of these criteria only the calendar time is independent of the flight route. It is also challenging to model multiple maintenance types and their complex interdependencies within a time-space model formulation.

Researchers favoring time-space networks usually handle these drawbacks by simplifying and aggregating the detailed maintenance restrictions. They translate them in requiring an aircraft to spend a night at a maintenance base after a certain amount of days [50, 51]. They argue that this approach corresponds with the maintenance planning of most airlines. However, if the allowed number of days between two maintenance nights is too high, it is not ensured that all maintenance limits are met. If the number is too low, the full potential for improvement may not be tapped [9].

Connection networks allow richer modeling possibilities than time-space networks. Since the connections between two flights are explicitly defined by the arcs of the

network, we can incorporate detailed operational restrictions in the model. We can also assign a cost parameter to the arcs to express the benefit of flying two flights consecutively. Forbidden connections can be modeled by either not including the corresponding arc in the networks or by assigning a very high cost parameter. But, the advantages come at the price of larger network sizes. The number of arcs increases quadratically with the number of flights in the schedule [9]. The complicated resource constraints to model the maintenance requirements are another disadvantage of the connection formulation, because they raise the complexity and destroy the flow structure of the network flow model [29].

String-based models claim to be more flexible and expressive than models based on the other two formulation techniques. Nonlinear, complex cost functions and constraints can be incorporated without causing too much difficulties [9]. The resource constraints to model the maintenance requirements are avoided, because only operational feasible flight strings are generated [29]. The main drawback of string-based models is that even small-sized schedules dispose of an extremely large number of flight strings. Extensive preprocessing is required, because the number of flight strings increases exponentially with the number of flights in the schedule [9].

String-based models require complex solution methodologies and are mainly used in combination with a branch-and-price algorithm [10, 33]. It is usually inefficient to generate all flight strings in advance, since their number is too large to enumerate explicitly. In the beginning, researchers only define a subset of flight strings. During the solving procedure, their algorithm specifically creates additional flight strings which are promising to improve the solution. The creation of promising flight strings (the pricing problem) is especially challenging. The procedure must be tailored to exploit the problem structure without examining all possible flight strings [9, 33].

The goal in this research is to build a comprehensive mathematical model that considers detailed maintenance requirements and all other operational regulations. The limited modeling possibilities of time-space networks contradict this goal. Connection networks and string-based formulations, on the other hand, are capable of incorporating complex operational restrictions and are both acceptable for our modeling needs. As we seek to create an intuitive and easy solution algorithm, we want to avoid complex implementation procedures and unnecessary large numbers of variables. Therefore, a connection formulation is more suitable. The larger model sizes and longer solving times compared to a time-space formulation are not relevant for our approach, as we only intend to optimize a small decomposed part of the main model in each iteration of our LNS procedure.

6.2 Main model

Tail assignment is the process of defining operational feasible routes for a set of aircraft, covering all activities in a schedule, while minimizing some cost function [29].

We present a model that allows several fleets to be solved simultaneously. The model captures various operational restrictions as well as the requirements of multiple interdependent maintenance types. The objective is to minimize the total occurring maintenance effort. In addition, we consider optional cost terms for assigning two consecutive flights to a specific tail. We propose a dated problem formulation for a finite planning horizon and allow the schedule to be changed from day to day.

Our main model is based on the work of Haouari et al. [33] and Grönkvist [29]. We extend and adjust their ideas to fit our purposes. Even though there are some similarities, there are also fundamental differences. Haouari et al. deal with the aircraft maintenance routing problem. They seek to solve cyclic problems on an infinite planning horizon considering a single maintenance type. Both, Haouari et al. and Grönkvist, propose an exclusive arc definition that forces the aircraft to perform maintenance whenever possible. In doing so, they focus on maintenance opportunities and not on real maintenance activities. An exclusive arc definition contradicts our objective and needs to be adjusted in our model. Haouari et al. use an objective function that maximizes through revenues and penalizes short connections. Similarly, Grönkvist seeks to minimize the costs associated with assigning two consecutive flights to a specific tail. Our objective, in contrast, is more flexible and holistic.

Index sets: Let T be the set of aircraft (tails) and A the set of activities. A also includes special source and sink activities that represent the start and end of the routes. We will refer to them as carry-in (CI) and carry-out (CO). These dummy activities are required for modeling and algorithmic purposes [65]. For every aircraft t, we define a set of preassigned activities P_t and a set of forbidden activities F_t.

The different types of maintenance are compiled in set M. Note that M only consists of frequent and regular types of maintenance, e.g., Daily Check. We model less frequent and irregular types, e.g., D-Check, as prescheduled activities that are preassigned to a particular tail. The criteria in terms of which the interval limit of maintenance m is defined are compiled in set Q_m. In general, maintenance interval limits can be defined in terms of the accumulated flight time, flight cycles and calendar time. The set R contains all possible connection types of two activities. Activities can be connected directly or via a maintenance check of type m.

Finally, let O be the set of connection arcs. As we intend to solve multiple fleets simultaneously, we have to deal with heterogeneous aircraft, characterized by individual turn times. Whether a connection between two activities is valid depends on the assigned tail. We therefore choose to create an individual network for each tail. The arcs of a network represent all valid connections for the particular tail. An arc is defined by four indices: a preceding activity $i \in A$, a succeeding activity $j \in A$, a connection type $r \in R$ and a performing tail $t \in T$. The arc $\langle i, j, r, t \rangle$ is included in O if j starts at the station i arrived at and if the time in-between i and j is sufficient for connection type r performed by tail t.

Parameters: Our model contains several parameters. C'_{ijrt} denotes fictitious costs for assigning arc $\langle i,j,r,t\rangle$. We particularly use this parameter to assign a penalty value to connections that incorporate a maintenance check. In case $r \in M$, C'_{ijrt} represents the work effort for maintenance m. The higher C'_{ijmt} the more laborious, time-consuming and thus expensive is a check of type m. Different properties of maintenance m can be used to derive C'_{ijmt}. Depending on the data availability, it can be derived from the duration, the required man-hours or the real costs. Besides, the parameter C'_{ijrt} is used to penalize undesirable connections and assignments of flights to undesirable fleets or subfleets. The user can also use the cost parameter to set preferences on connections where many through values are realizable.

C''_m is the incremental cost of performing maintenance m before the interval limits are reached. The parameter max_{mq} specifies the allowed interval limit of criterion q between two checks of type m, e.g., the allowed flight hours between two A-Checks.

Decision variables: We include two types of decision variables. The binary decision variable x_{ijrt} is 1 if arc $\langle i,j,r,t\rangle$ is part of the solution and 0 otherwise. y_{imq} are the maintenance counters. They show how many units of criterion q have been accumulated since the last maintenance m after activity i. Now, the main model can be written as a multi-commodity network flow model with side constraints:

$$min \sum_{\langle i,j,r,t\rangle \in O} C'_{ijrt} \cdot x_{ijrt} + \sum_{i \in CO} \sum_{m \in M} \sum_{q \in Q_m} C''_m \cdot y_{imq} \tag{6.1}$$

$$\sum_{\langle i,j,r,t\rangle \in O} x_{ijrt} = 1 \qquad \forall i \in A \backslash CO \tag{6.2}$$

$$\sum_{\langle i,j,r,t\rangle \in O} x_{ijrt} = 1 \qquad \forall j \in CO \tag{6.3}$$

$$\sum_{\langle j,i,r,t\rangle \in O} x_{jirt} = \sum_{\langle i,j,r,t\rangle \in O} x_{ijrt} \qquad \forall i \in A \setminus \{CI \cup CO\}, \forall t \in T \tag{6.4}$$

$$\sum_{\langle i,j,r,t\rangle \in O} x_{ijrt} = 1 \qquad \forall i \in P_t, \forall t \in T \tag{6.5}$$

$$\sum_{\langle i,j,r,t\rangle \in O} x_{ijrt} = 0 \qquad \forall i \in F_t, \forall t \in T \tag{6.6}$$

$$y_{imq} \leq max_{mq} \qquad \forall i \in A, \forall m \in M, \forall q \in Q_m \tag{6.7}$$

$$x_{ijrt} \in \{0,1\} \qquad \forall \langle i,j,r,t\rangle \in O \tag{6.8}$$

$$y_{imq} \geq 0 \qquad \forall i \in A, \forall m \in M, \forall q \in Q_m \tag{6.9}$$

Objective function: Equation 6.1 denotes the objective function of our main model. The first term minimizes the total connection costs. Connections including a maintenance check are characterized by a higher cost factor than direct connections.

The more effort a check requires, the higher the cost factor. Thereby, we minimize the total maintenance effort. The second term of the objective function is needed to avoid distortions due to the finite planning horizon. In this term, we minimize the maintenance counters at the end of the routes. As we minimize the number of checks in the route and the counters at the end of a route, we reward to delay the occurrence of a check until the allowed interval limits have been reached.

Network flow constraints: Constraints 6.2 guarantee that every activity - except carry-out activities - has exactly one successor. Carry-out activities have exactly one predecessor (6.3). The aircraft flow balance constraints ensure that the arc leaving an activity is assigned to the same aircraft as the arc leading to the activity (6.4). Constraints 6.5 - 6.6 force preassigned and block forbidden activities.

Notice that the assignment of multi-leg flights to the same aircraft is implicitly achieved through the definition of the arc set. Assuming i and j are two consecutive legs of a multi-leg flight, then the arc set only includes arcs connecting i with j and no arc connecting i with other activities. In combination with constraints 6.2, the multi-leg flight restrictions are satisfied.

Maintenance constraints: The most complicated constraints are probably constraints 6.7, which guarantee that the maintenance counters y_{imq} never exceed the allowed limits max_{mq}. The values of y_{imq} can be derived from the assignment of arcs.

For simplicity and illustration, let us first assume that activity h is the unique activity preceding activity i and let connection type r connect activity h with activity i. Now, y_{imq} is defined recursively as stated in equations 6.10. The parameter rc_{hirmq} indicates the resource consumption. It specifies how much the maintenance counter of criterion q of maintenance type m is increased if activity h is connected to activity i via connection type r. E_m denotes the set of maintenance m and all more comprehensive maintenances that replaces a check of type m.

$$y_{imq} = \begin{cases} init_{tmq} & if\ i \in CI \\ rc_{hirmq} & if\ r \in E_m \\ rc_{hirmq} + y_{hmq} & if\ r \notin E_m \end{cases} \qquad \forall i \in A, \forall m \in M, \forall q \in Q_m \tag{6.10}$$

In case activity i is a carry-in activity ($i \in CI$), the maintenance counters after activity i are initialized with the start maintenance counters of the corresponding aircraft $init_{tmq}$. In case activity h and i are connected via a resetting maintenance activity ($r \in E_m$), the maintenance counters after activity i equal the resource consumption of connecting h and i via r. In case no resetting maintenance is placed in-between h and i ($r \notin E_m$), the maintenance counters after activity i equal the maintenance counters after activity h plus the resource consumption of connecting h and i via r.

The resource consumption is determined according to equations 6.11. The parameters dur_i and cyc_i indicate the number of flight hours and take offs of activity i. For the calendar time criterion, the resource consumption depends on the preceding activity and the used connection type. The parameter $time_{hirm}$ indicates how many calendar hours are added to the maintenance counter of check m if activities h and i are connected via r.

$$rc_{hirmq} = \begin{cases} dur_i & \text{if } q = \textit{flight time} \\ cyc_i & \text{if } q = \textit{flight cycles} \\ time_{hirm} & \text{if } q = \textit{calendar time} \end{cases} \quad \forall h,i \in A, \forall r \in R, \forall m \in M, \forall q \in Q_m \tag{6.11}$$

In reality, most activities have multiple potential predecessors. We therefore need to incorporate a "big M" formulation in equations 6.10 to select the actually assigned predecessor. Appropriate values as big M are the allowed interval limits max_{mq}. We transform equations 6.10 to equations 6.12 - 6.14.

$$y_{imq} = init_{tmq} \qquad \forall i \in CI, \forall m \in M, \forall q \in Q_m \tag{6.12}$$

$$y_{imq} \geq \sum_{\langle h,i,r,t \rangle \in O} rc_{hirmq} x_{hirt} \qquad \forall h,i \in A, \forall m \in M, \forall q \in Q_m \tag{6.13}$$

$$y_{imq} \geq \sum_{\langle h,i,r,t \rangle \in O} rc_{hirmq} x_{hirt} + y_{hmq} - (1 - \sum_{\langle h,i,r,t \rangle \in O | r \notin E_m} x_{hirt}) max_{mq} \qquad \forall h,i \in A, \forall m \in M, \forall q \in Q_m \tag{6.14}$$

Finally, constraints 6.8 - 6.9 indicate the range of values of the decision variables.

6.3 Model enhancements

Without constraints 6.7, the main model is a pure multi-commodity network flow problem. Constraints 6.7 break the flow structure and raise the complexity of the problem [29]. For realistic instance sizes, the connection networks are extremely large. The sole creation of the connection networks can already take several minutes. In this section, we develop several model enhancements to reduce the problem size.

Aggregation of aircraft

We originally formulated the main model as a multi-commodity problem, in which each aircraft is represented by an own commodity. To reduce the problem size, it

would be desirable to create a single-commodity problem formulation. But this is not possible, since we solve multiple fleets simultaneously and some aircraft have preassigned or forbidden activities and therefore have to be treated separately. However, we observe that most aircraft in a fleet neither have preassigned nor forbidden activities. These aircraft are identical in terms of optimization and can be treated as a single commodity. By aggregating multiple aircraft in a single aircraft group, we can reduce the problem size drastically. Notice that different start positions and initial conditions of the aircraft are irrelevant, because they are introduced to the routes via the carry-in activities.

From now on, we will not consider separate tails anymore, but aggregate all aircraft of the same fleet that have the same list of allowed activities in a common group. Aircraft that have an individual list of allowed activities are represented by an own aircraft group.

Let G denote the set of aircraft groups. We transform the binary decision variable x_{ijrt} to x_{ijrg} telling us whether arc $\langle i, j, r, g\rangle$ is part of the solution.

Only create arcs for allowed aircraft groups

Another valuable observation is that not all aircraft are allowed to perform all activities. In preprocessing, we can identify the subset of aircraft groups that are allowed to perform a particular activity. We then create only arcs for those aircraft groups that are allowed to perform both of the connected activities. Whenever an activity is preassigned to a specific tail, by consequence all other aircraft groups are not allowed to perform any connection to or from this activity. With this concept, we can eliminate constraints 6.5 and 6.6, which force preassigned and restrict forbidden activities. As only the variables of valid assignments are available, the compactness of our model is further increased. The enhanced main model looks as follows:

$$min \sum_{\langle i,j,r,g\rangle \in O} C'_{ijrg} \cdot x_{ijrg} + \sum_{i \in CO} \sum_{m \in M} \sum_{q \in Q_m} C''_m \cdot y_{imq} \tag{6.15}$$

$$\sum_{\langle i,j,r,g\rangle \in O} x_{ijrg} = 1 \qquad \forall i \in A \backslash CO \tag{6.16}$$

$$\sum_{\langle i,j,r,g\rangle \in O} x_{ijrg} = 1 \qquad \forall j \in CO \tag{6.17}$$

$$\sum_{\langle j,i,r,g\rangle \in O} x_{jirg} = \sum_{\langle i,j,r,g\rangle \in O} x_{ijrg} \qquad \forall i \in A \setminus \{CI \cup CO\}, \forall g \in G \tag{6.18}$$

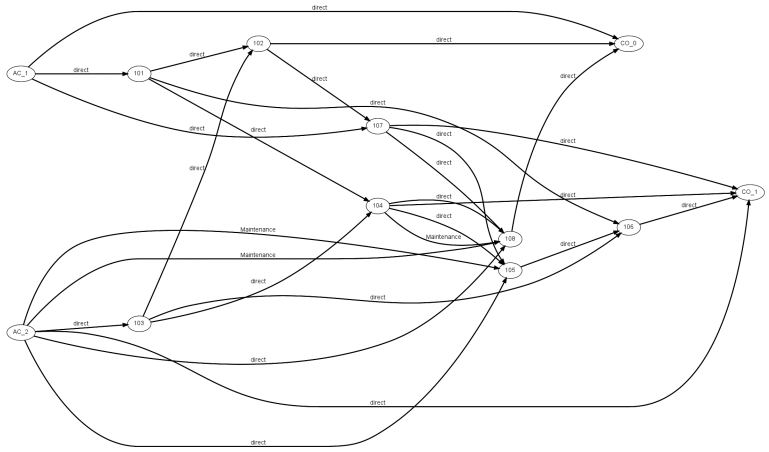
$$y_{imq} \leq max_{mq} \qquad \forall i \in A, \forall m \in M, \forall q \in Q_m \tag{6.19}$$

$$x_{ijrg} \in \{0, 1\} \qquad \forall \langle i, j, r, g\rangle \in O \tag{6.20}$$

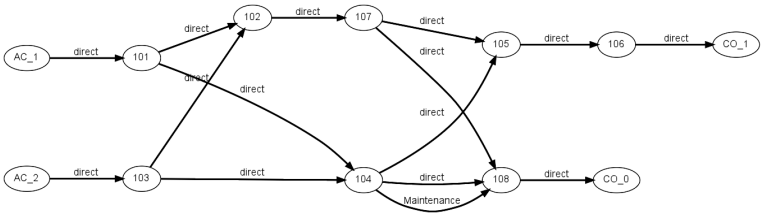
$$y_{imq} \geq 0 \qquad \forall i \in A, \forall m \in M, \forall q \in Q_m \tag{6.21}$$

Elimination of arcs that violate aircraft flow balance

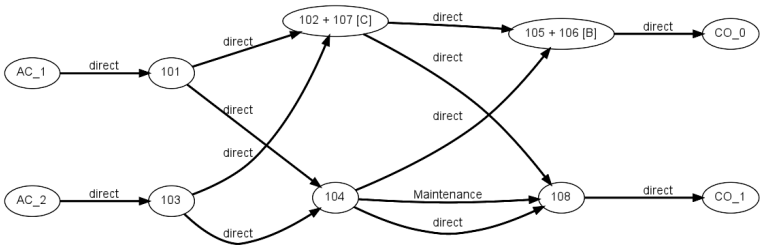
The arc set O can be further reduced. Since all activities have to be performed, the aircraft flow is known in advance. We know the location and time the aircraft become available as well as when and where activities start and end. Therefore, we can compute the number of aircraft on ground at any airport at any time. We also know that valid connections require the aircraft to stay on ground during the connection time. These two observations allow us to conclude that whenever the aircraft flow balance equals zero at an airport, all arcs connecting activities at that airport over this point in time are invalid. By removing them from the set of arcs, we reduce the size of O without losing valid solutions [31].



(a) Network of example schedule after aggregating aircraft



(b) Network of example schedule after removing invalid arcs



(c) Network of example schedule after aggregating activities

Figure 6.1: Enhancements of connection networks

The effect of this enhancement on our small example schedule with two aircraft and eight flights as presented in Sec. 4.1 is shown in Fig. 6.1. Fig. 6.1a equals Fig. 4.2. It shows the connection network of the example schedule after aggregating aircraft. Fig. 6.1b shows the reduced network after eliminating invalid arcs.

Aggregation of activities

Another valid model enhancement is to aggregate activities. Whenever an activity can only be connected to a single other activity and no maintenance can be planned in-between, that is, whenever there is only a single direct connection arc leaving an activity, we can remove the two activities and the interlinking arc from the networks. Instead, we create an aggregated activity departing from the origin of the first activity and arriving at the destination of the second activity. The resource consumption of the aggregated activity equals the combined resource consumption of the replaced activities. The enhancement is especially effective in hub-and-spoke networks. The schedules are typically designed so that an aircraft heads out from the hub to a spoke station and returns after a short turnaround. As there is no other aircraft available at the spoke station except for the aircraft performing the arriving flight, it is obvious that the same aircraft needs to perform the returning flight. Also, it is often impossible to perform any maintenance at the spoke station.

Although we might be able to reduce the number of activities considerably, the benefit of this enhancement is fairly marginal. For every two activities we aggregate, we only eliminate a single connection arc. The vast majority of connection arcs exists at the hub stations and is not affected by this enhancement. Fig. 6.1c shows the adjusted network of our example schedule after aggregating activities. In Fig. 6.1b, we observe that there is only one arc leaving activity 102 and 105. We therefore can conclude that activities 102 and 107 as well as 105 and 106 are definitely performed consecutively by a single tail. In Fig. 6.1c, we see that these activities have been aggregated in a single node in each case.

6.4 Maintenance capacity

There are two possibilities to include the limited capacity of the maintenance stations in the model. The first option is to add a constraint forbidding any excess of utilization. The second option is to consider a penalty term in the objective function for exceeding the available capacity. We have selected the second option. The reason for this choice is the definition of the maintenance arcs. A maintenance arc $\langle i, j, m, g \rangle$ in our model implies that a single maintenance check of type m is performed in-between activities i and j. The start of the check is scheduled as late as possible so that the check ends just before the departure of activity j minus a short buffer time. However, the ground times in-between two activities are often longer than the required time to perform a check, and it might be possible to reschedule the

actual check times to an earlier point in time without compromising the maintenance feasibility of the solution. We observe that a capacity overload can often be reduced by subsequent manual rescheduling.

Example: Short range aircraft are mainly used during daytime and usually spend the entire night at an airport. The definition of the maintenance arcs states that the overnight checks of all aircraft are done right before their first departures in the morning. In reality, maintenance can be done anytime during the night and it is assumable that the airline distributes the occurring maintenance effort over the entire available time period. □

For our capacity considerations, we assume that hangar space is the limiting resource. When checking whether the maintenance capacity is respected, we only need to focus on those stations S where hangar checks are performed. It is also not necessary to continuously examine the number of aircraft in maintenance, but only at the starting points of checks when the number of aircraft in maintenance at a station might increase. As the maintenance arcs and preassigned maintenance activities are defined a priori, we know the potential starting points of hangar checks in advance. We merely need to focus on those points in time when the maximum potential number of simultaneously maintained aircraft exceeds the available maintenance capacity K_s of the corresponding station. The critical points at station s are collected in the set B_s.

We introduce a new decision variable z_{sb} which denotes the excess of capacity in number of aircraft at station s at critical point b. We create new sets of constraints to measure the maintenance utilization (6.22 - 6.23). $\hat{O}_{sb}$ denotes the set of maintenance arcs that incorporate a hangar check at station s at point in time b.

$$z_{sb} \geq \left(\sum_{\langle i,j,r,g \rangle \in \hat{O}_{sb}} x_{ijrg} \right) - K_s \qquad \forall s \in S, \forall b \in B_s \tag{6.22}$$

$$z_{sb} \geq 0 \qquad \forall s \in S, \forall b \in B_s \tag{6.23}$$

With these constraints, we can add the following term (6.24) as summand to the objective function. C''' represents a penalty cost for exceeding the available maintenance capacity by one aircraft.

$$\sum_{s \in S} \sum_{b \in B_s} C''' \cdot z_{sb} \tag{6.24}$$

We now have completed our main model. In the following chapter, we describe the ALNS algorithm we use to obtain problem solutions.

7 Solution approach

As we have explained in Chap. 5, every LNS approach consists of two stages: the initial solution stage and the improvement stage. In this chapter, we describe how we have designed these two stages for our optimizing approach.

7.1 Initial solution stage

Starting point of our LNS algorithm is the construction of a feasible initial solution. Considering that many studies treat tail assignment as a pure feasibility problem, the challenge of finding a feasible initial tail assignment solution becomes obvious.

During our research, we have identified three generic approaches to create a feasible solution for a problem. The first and most pragmatic approach is to create a mixed-integer program (MIP) representation and use a MIP solver, but solve the MIP as a pure feasibility problem. The second idea is to heuristically construct a feasible solution. The specific heuristic can be based on naïve strategies, like greediness, as well as more intelligent strategies. The idea of the third approach is to design an iteratively solved row generation framework as illustrated in Fig. 7.1. The original problem is split up in a restricted master problem (RMP) and a subproblem. As the RMP is only a partial representation of the original problem, it is much faster to solve. However, it is not guaranteed that the obtained solution is feasible for the original problem. We use the subproblem to complete the solution and analyze possible conflicts. If the obtained solution is infeasible, we start a new iteration. We create a constraint which inhibits the conflict from recurring in the following iterations. The constraint is added to the RMP before it is resolved. We continue the iteration loop until we obtain a solution which is feasible for the original problem.

The third approach features elements of a logic-based Benders decomposition, which is why we will refer to it as *Benders framework*. We apply the concept of "learning from one's mistakes", which is also prevalent in constraint programming under the rubric of nogoods [39]. Whenever we detect that a certain partial solution, provided by the RMP, cannot be completed to obtain a feasible solution for the original problem, we examine the reasons for this conflict. Based on this information, we can formulate a constraint that excludes a number of infeasible partial solutions in the RMP. In logic-based Benders decomposition, such a constraint is referred to as a *Benders cut*. In constraint programming, it is known as a *nogood* [39].

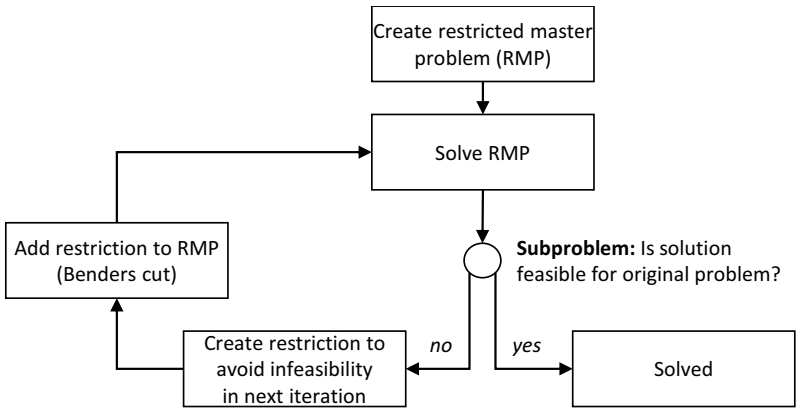


Figure 7.1: Structure of Benders framework

The three generic approaches are the basis for our attempts to create a feasible initial solution for the tail assignment problem. We have implemented all three ideas and analyzed their performance on 220 real-world instances. The instances are based on data from two major international carriers. In chapter 8, we describe how the instances are created. In Tab. 7.1, we present a subset of eight example instances. The chosen example instances are heterogeneous. They differ in length of planning horizon, number of fleets, number of aircraft and schedule size. The upper four instances are based on data of *Carrier One*, the bottom four on data of *Carrier Two*.

Throughout this chapter, we use the example instances to illustrate our proceedings. However, all results and conclusions are based on the entire set of 220 instances. For our benchmarking, we employ the non-commercial MIP solver SCIP 3.1.1. We use LP files to communicate with the solver. We set a solving time limit of 30 minutes per instance. Our algorithms are coded in Python 3.4 programming language. The test runs are carried out on a system with quad core 3.20 GHz Intel Core i5-3470 processors and 8GB RAM running under 64-bit Windows 7 operating system.

Table 7.1: Example instances

Instance	Days	(Sub-) Fleets	Aircraft	Activities	Airports	Preassigned or curfews
03-767x	3	2	65	330	34	No
05-737	5	1	186	2,469	68	No
07-757	7	1	133	2,031	53	No
28-777	28	1	55	1,531	12	No
03-380x^{t}	3	2	31	165	124	Yes
05-380x	5	4	55	409	124	No
10-330/340x^{t}	10	4	29	727	124	Yes
10-777x	10	4	28	660	124	No

x: mixed (sub-)fleets, ^{t}: tail-dependent restrictions (preassigned or curfews)

In our evaluation, we analyze the ratio of performance indicators using different approaches and calculate the mean of this ratio across all instances. We use the geometric mean instead of the arithmetic when evaluating relative performances. The geometric mean for a series of n values $x_1, x_2, ..., x_n$ is defined as equation 7.1.

$$\overline{x}_{geom} = \sqrt[n]{\prod_{i=1}^{n} x_i} \tag{7.1}$$

7.1.1 Main Model as feasibility problem

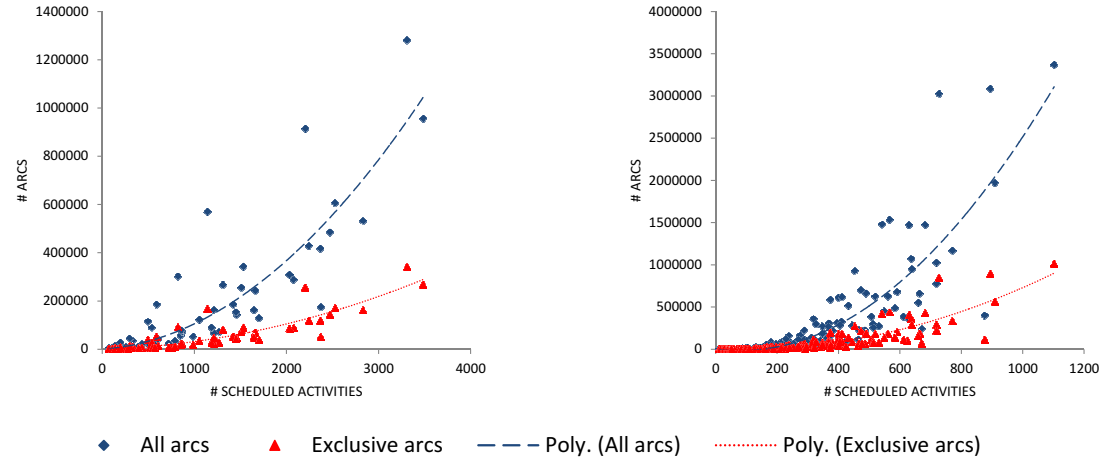
We start our analysis by solving the main model as a pure feasibility problem using a MIP solver. Our main model is a comprehensive linear representation of the tail assignment problem. We replace the original objective function with the dummy objective function $min\, Z = 0$. In Tab. 7.2, we present problem characteristics and solving times of our eight example instances. We are able to generate initial solutions for only two instances within the time limit of 30 minutes. Across all 220 instances, we are able to create initial solutions for the smallest 92 instances.

The quadratically increasing number of arcs is responsible for slowing down solving times of larger instances. However, since we are interested in finding any feasible solution, we can reduce the number of arcs by implementing an exclusive arc definition. Like Haouari et al. and Grönkvist, we only allow any two activities to be connected by a single connection arc [29, 33]. To be conservative, we eliminate all arcs but the one that is the hierarchically highest from maintenance point of view. In other words, we place the hierarchically highest possible maintenance check in-between any two activities. As we eventually solve multiple fleets simultaneously and the maintenance programs of different fleets might vary, we have to provide the hierarchically highest possible maintenance arcs for all fleets that are allowed to perform both connected activities.

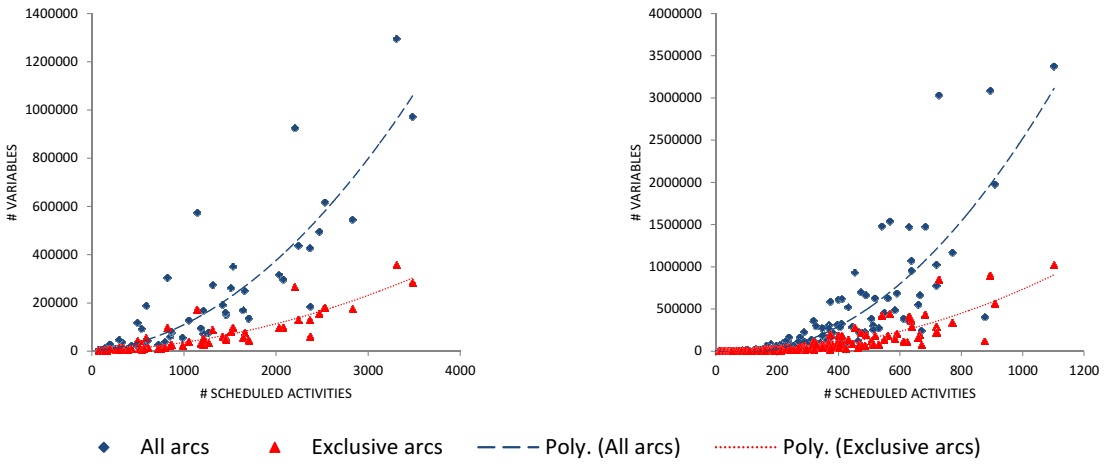
Table 7.2: Benchmarking: main model as feasibility problem

	All arcs				**Exclusive arcs**			
Instance	Arcs	Variables	Constr.	Time [s]	Arcs	Variables	Constr.	Time [s]
03-767x	34,072	35,912	47,160	1577	11,329	13,169	47,160	54
05-737	483,739	495,103	1,083,502	-	142,413	153,777	1,083,502	501
07-757	306,429	315,617	678,884	-	85,328	94,516	678,884	195
28-777	342,760	349,324	715,484	-	90,003	96,567	715,484	162
03-380x^{t}	56,476	57,384	34,536	1753	22,529	23,437	34,536	250
05-380x	323,691	325,767	228,543	-	102,987	105,063	228,543	-
10-330/340x^{t}	3,024,165	3,027,305	606,209	-	843,819	846,959	606,209	-
10-777x	547,852	550,716	306,836	-	153,480	156,344	306,836	-

Our analysis shows that the number of arcs can on average be reduced by 68 percent. The number of variables is reduced by the same absolute amount as the arcs, which equals a relative mean reduction of 63 percent. The number of constraints, on the other hand, is unaffected by this enhancement.



(a) Change in number of arcs for Carrier One (left) and Carrier Two (right)



(b) Change in number of variables for Carrier One (left) and Carrier Two (right)

Figure 7.2: Change of problem characteristics depending on arc definition

Fig. 7.2 illustrates the reductions in number of arcs and variables for both airlines. The dashed/ dotted lines represent second-order polynomial trend lines. Although the scatter plots and trends look similar for both carriers, there is a significant difference. Given an identical number of scheduled activities, the numbers of arcs and variables are generally higher and increase a lot earlier for Carrier Two (observe

different range and scale of axes). This phenomenon can be explained by the network structures of the airlines. While Carrier One operates five different hubs, Carrier Two serves all his flights out of a single hub. The large number of connection possibilities at this station is responsible for the higher complexity of the instances.

The enhanced results are shown in Tab. 7.2. They prove that an exclusive arc definition is capable of reducing problem sizes and solving times considerably. Nevertheless, the reported solving times to create a feasible initial solution are still too long to be acceptable. Within a time limit of 30 minutes, the enhancement allows us to generate initial solutions for five of our eight example instances and 138 out of our 220 instances. Fig. 7.3 shows the relation between the size of the schedule and the solving time for Carrier One. The dashed line is again a second-order polynomial trend line. The figure highlights that the solving times increase drastically with the size of the schedule, even when implementing an exclusive arc definition.

Figure 7.3: Relation between solving times and schedule size for Carrier One

7.1.2 Heuristics

Next, we investigate heuristic approaches to construct an initial tail assignment solution. We consider greediness and other naïve sequencing methods.

First-in, first-out routing

The simplest method is to apply a first-in, first-out (FIFO) rule to create the routings of the aircraft. Pseudo-code for the naïve FIFO algorithm is shown in Tab. 7.3. In

a loop, we look at aircraft by aircraft and add the next available activity departing from the current position to the route of the aircraft (line 1-10). This procedure is repeated until the end of the planning horizon is reached (line 9). We then continue with the routing for the next aircraft. The activities that are already assigned to previous routed aircraft are obviously not available anymore (line 10). Special attention needs to be paid to multi-leg flights. To make sure that all legs are assigned to the same aircraft, we neglect the FIFO rule whenever encountering a multi-leg flight and instead add the succeeding legs to the route of the aircraft (line 6-7).

Table 7.3: Pseudo-code for FIFO heuristic

FIFO algorithm	
1:	**for** t in aircraft
2:	initialize current position with start position of aircraft t
3:	**repeat**
4:	find next activity departing from current position that is not rear leg in a multi-leg flight
5:	add activity to route of aircraft t
6:	**if** activity is first leg of multi-leg flight **then**
7:	add all other legs to route of aircraft t
8:	update current position
9:	**until** end of planning horizon is reached
10:	remove activities performed by aircraft t
11:	**for** t in aircraft
12:	determine required maintenance checks to perform route
13:	**for** activity in route of aircraft t
14:	**if** activity ends at maintenance base **and** succeeding ground time sufficient for maintenance **then**
15:	find hierarchically highest of required checks that fits in ground period
16:	place check in route of aircraft t

After completing the routing for all aircraft, we manually incorporate maintenance checks in-between the planned activities (line 11-16). We first analyze which check types are required for a particular aircraft to perform its route (line 12). A check type is required when the initial counters plus the accumulated resource consumption on the route exceed the check limits. We then insert the hierarchically highest of the required check types in every ground period at a maintenance station for which the available ground time is sufficient (line 13-16).

Example: For illustration of the FIFO algorithm, let us consider the simple schedule of two aircraft, three airports and eight flights as presented in Sec. 4.1. The graphical representation is shown in Fig. 7.4. Let us assume that only a single type of maintenance exists, which takes five hours and can only be performed at airport A. The maintenance condition of aircraft 2 is critical. Only two additional take offs are allowed before reaching the maintenance interval limits. For simplification, let us further assume that both aircraft have neither preassigned nor forbidden activities.

The FIFO algorithm starts with the routing of aircraft 1, which is currently located at airport C. The algorithm assigns the next departing activity to the route, i.e., flight 101. Next, aircraft 1 performs flight 102, because it is the first activity departing from airport B after the arrival of flight 101. Similarly, flights 107, 105 and 106 are added to the route of aircraft 1. As there is no activity starting at airport A after the arrival of flight 106, the end of the planning horizon has been reached. In the next step, we create the routing for aircraft 2. Its route starts with flight 103. According to the FIFO rule, we would then add flight 102, because it is the first activity departing from airport B after the arrival of flight 103. However, this activity is already performed by aircraft 1 and hence not available anymore. The first available activity is flight 104. The routing of aircraft 2 continues with flight 108, after which the end of the planning horizon is reached.

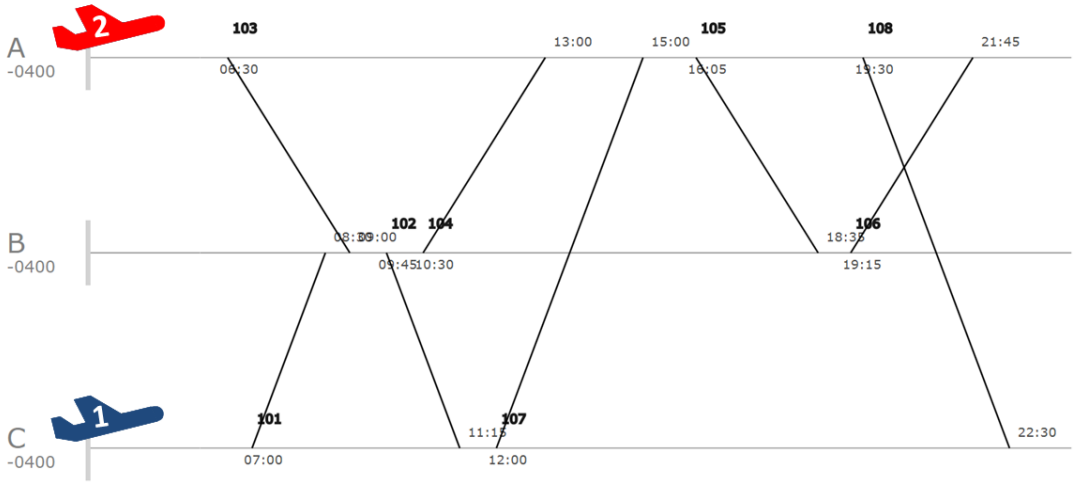


Figure 7.4: Example schedule

After obtaining the routings for both aircraft, we focus our view on maintenance. The route of aircraft 1 does not provide sufficient ground time at airport A, thus no check can be incorporated. Aircraft 2, on the other hand, spends six and a half hours at airport A in-between flights 104 and 108. When placing a check in this ground period, we create a valid solution. Aircraft 2 performs exactly two flights before the required check and therefore does not exceed the allowed interval limits. □

The main advantage of the FIFO rule is its simplicity and easy implementation. The required time to run the algorithm is extremely low. The average solving time is 0.2 seconds. However, we observe that in many cases it is not suitable to find a maintenance feasible routing. We are able to generate feasible solutions for only 42 out of 220 instances. Especially, the first routed aircraft often violate the maintenance interval limits. Due to the FIFO rule, these aircraft dispose of the shortest ground times, which are often too short to incorporate sufficient maintenance checks.

Example: Consider the example schedule again, but this time we assume that the maintenance condition of aircraft 1 is critical and only two more take offs are allowed for this aircraft. As stated above, the route of aircraft 1 consists of five flights, but no check can be placed in-between. Aircraft 1 would reach the allowed interval limits after performing flights 101 and 102. It would not be allowed to depart for flight 107 without prior maintenance. In this simple example, we see that the performance of the FIFO algorithm largely depends on the routing order of the aircraft. It also demonstrates that a more intelligent routing rule is inevitable. □

Priority routing

An alternative idea is to employ a priority routing. In contrast to the naïve FIFO rule, this algorithm evaluates the impact on the subsequent routing step when taking a decision. The pseudo-code is shown in Tab. 7.4. We first sort the aircraft based on their urgency for maintenance (line 4). Then, we start with the routing for the most urgent aircraft (line 5). We try to incorporate the necessary ground time for the required check at a suitable station as soon as possible in the route. If the current station is incapable of performing the check or no activity departing after the required ground time is available without violating the aircraft flow balance, the aircraft is assigned to the earliest departing flight arriving at a capable maintenance station (line 10-11). If such a flight is also not existent, we use the FIFO rule and assign the next departing activity to the aircraft (line 12-14). This procedure is repeated until we were able to place a maintenance slot in the route (line 18). We then stop the routing for this aircraft, update its check counters (line 19) and renew the aircraft sorting (line 4). In the next iteration, we repeat the same steps with the currently most urgent aircraft (line 5). Whenever the routing of an aircraft reaches the end of the planning horizon, we remove it from the aircraft list (line 21-22). The routing is finished when all aircraft have been removed from the aircraft list (line 3).

As in the FIFO approach, we then incorporate the largest possible of the required maintenance types in every ground period at a maintenance station (see Tab. 7.3, line 11-16). The created slots in the routes of the aircraft support us in incorporating required maintenance checks in routes of urgent aircraft.

Example: For a better understanding, we again use the simple schedule and the assumptions from above to describe our algorithm. The schedule consists of eight flights to be performed by two aircraft out of which aircraft 1 is maintenance critical.

Our algorithm starts by routing aircraft 1. Our goal is to incorporate a ground time of more than five hours at airport A as soon as possible in the route of this aircraft. The aircraft is currently located at airport C. As there is no other aircraft available and all activities must be performed, flight 101 is assigned to aircraft 1. The flight arrives at airport B, which is incapable of performing the required check. Next, we look at the available activities departing from airport B after the arrival of

Table 7.4: Pseudo-code for priority heuristic

```
Priority algorithm
1:  initialize maintenance counters of aircraft with start counters
2:  initialize current positions of aircraft with start positions
3:  while aircraft list
4:      sort aircraft based on urgency for maintenance
5:      select most critical aircraft t
6:      repeat
7:          if (current station capable of performing required check and sufficient ground time
                available without violating aircraft flow balance) then
8:              add next departing activity from current position after required ground time
                that is not rear leg in a multi-leg flight to route of aircraft t
9:          else
10:             if (activity that is not rear leg in a multi-leg flight departing from current position
                    to capable maintenance base available without violating aircraft flow balance) then
11:                 add activity to route of aircraft t
12:             else
13:                 find next activity departing from current position that is not rear leg in a multi-leg flight
14:                 add activity to route of aircraft t
15:         if added activity is first leg of multi-leg flight then
16:             add all other legs to route of aircraft t
17:         update current position
18:     until slot for required check has been created or end of planning horizon reached
19:     update maintenance counters of aircraft t
20:     remove activities performed by aircraft t
21:     if end of planning horizon is reached then
22:         remove aircraft from aircraft list
```

flight 101. There are two candidates, i.e., flights 102 and 104. As flight 104 is bound for maintenance base A, we assign it to the route of aircraft 1. In the next step, we check if it is possible to incorporate a ground time of at least five hours after flight 104. That means, we check if an activity departing from airport A after the required ground time is available without violating the aircraft flow balance. In this example, it is possible to assign flight 108 to aircraft 1 allowing sufficient ground time in-between flight 104 and 108. The aircraft flow balance is not violated, because the departing flight 105 can be performed by the aircraft arriving on flight 107. After flight 108, the end of the planning horizon is reached. We thus remove aircraft 1 from the aircraft list. The routing of aircraft 2 is done in the next step. The algorithm assigns the remaining flights 103, 102, 107, 105 and 106 to this aircraft.

After completing the routings for both aircraft, we add checks to the available ground periods. As the route of aircraft 1 provides a maintenance slot in-between flights 104 and 108, the obtained solution is maintenance feasible. □

Our benchmarking results show that the priority routing outperforms the naïve FIFO routing. Even though the average solving times have more than doubled to 0.5 seconds, the algorithm generates feasible solutions for 122 out of our 220 instances. As urgent aircraft are treated with priority, it is more suitable to avoid exceeding the interval limits and therefore infeasibilities. However, also when applying this more intelligent heuristic, we still cannot find feasible initial solutions for all analyzed scenarios. All heuristic routing methods share the drawback that finding a feasible solution is not guaranteed. Also, the presented heuristics are incapable of treating preassigned activities. Therefore, they are unsuitable to be used directly in the initial solution stage of our LNS approach. To benefit from their outstanding low time requirement, we have to integrate them in a larger framework.

7.1.3 Benders framework

As the first two ideas did not yield the desired results, we have also analyzed the third generic approach. The tail assignment problem is split up in two problems. In the restricted master problem, we consider the routing of the aircraft. To increase the solving speed, we do not include any maintenance restrictions. The subproblem addresses the maintenance feasibility of the obtained routings.

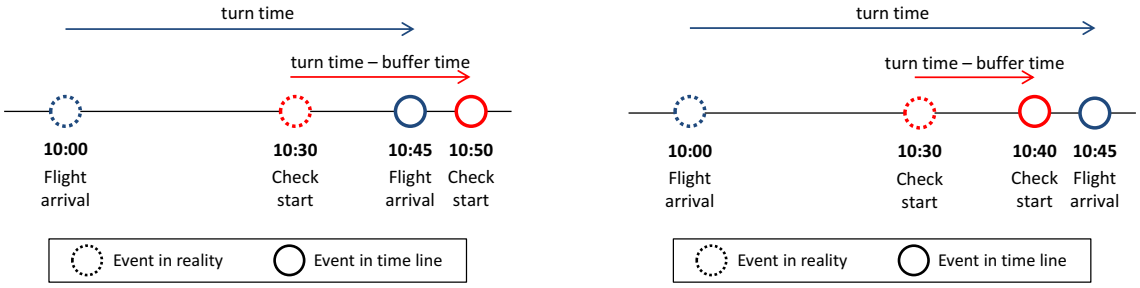
Restricted master problem

We express the RMP as a MIP. The model is based on a time-space network. For our main model, we concluded that a connection network formulation is preferable, because it is more expressive. For the RMP, we do not need a high degree of expressiveness. The goal is to quickly obtain aircraft routings. Concerning the required time to solve a problem, we see a major benefit of the time-space formulation.

As base model, we formulate the RMP as a multi-commodity network flow model. We create an individual time-space network for each aircraft. The arc set represents the scheduled activities. The nodes represent departure and arrival events at a station. Ground arcs connect the events at a station. To take into account turn time restrictions, we delay the event time of flight arrivals by the required turn time. We assume that the turn time is fleet and airport dependent.

A problem of delaying flight arrival events arises when a flight is followed by a maintenance activity. In this case, it is not necessary to preserve the required time for a full turnaround, because no passengers and cargo are loaded. Instead, we consider a buffer time to unload the aircraft and prepare it for the check (e.g., tug to a hangar). Another buffer time is required for re-entry into service after the check is finished. To map these buffer times in our networks, we delay the start events of prescheduled maintenance activities by the turn time of the preassigned aircraft at the particular station minus the buffer time for entering into maintenance. The arrival events are delayed by the buffer time required for re-entering into service.

Example: Let us illustrate this idea with an example as shown in Fig. 7.5a. Consider a flight arriving at 10:00am. The turn time at the airport is 45 minutes. We thus place the arrival event at 10:45am in the time line of the airport. Now, let us assume there is a preassigned maintenance activity for the aircraft which starts at 10:30am and the buffer time for entering into maintenance is 25 minutes. Obviously, it should be possible to perform the maintenance after the arriving flight. However, the network structure only allows the flight to be followed by activities starting after 10:45am. By delaying the flight arrival event, we have blocked a valid connection.



(a) Buffer time of 25 minutes **(b)** Buffer time of 35 minutes

Figure 7.5: Event dates in time-space network

As solution, we propose to delay the start event of the maintenance activity by the turn time minus the buffer time, hence 20 minutes. In doing so, it is placed at 10:50am in the time line, and a connection from the flight to the maintenance activity is possible. If the time between the flight and maintenance activity is shorter than the buffer time, they cannot be performed consecutively by the same tail. For example, we would place the start event of the maintenance activity at 10:40am if the buffer time is increased to 35 minutes (see Fig. 7.5b). □

It should be noted that this manipulation only works when the duration of the maintenance activity is longer than the turn time. Otherwise, we would place the start event of the maintenance activity behind the arrival event in the time line. Considering that the typical turn time is less than 60 minutes and that check durations are scheduled generously to allow for unforeseen occurrences, this assumption is valid in most situations.

The resulting networks consist of all activities that the particular aircraft is allowed to perform. Preassigned activities are only added to the network of the respective aircraft. For modeling and algorithmic purposes, we also add a dummy carry-in event (arrival node) that represents the time and place the aircraft becomes available to each network. The model is treated as a pure feasibility problem.

Let T be the set of aircraft. A denotes the set of activities, which also includes dummy carry-in activities CI. Let S be the set of airports (stations) in the schedule. N_{st} denotes the set of nodes at airport s for tail t. O is the set of arcs in the time-space networks. An arc in O comprises four elements $\langle i, t, dep, arr \rangle$: $i \in A$ denotes the

activity, $t \in T$ the performing aircraft, $dep = \langle s^d, n^d \rangle$ the tuple of departure airport $s^d \in S$ and node $n^d \in N_{s^d t}$ and $arr = \langle s^a, n^a \rangle$ the tuple of arriving airport $s^a \in S$ and node $n^a \in N_{s^a t}$. The sets O^{out}_{tsn} and O^{in}_{tsn} contain all arcs that depart from, or arrive at, respectively, node n in the time line of airport s in the network of aircraft t. The binary decision variable $x_{i,t,dep,arr}$ is 1 if arc $\langle i, t, dep, arr \rangle$ is included in the solution and 0 otherwise. Besides, we introduce the decision variable y_{tsn} which is 1 if aircraft t is on ground at airport s after node n and 0 otherwise. Due to the model structure, we can specify this variable as continuous on the interval $[0, 1]$, even though it is binary in reality.

To guarantee valid assignments of multi-leg flights, we combine all legs in a single activity starting at the origin of the first leg and ending at the destination of the last leg. The base model can be stated as follows:

$$min\, Z = 0 \tag{7.2}$$

$$\sum_{\langle i,t,dep,arr \rangle \in O} x_{i,t,dep,arr} = 1 \qquad \forall i \in A \tag{7.3}$$

$$y_{ts,n-1} - \sum_{\langle i,t,dep,arr \rangle \in O^{out}_{tsn}} x_{i,t,dep,arr} + \sum_{\langle i,t,dep,arr \rangle \in O^{in}_{tsn}} x_{i,t,dep,arr} = y_{tsn} \qquad \forall t \in T, \forall s \in S, \forall n \in N_{st} \tag{7.4}$$

$$y_{ts0} = 0 \qquad \forall t \in T, \forall s \in S \tag{7.5}$$

$$x_{i,t,dep,arr} \in \{0, 1\} \qquad \forall \langle i, t, dep, arr \rangle \in O \tag{7.6}$$

$$y_{tsn} \in [0, 1] \qquad \forall t \in T, \forall s \in S, \forall n \in N_{st} \tag{7.7}$$

As we solve a pure feasibility problem, we incorporate a dummy objective function (7.2). The solution assigns all activities in the schedule to the aircraft (7.3) and assures a balanced flow at each node of the networks (7.4). For initialization, the numbers of aircraft on ground before the first nodes are set to zero (7.5).

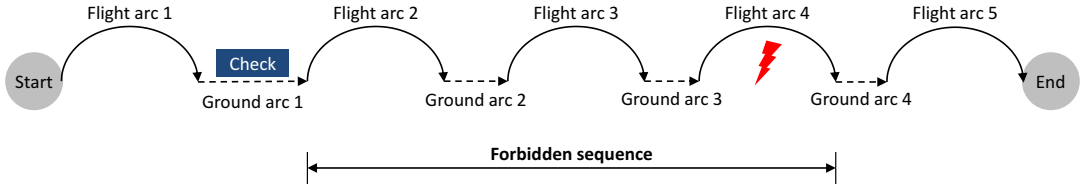
Subproblem

From the solution of the RMP, we can determine the route of each aircraft. Afterwards, we manually incorporate the largest possible required maintenance check in the ground times at maintenance stations. However, since we did not consider any maintenance restrictions in the RMP, we do not necessarily obtain a maintenance feasible routing. Our subproblem algorithm identifies and analyzes existing conflicts. They occur whenever a maintenance interval limit is exceeded on the route of an aircraft. That is, whenever we were not able to incorporate sufficient maintenance checks in the route. The main reasons for infeasibilities are too short ground times at maintenance stations and too long ground times at non-maintenance stations.

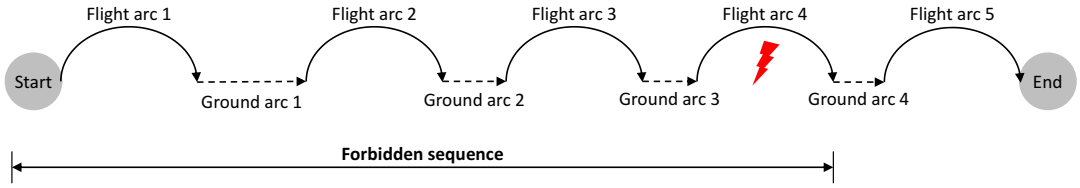
Benders cuts

We solve RMP and subproblem iteratively until we find a maintenance feasible routing. We use additional constraints to avoid running into the same conflicts in subsequent iterations.

The additional constraints restrict the sequence of activity and ground arcs that led to an infeasibility. We do not need to block the entire route of the affected aircraft, instead it suffice and is more efficient to block the minimal forbidden sequence. This sequence starts with the first activity after the last check that resets the exceeded maintenance counter (see Fig. 7.6a). If the route does not include such a check before the infeasible activity, the forbidden sequence starts with the carry-in activity of the route (see Fig. 7.6b). The end of the sequence is the infeasible activity itself. To block this sequence in subsequent iterations, we have to extend our RMP by a new restriction. This restriction works like a Benders cut and guarantees that at least one activity arc or one ground arc of the forbidden sequence is not assigned to the aircraft in the next iterations. As the additional constraints constitute of a single linear inequality per conflict, they do not strongly increase the solving time.



(a) Minimal forbidden sequence with prior resetting check



(b) Minimal forbidden sequence without prior resetting check

Figure 7.6: Minimal forbidden sequence

Let O' be the set of activity arcs and W' the set of ground arcs in the forbidden sequence. We can then write the Benders cut as stated in equation 7.8. To keep the model slim, we only create a Benders cut for one conflict in every iteration, even when multiple routes are maintenance infeasible. By forcing changes in the route of one aircraft, the routes of other aircraft are compulsory changed as well. We add the Benders cut to the RMP before it is resolved. Later in this chapter, we evaluate strategies of adding multiple Benders cuts to the RMP in every iteration.

$$\sum_{\langle i,t,dep,arr\rangle \in O'} x_{i,t,dep,arr} + \sum_{\langle t,s,n\rangle \in W'} y_{tsn} \leq \mid O' \mid + \mid W' \mid -1 \tag{7.8}$$

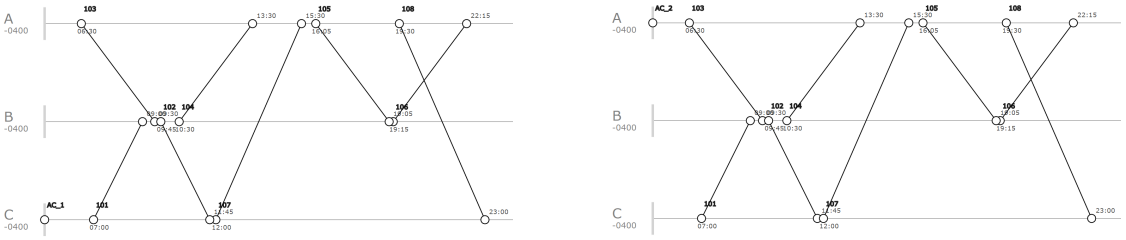
Our communication with the MIP solver is non-incremental. In every iteration, we create a new LP file with the updated problem and hand it to the solver. We chose this method for ease of implementation. However, an incremental communication would perform better, since the solver would not need to reconstruct the problem in every iteration, and also it could use the previous solution or LP basis as warm start.

On the left hand side of Tab. 7.5, we present problem and solving characteristics of our eight example instances. We observe that the size and solving time of the RMP increase rapidly with the number of aircraft and number of scheduled activities impeding us to quickly find solutions for many instances. Providing a time limit of 30 minutes per instance, the base model integrated in a Benders framework allows us to generate initial solutions for 149 of our 220 instances. However, there are a couple of preprocessing techniques to reduce the size of the time-space networks and thus improve the solving time of the RMP. Some of the enhancements are similar to the ones presented for our main model in Sec. 6.3. We will use the simple schedule consisting of two aircraft and eight flights from Sec. 4.1 to illustrate the enhancements. The initial networks of the two aircraft are shown in Fig. 7.7a.

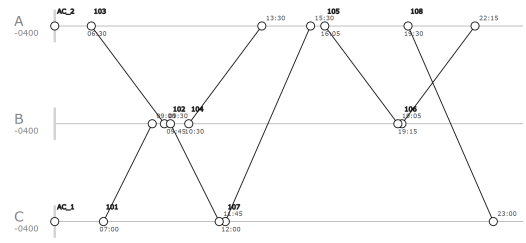
Aggregation of aircraft

When examining the time-space networks, it is noticeable that they are identical for the majority of aircraft in a (sub-)fleet. We can reduce the size of our RMP by aggregating as many aircraft of the same (sub-)fleet as possible in a common network. However, as we cannot control which activity is assigned to which aircraft in the group, we still have to treat aircraft with special restrictions on allowed activities or preassigned activities separately in own aircraft groups. We will refer to aircraft groups consisting of a single aircraft as *single aircraft groups* and aircraft groups consisting of multiple aircraft as *multi aircraft groups*.

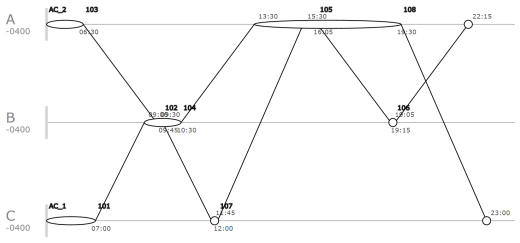
Since the connections between two activities are not explicitly defined in time-space networks, the actual sequence of consecutively performed activities by an aircraft is not part of the solution of the RMP. The restrictions of the RMP only guarantee that the flow at each node is balanced and thus that a routing exists. Before aggregating aircraft, this disadvantage was irrelevant, because each aircraft was represented by an own commodity. The routes could implicitly be determined from the list of assigned arcs. When aggregating aircraft, we can only determine the assignment of activity and ground arcs to aircraft groups. We thus have to transfer the creation of the tail-based routings to the subproblem. In the literature, an Eulerian tour algorithm is commonly used to derive a routing sequence [50, 51]. We would like to employ our priority heuristic from above. The heuristic can quickly derive a routing sequence for the aircraft in multi aircraft groups.



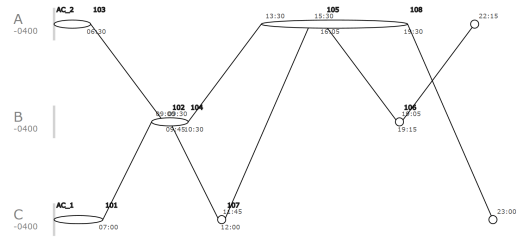
(a) Initial time-space networks of example schedule (left: aircraft 1, right: aircraft 2)



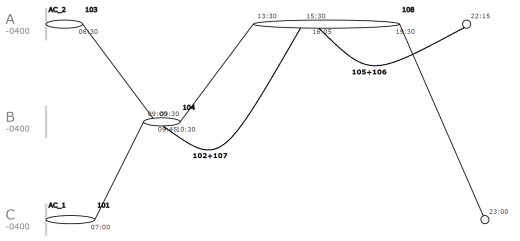
(b) Time-space network after aggregating aircraft



(c) Time-space network after applying node aggregation



(d) Time-space network after eliminating unnecessary ground arcs



(e) Time-space network after aggregating activities

Figure 7.7: Illustration of model enhancements

Dynamic disaggregation: A challenging question is how to handle infeasibilities when aggregating aircraft. In general, we apply the same iterative procedure of manually incorporating maintenance checks, analyzing conflicts, adding new restrictions to the RMP and resolving. However, this process requires slight modifications depending on whether the conflicts occur on the route of an aircraft that has its own aircraft group or is member of a multi aircraft group. In the first case, the procedure is equivalent to above. We first determine the forbidden sequence and then add an additional constraint to the RMP that restricts this sequence from recurring. In the second case, the procedure is slightly different. We first have to remove the affected aircraft from its original aircraft group and create a new single aircraft group. Then, we can proceed with the same steps as above. This procedure implies that the number of commodities is continuously enlarged during the iterations.

Cut selection: Creating additional aircraft groups increases the size and slows down solving times of the RMP. We therefore preferentially create Benders cuts for single aircraft groups when encountering multiple infeasible routes. Only when all conflicts occur on routes of aircraft that are member of multi aircraft groups, we select one of these conflicts to block in subsequent iterations.

In our example schedule, the networks are identical for both aircraft. The aircraft have neither preassigned nor forbidden activities. Therefore, we can aggregate them in a single commodity and use a common network as shown in Fig. 7.7b.

When running benchmarks on all 220 instances, we observe that the initial number of commodities is reduced for all instances. The average reduction amounts to impressive 91 percent. The largest reduction is achieved for instance 05-738 where a total of 186 aircraft are aggregated in a single group. The enhancement is less powerful when many aircraft have preassigned or forbidden activities.

Table 7.5: Evaluation of aircraft aggregation enhancement

	Base model					**Base model + aircraft aggregation**				
Instance	Init. ac groups	Init. variables	Init. constr.	Itera-tions	Time [s]	Init. ac groups	Init. variables	Init. constr.	Itera-tions	Time [s]
03-767x	65	68,641	47,520	4	527.1	2	2,175	1,845	1	0.4
05-737	186	1,412,485	1,458,477	-	-	1	7,779	10,482	-	-
07-757	133	828,192	898,185	-	-	1	6,359	8,901	1	3.1
28-777	55	255,696	173,021	-	-	1	4,703	4,703	1	4.8
03-380x^{t}	31	16,212	11,351	2	8.2	5	2,643	1,995	1	0.4
05-380x	55	70,566	48,479	-	-	4	5,183	3,956	10	49.3
10-330/340x^{t}	29	63,598	43,465	2	1519.5	12	26,337	18,428	4	726.9
10-777x	28	56,253	38,432	-	-	4	8,060	6,080	3	20.0

The enhancement does not only affect the network topography, but also the solving performance of the Benders framework. We notice a large average reduction in solving times of 94 percent. The performance is improved for all instances, which suggests that aircraft aggregation is a valid enhancement for diverse types of schedules. The improvement ranges from a minimum of 2 percent to a maximum of 99 percent. When aggregating aircraft, we are able to solve 198 of our 220 instances within the time limit of 30 minutes. The problem and solving characteristics of our eight example instances are shown on the right hand side of Tab. 7.5.

Node aggregation

The size of the RMP can be further reduced by applying a preprocessing method called node aggregation, which was first proposed by Hane et al. in the context of fleet assignment [31]. In our base model, every activity is represented by an

individual departure and arrival event. Consequently, the networks consist of a large number of nodes. Since the exact event times are irrelevant as long as the order is correct, we can combine consecutive arrival and following consecutive departure events in a single node. In doing so, we do not only reduce the amount of nodes, but also eliminate interlinking ground arcs. The number of nodes is at least halved, because we can at least combine an arrival event with a departure event.

Fig. 7.7c shows the time-space network of our example schedule after aggregating nodes. For this instance, the number of nodes is reduced by 56 percent from 18 to eight. Measured over all 220 instances, we notice a mean reduction of 64 percent.

Table 7.6: Evaluation of node aggregation enhancement

	All prior enhancements					**All prior enhancements + node aggregation**				
Instance	Init. nodes	Init. variables	Init. constr.	Itera-tions	Time [s]	Init. nodes	Init. variables	Init. constr.	Itera-tions	Time [s]
03-767x	1,450	2,175	1,845	1	0.4	439	1,164	834	1	0.3
05-737	5,124	7,779	10,482	-	-	1,571	4,226	5,073	7	257.5
07-757	4,195	6,359	8,901	1	3.1	1,254	3,418	4,122	1	2.4
28-777	3,117	4,703	4,703	1	4.8	820	2,406	2,406	1	4.3
03-380x^{t}	1,799	2,643	1,995	1	0.4	663	1,507	859	1	0.3
05-380x	3,492	5,183	3,956	10	49.3	1,280	2,971	1,744	3	5.2
10-330/340x^{t}	17,672	26,337	18,428	4	726.9	6,330	14,995	7,086	2	192.2
10-777x	5,392	8,060	6,080	3	20.0	1,920	4,588	2,608	7	58.2

Node aggregation is commonly used in the literature to reduce the size of time-space networks. In Tab. 7.6, we compare problem and solving characteristics before and after applying node aggregation to our example instances. Across all 220 instances, we measure a mean reduction in solving times of 33 percent. We are able to solve 202 instances within the time limit. The large majority of 187 instances follow the prediction from literature that node aggregation is capable of speeding up the solving process. Surprisingly, we also encounter 19 instances that seem to be negatively affected by node aggregation. For example, we observe an increase in solving time of 191 percent for instance 10-777x (see Tab. 7.6). The change in solving times ranges from a maximum reduction of 99 percent to a maximum increase of 756 percent. We conclude that this phenomenon is explained by coincidence and random factors. To be able to reproduce our results, we have eliminated all non-deterministic elements from our algorithms and set a random seed. However, this procedure does not prevent us from being exposed to some “lucky” or “bad” permutations.

Island isolation

Island isolation is another preprocessing method proposed by Hane et al. It describes the elimination of unnecessary ground arcs in the time-space networks [31].

Since we know the number of aircraft at each airport at the beginning of the planning horizon and we assume that all activities have to be performed, we can compute the number of aircraft on ground at each airport at any time. Whenever this number equals zero, we can eliminate the corresponding ground arc in the networks.

In Fig. 7.7d, we show how the time-space network of our example schedule looks after eliminating unnecessary ground arcs. On average, 66 percent of the ground arcs in the networks of our 220 instances are eliminated by this enhancement.

Table 7.7: Evaluation of island isolation enhancement

	All prior enhancements					**All prior enhancements + island isolation**				
Instance	Init. gr. arcs	Init. variables	Init. constr.	Itera-tions	Time [s]	Init. gr. arcs	Init. variables	Init. constr.	Itera-tions	Time [s]
03-767x	439	1,164	834	1	0.3	223	1,164	1,050	1	0.2
05-737	1,571	4,226	5,073	7	257.5	1,045	4,226	5,599	7	266.0
07-757	1,254	3,418	4,122	1	2.4	871	3,418	4,505	1	2.3
28-777	820	2,406	2,406	1	4.3	496	2,406	2,730	1	4.3
03-380x^{t}	663	1,507	859	1	0.3	193	1,507	1,329	1	0.2
05-380x	1,280	2,971	1,744	3	5.2	488	2,971	2,536	9	11.5
10-330/340x^{t}	6,330	14,995	7,086	2	192.2	1,998	14,995	11,418	2	8.4
10-777x	1,920	4,588	2,608	7	58.2	600	4,588	3,928	2	2.2

Concerning the solving times, we measure a mean reduction of 21 percent. In total, we are now able to solve 207 of our 220 instances within the time limit. We observe an improved solving performance for 124 instances. For 12 instances, we do not observe a change in performance and for 72 instances, longer solving times are measured. Tab. 7.7 shows problem and solving characteristics of our eight example instances before and after this enhancement. We notice that the initial number of variables is unchanged and the initial number of constraints is increased by this preprocessing method. This unexpected change is due to our implementation procedure. We first create the entire networks including all ground arcs. We then add additional constraints to the model that force the variables of invalid ground arcs to be zero. Our implementation could be improved by not creating invalid ground arcs in the beginning. In consequence, the number of variables should decrease while the number of constraints should remain unchanged.

Aggregation of activities

The next logical step is to aggregate activities. When analyzing the structure of our time-space networks, we identify nodes where preceding and succeeding ground arcs have been eliminated, and that only consist of a single arrival and departure event. Given this structure, it is obvious that the departing activity from that node

has to be assigned to the same aircraft as the arriving activity. Due to the lack of ground arcs, there is no other aircraft available than the one performing the arriving activity. By aggregating both activities in one single activity arc, starting at the departure event of the first activity and ending at the arrival event of the second activity, we can reduce the number of arcs in our networks.

We measure an average reduction in number of scheduled activities of 38 percent. The enhancement is especially powerful for schedules that include many forced turns at spoke stations (e.g., 10-330/340x^{t}). Schedules of fleets that serve major destinations or connect different hubs, in contrast, are less likely to be improved (e.g., 07-757). Fig. 7.7e shows the adjusted network for our example schedule.

Table 7.8: Evaluation of activity aggregation enhancement

	All prior enhancements					**All prior enhancements + activity aggregation**				
Instance	Activities	Init. variables	Init. constr.	Iterations	Time [s]	Activities	Init. variables	Init. constr.	Iterations	Time [s]
03-767x	725	1,164	1,050	1	0.2	647	1,086	1,011	1	0.2
05-737	2,655	4,226	5,599	7	266.0	2,468	4,039	5,412	1	14.4
07-757	2,164	3,418	4,505	1	2.3	2,083	3,337	4,424	1	2.3
28-777	1,586	2,406	2,730	1	4.3	1,460	2,280	2,604	1	1.5
03-380x^{t}	844	1,507	1,329	1	0.2	499	1,162	1,260	1	0.2
05-380x	1,691	2,971	2,536	9	11.5	991	2,271	2,361	6	5.3
10-330/340x^{t}	8,665	14,995	11,418	2	8.4	4,453	10,783	11,067	1	3.8
10-777x	2,668	4,588	3,928	2	2.2	1,392	3,312	3,609	7	11.3

Aggregating activities supports us to solve 208 of our 220 instances within the time limit. We measure a mean reduction in solving time of six percent. For the majority of 144 instances, we observe a reduction in solving time. But unexpectedly, there is also a large group of 63 instances for which aggregating activities leads to longer solving times. We can only explain this result by non-deterministic processes in the MIP solver.

Objective function

During our result analysis, we discovered that the utilization rate of the commodities is widely scattered. While some aircraft groups are characterized by a high ratio of flying time to disposable time, the aircraft of other groups spend much time on ground. For the aircraft of aircraft groups with a high utilization, it is less likely that sufficient ground time is available to incorporate maintenance checks. So far, the RMP is solved without knowing that we will try to incorporate maintenance checks in the ground periods of the routes. We realized that an objective function in the time-space model might support us in balancing the utilization and thus distribute the available ground times evenly over the aircraft groups.

We consider three different objective functions. All of them reward to equalize utilization. The first minimizes the maximum utilization over all groups. The second maximizes the minimum utilization over all groups. And the third minimizes the difference between maximum and minimum utilization over all groups.

Let D_g be the sum of the disposable times of the aircraft in aircraft group g. The parameter dur_i indicates the flying time of activity i. We introduce the new decision variables ℓ which represents the minimum ratio of flying time to disposable time of all aircraft groups and u which represents the maximum ratio. Consequently, the three objective functions can be formulated as $min\, u$,$max\, \ell$ and $min\, u - \ell$. The flight time utilization of an aircraft group $FTU(g)$ is calculated according to equation 7.9.

$$FTU(g) = \frac{1}{D_g} \cdot \sum_{\langle i,g,dep,arr\rangle \in O} dur_i \cdot x_{i,g,dep,arr} \tag{7.9}$$

We add constraints 7.10 - 7.11 to the RMP to measure the maximum and minimum utilization over all groups.

$$u \geq FTU(g) \qquad \forall g \in G \tag{7.10}$$

$$\ell \leq FTU(g) \qquad \forall g \in G \tag{7.11}$$

In our test runs, we discovered that the MIP solver requires much time to find the optimal solution to these problems. Near optimal solutions, on the other hand, are found quickly. We therefore set a gap limit of ten percent and implement two additional constraints that provide an upper/ lower bound on our decision variables. By telling the solver that the minimum/ maximum utilization cannot be higher/ lower than the average utilization measured over all groups, we help the solver to quickly find appropriate dual bounds and assess the optimality gaps (7.12/ 7.13).

$$u \leq \frac{\sum\limits_{i \in A} dur_i}{\sum\limits_{g \in G} D_g} \tag{7.12}$$

$$\ell \geq \frac{\sum\limits_{i \in A} dur_i}{\sum\limits_{g \in G} D_g} \tag{7.13}$$

The gap limit and the bounds perform satisfactorily when minimizing the maximum utilization and maximizing the minimum utilization. However, when minimizing the difference between maximum and minimum utilization, the solver still shows difficulties finding appropriate dual bounds. We thus narrow our evaluation on the first two objective functions. We compare their performance on our 220 instances to find out whether the implementation of an objective function is beneficial for us.

In both cases, the number of variables is increased by a single variable (either the maximum utilization u or the minimum utilization ℓ). The additional number of initial constraints equals the number of initial aircraft groups (constraints 7.10 or 7.11) plus one (constraint 7.12 or 7.13).

Table 7.9: Evaluation of possible objective functions

	All prior enhancements				**All prior enhancements + objective function**					
							Minimize max util		Maximize min util	
Instance	Init. variables	Init. constr.	Itera-tions	Time [s]	Init. variables	Init. constr.	Itera-tions	Time [s]	Itera-tions	Time [s]
03-767x	1,086	1,011	1	0.2	1,087	1,014	1	0.3	1	0.3
05-737	4,039	5,412	1	14.4	4,040	5,414	1	13.1	1	13.4
07-757	3,337	4,424	1	2.3	3,338	4,426	1	2.3	1	2.3
28-777	2,280	2,604	1	1.5	2,281	2,606	1	1.6	1	1.5
03-380x^{t}	1,162	1,260	1	0.2	1,163	1,266	1	0.3	1	0.3
05-380x	2,271	2,361	6	5.3	2,272	2,366	1	0.9	1	1.0
10-330/340x^{t}	10,783	11,067	1	3.8	10,784	11,080	4	27.5	3	19.2
10-777x	3,312	3,609	7	11.3	3,313	3,614	1	1.4	3	6.1

The computational results for our eight example instances are shown in Tab. 7.9.

With respect to all 220 instances, the usefulness of this enhancement is hard to evaluate. On one hand, we observe that an equalized utilization support us in solving more instances within the 30 minutes time limit. When minimizing the maximum utilization, we are able to solve 213 instances, when maximizing the minimum utilization 211 instances. On the other hand, we measure an average increase in solving times for both approaches. When minimizing the maximum utilization, the average increase amounts to 24 percent. The solving performance is improved for only 51 instances. For 160 instances, we detect a worse performance. When maximizing the minimum utilization, the average increase in solving time amounts to 22 percent. We observe improvements for 44 and setbacks for 170 instances.

The results suggest that the first objective performs slightly better. For a few instances, this enhancement is very helpful. For example, the solving time for instance 10-777x is reduced by 88 percent. But, for the majority of instances, we observe a negative effect. For instance 10-330/340x^{t}, we measure an increase in solving time of 632 percent. We therefore drop the idea to implement an objective function in the RMP.

Generation of Benders cuts

Another idea to improve the row generation process is to implement a different strategy of how to add Benders cuts to the RMP. As described above, we currently select a single conflict and solely add a single Benders cut to the RMP in every iteration. Alternatively, we could implement a strategy of adding multiple Benders cuts to the RMP in every iteration.

We developed two ideas. First, and most straightforward, we could add Benders cuts for every conflict we identify. In doing so, we cut off as many invalid solutions from the solution space of the RMP as possible in every iteration. The drawback of this approach is that we eventually create many new aircraft groups for aircraft that were pooled in multi aircraft groups before. We then quickly return to a multi-commodity formulation, in which each aircraft is represented by an own commodity.

The second idea is somewhere in the middle between the first idea and our current strategy. To avoid opening multi aircraft groups, the strategy is to solely add the constraints blocking all conflicts in single aircraft groups in an iteration. In other words, if we encounter multiple infeasible routes in single aircraft groups in an iteration, we add multiple Benders cuts to restrict all these conflicts. In case there are only infeasibilities in multi aircraft groups, the strategy is to select a single conflict, create one additional aircraft group and add only one Benders cut to the RMP.

Table 7.10: Evaluation of strategies to generate Benders cuts

	single conflict per iteration		**all conflicts per iteration**		**all single ac group conflicts per iteration**	
Instance	Iterations	Time [s]	Iterations	Time [s]	Iterations	Time [s]
03-767x	1	0.2	1	0.2	1	0.2
05-737	1	14.4	1	13.7	1	12.7
07-757	1	2.3	1	2.3	1	2.4
28-777	1	1.5	1	1.7	1	1.6
03-380x^{t}	1	0.2	1	0.2	1	0.2
05-380x	6	5.3	2	1.3	6	5.1
10-330/340x^{t}	1	3.8	1	3.5	1	3.6
10-777x	7	11.3	8	15.1	7	11.3

Tab. 7.10 shows the required numbers of iterations and solving times of our eight example instances given the three strategies of how to generate Benders cuts. When running benchmarks on all 220 instances, we see that blocking all conflicts encountered in an iteration performs worst. Although, the average solving time is slightly improved compared to our currently implemented strategy (-1 %), we are only able to generate initial solutions for 202 of our 220 instances. For 106 instances, we see improvements in solving time, but for 96 instances, the performance is worse.

The last strategy, in contrast, performs best. When adding multiple Benders cuts to block all infeasibilities in single aircraft groups, the average solving time is reduced

by five percent compared to our current strategy. We are still able to solve 208 of the 220 instances within the time limit of 30 minutes per instance. For the majority of 128 instances, we measure improvements in solving performance. We thus replace our currently implemented strategy with the last strategy.

7.1.4 Conclusion

In this section, we have analyzed three ideas of how to generate a feasible initial solution for the tail assignment problem. Using a MIP solver to solve the main model as a feasibility problem requires the least implementation effort. But, for medium and large problem sizes, it is a very time-consuming approach. Heuristics are promising in terms of required time. Their disadvantage is that they have to be tailored to the specific problem. We have seen that it is challenging to create a heuristic that always finds a feasible solution. The iterative Benders framework circumvents this disadvantage as it guarantees that a feasible solution is found if any exists. Although the time requirement is generally longer than when using a pure heuristic approach, we are able to generate feasible solutions for nearly all instances within the time limit. Compared to the required time when using the first approach, the Benders framework is far superior. Therefore, it is our methodology of choice for the initial solution stage.

We have shown how preprocessing is capable of reducing problem sizes and solving times. We use aircraft aggregation, node aggregation, island isolation and activity aggregation to enhance the solving performance. In Sec. 7.2, we describe the ALNS framework we have designed to iteratively improve the generated initial solution.

7.2 Improvement stage

The improvement stage is the core of every LNS algorithm. Essential elements are the destructor and repair methods, the acceptance rule and the stopping criterion. In our approach, the improvement stage is embedded in an adaptive framework, allowing us to implement several competing destructor methods.

7.2.1 Destructor methods

The ideas for our destructors originate from observations we made when analyzing routing plans. Most of them are based on intuitive ideas of how to improve an existing routing. Some are also based on the proposals of Kabbani and Patty [44]. The destructors are designed so that they release different parts of the assignments. Some destructors consider the routes of many aircraft, but focus on a small time window or a limited number of stations. Other destructors take a more global

perspective and seek to open larger time windows, but on a limited number of routes. All of our destructors contain some element of randomness to vary the released assignments in every call. In addition, we implement a tabu list containing the information about opened assignments in all past iterations to make sure that a unique set of assignments is released in every iteration.

Single route destructor

The need for the single route destructor arises from the way we construct our initial solution. Since we place a maintenance check in every possible ground period, the initial routes of the aircraft are packed with maintenance checks. Most of these checks are not required for the routing to be feasible. The purpose of the single route destructor is to eliminate all superfluous checks in the route of an aircraft. When it is called, a specific tail is chosen randomly. The routes of all other aircraft are locked. In consequence, the destructor is incapable of performing changes in the flying sequences. We then identify the set of all arcs connecting any pair of neighboring activities in the route of the selected aircraft and release them for reoptimization. As result, activities that were formerly connected by a costly maintenance arc might now be connected by a direct connection arc or a cheaper maintenance arc.

We realized that this destructor is very useful to "clean up" the initial routes of the aircraft. We therefore decided to incorporate a preliminary improvement stage. We refer to this stage as *maintenance reduction stage*. In a loop, we call the single route destructor for every tail to remove all superfluous checks from the initial routes. Afterwards, we start the actual improvement stage in which we do not implement the single route destructor anymore.

Multi route destructor

The multi route destructor is an advancement of the single route destructor. Instead of solely open the route of a single tail, this destructor allows the routes of several tails to be reconfigured. At first, we identify all aircraft which have maintenance slack in the current solution. Maintenance slack means that checks are performed before the interval limits are reached. We then randomly select one of these aircraft. Next, we determine all aircraft that cross the route of the selected aircraft. By crossing a route, we mean that the aircraft share common ground times at a location. Out of the potential partner aircraft, we select either one or two in a random way. Then, we release the assignments of all arcs connecting any pair of activities in the routes of the selected aircraft for reoptimization.

This destructor allows swaps in the routes of the selected aircraft during the entire planning horizon. Also, maintenance activities can be inserted or removed. The routes of the unselected aircraft, on the other hand, remain unchanged.

Overnight destructor

The overnight destructor follows the idea of Kabbani and Patty to swap routings of two aircraft which are overnighting at the same location [44]. They refer to this swap as changing termination-origination connections (T-O swap). Especially for short-haul fleets, flights are usually carried out during the day and maintenance is performed at night. We observe though that checks are often performed too early, because the particular aircraft is not overnighting at an appropriate maintenance station in the required night. We therefore extend the idea of Kabbani and Patty.

The two aircraft are selected so that the route of one involves a check during the night which is performed before the interval limits are met. The partner aircraft is selected so that its routing ends at a maintenance base in the evening of the next day (see Fig. 7.8). We then disconnect the ground arcs of the aircraft during the selected night and try to reconnect them in a better way. The routes before and after the common ground time are not allowed to be changed. However, as the latter might be assigned to the other tail, we allow maintenance activities to be rescheduled.

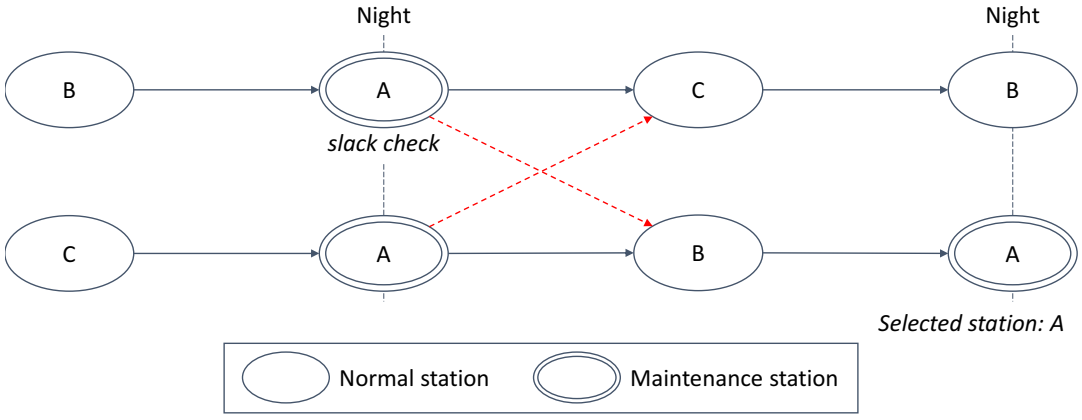


Figure 7.8: Overnight destructor [44]

Route swap destructor

The route swap destructor is a generalization of the overnight destructor. Instead of allowing the connections to be swapped during nighttime, we now allow the routings of multiple aircraft to be swapped at a randomly chosen time and station. The combination of time and station is selected so that at least two aircraft are simultaneously on ground in the current solution. Otherwise, it would not be possible to swap any connections and the iteration would be useless. We then allow the routes of all aircraft which are on ground at the particular station at the selected time to be swapped (see Fig. 7.9). The routes prior and after the swap are not allowed to be changed, only maintenance activities can be rescheduled. Unlike the overnight destructor, this destructor does not pose additional requirements on the selected aircraft. Kabbani and Patty refer to this idea as turn swap or through swap [44].

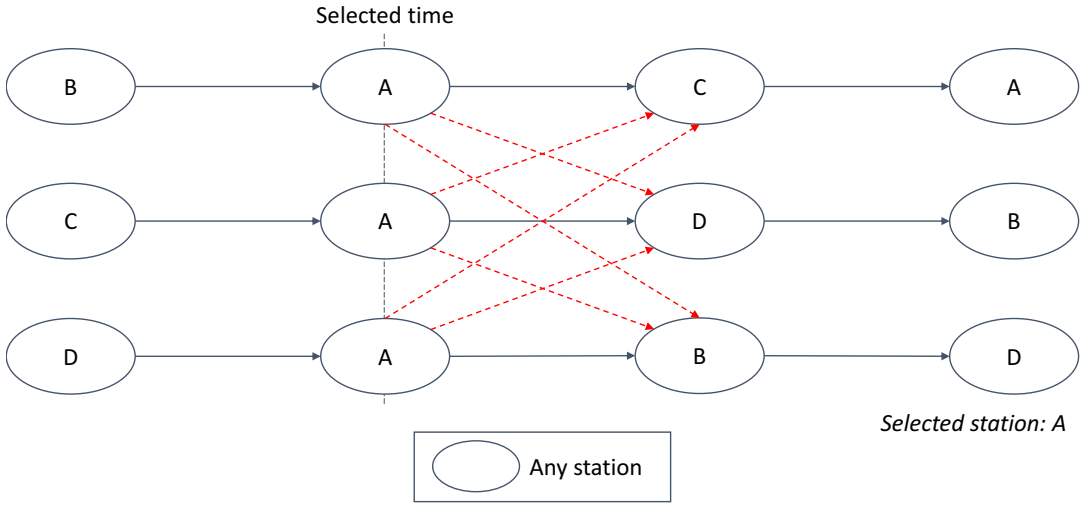


Figure 7.9: Route swap destructor [44]

Time-space destructor

The time-space destructor follows a different strategy. Instead of releasing assignments at a fixed point in time, we now release all connections at a station within a larger time window (see Fig. 7.10). The appropriate length of the time window depends on the instance at hand. We implement three different possibilities, i.e., six, twelve and 24 hours. The starting time, the length of the time window as well as the station are chosen randomly. At least three different aircraft have to perform a connection within the time window at the selected station. Otherwise, the selection process is renewed. We do not set an upper limit of affected aircraft. The routes before and after the time window are not allowed to be changed, but the latter might be reassigned to a different tail and maintenance activities rescheduled.

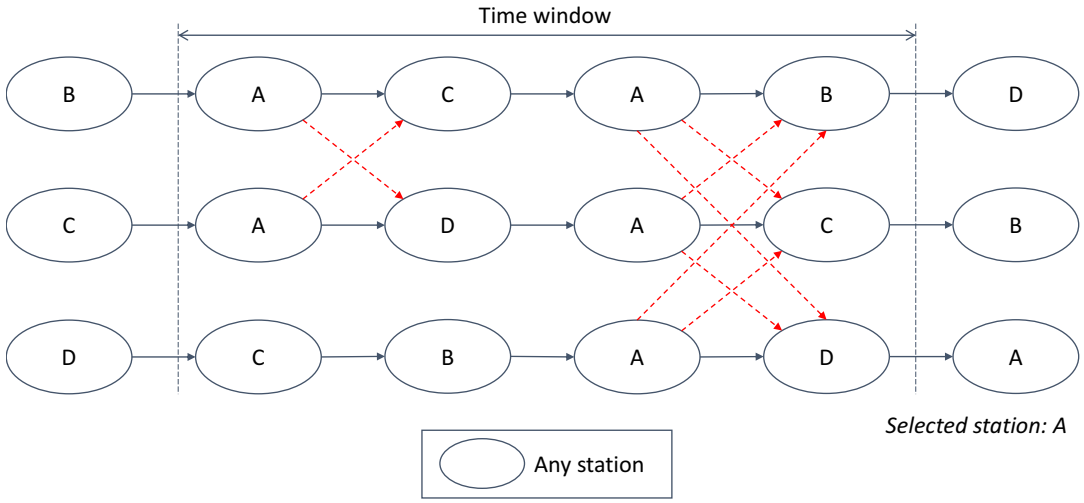


Figure 7.10: Time-space destructor

Utilization balancing destructor

LNS sometimes works best when not doing the most intuitive things. The utilization balancing destructor works similar to the multi route destructor. The difference lies in the way we select the aircraft. When calling this destructor, we randomly choose one of the aircraft with the ten percent highest utilization and one of the aircraft with the ten percent lowest utilization. The utilization is defined as the ratio of flying time to disposable time. We release the entire routes of the two aircraft for reoptimization, allowing swaps at any time during the planning horizon.

7.2.2 Repair method

To repair the subproblems, we reoptimize the decomposed main model using a MIP solver. Since the destructors only release a small fraction of variables for reassignment, a MIP solver is capable of repairing the subproblems in acceptable time frames. We provide the old routing as a warm start solution. The search for improving neighbors is limited by an iteration time limit between four and 20 seconds depending on the instance size. We thereby force the algorithm to perform many iterations in short time and search many promising areas of the solution space.

7.2.3 Acceptance rule

Our acceptance rule is simple. Since we use a MIP solver to repair the subproblems and provide the incumbent solution as warm start, we only generate solutions of equal or improved quality. We therefore chose to accept any unvisited solution.

7.2.4 Stopping criterion

We implement multiple stopping criteria to terminate the destroy-and-repair iterations. As it is unknown beforehand how long it takes to generate an acceptable solution, it is insufficient to only implement an upper time and iteration limit. We designed a third criterion which is independent of the instance at hand. It is based on the improvement rate, defined as the ratio of improvement in objective value to number of iterations. In the beginning of a search, this rate is high as there are many improving neighbors. With increasing number of iterations, the improvement curve flattens and the improvement rate declines. Our idea is to stop the search when the most recent iterations have shown no or only marginal changes in objective value.

In our tests, we stop the iterations when less than one percent improvement in objective value has been achieved in the past 25 iterations. Besides, we implement an upper time limit of 30 minutes and an iteration limit of 500. These values appeared to be reasonable in our test runs and also appropriate considering the time availability of airlines. Whichever criterion is first met terminates the search.

7.2.5 Adaptive framework

Our LNS approach includes multiple destructor methods $d \in D$, but only a single repair method. Therefore, we solely need to track the performance of the destructor methods. In the initial state, all destructors are characterized by an equal weight, which is calculated according to equation 7.14.

$$w_d = \frac{1}{|D|} \qquad \forall d \in D \tag{7.14}$$

The weight w_d of destructor d equals its selection probability. Ropke and Pisinger propose to divide the search into segments and update the weights of the competing methods at the end of each segment [61]. We propose a different paradigm. As the search is a continuous process, we believe that it should not be discretized into segments. Instead, we adjust the weights of the destructors after every iteration.

The used destructor method is rewarded according to its performance. If it yields a new global best solution, a rewarding score of $s = \sigma_1$ is granted. A destructor that finds a non-improving but unvisited solution is rewarded with $s = \sigma_2$. A destructor that only finds a known solution is penalized with $s = \sigma_3$. The new weight w_x of the used destructor x is calculated according to equation 7.15. Similar to Pisinger and Ropke, we introduce a reaction parameter $r \in [0, 1]$ which controls how sensitive the weights follow the recent performance [58]. The weights of the destructors that have not been used in the current iteration remain unchanged.

$$w_x := w_x(1 - r) + rs \qquad \textit{if } x \textit{ was used} \tag{7.15}$$

Afterwards, we normalize all weights so that their sum equals 1 (7.16). In doing so, we can directly derive the new selection probabilities from the weights. To do justice to the fact that the suitability of a destructor might change during the search, we reset the weights of all destructors to their initial values every 25 iterations.

$$w_d := \frac{w_d}{\sum\limits_{d \in D} w_d} \qquad \forall d \in D \tag{7.16}$$

Parameter tuning

We employ a similar parameter tuning approach as Ropke and Pisinger to assign values to the parameters r, σ_1, σ_2 and σ_3 [61]. At first, we assign a trial value to each parameter that appears to be reasonable. Afterwards, we improve the initial assignments by systematically varying the value of one parameter while fixing all

other parameters at their current values. We define candidate values for each parameter and run benchmarks. The best candidate value is determined by assessing the average deviation from the best known solution. We then fix the parameter at this value and move on to the next parameter. We continue the process until we have optimized all parameters. Ropke and Pisinger note that it would be possible to repeat this process several times using the final results of the preceding run as initial values in the succeeding run. However, like Ropke and Pisinger, we refrain from doing this in our tuning [61].

To determine the best candidate value for a parameter, we assess both the deviation in time and quality from the best known solution. The optimal parameter settings are subject to the preferences of the user. Some users are more focused on time, others on quality. We define two user profiles. The preferences for solution time and quality are indicated by different weights. The first user has a high preference for time. The weights in this profile are 0.8 for time and 0.2 for quality. The second profile is defined complementary, using weights 0.2 for time and 0.8 for quality. With these weights, we can aggregate the deviation in time and quality from the best known solution and determine the overall best candidate value for each profile.

For both profiles, we start with the same set of trial values. We use the vector $(r, \sigma_1, \sigma_2, \sigma_3) = (0.1, 0.3, 0.1, 0.0)$, which is derived from the tuning results of Ropke and Pisinger [61]. Also, the candidate values are identical (Tab. 7.11). The only difference lies in the way we aggregate the deviation in time and quality from the best known solution. For the time-sensitive user, we place a higher weight on the deviation in time, for the quality-sensitive user on the deviation in quality.

Table 7.11: Candidate values for the ALNS parameters

Parameter	Candidate values		
r	0.1	0.3	0.5
σ_1	0.3	0.4	0.5
σ_2	0.1	0.2	0.3
σ_3	0.0	-0.1	

Example: Let us illustrate the process of determining the optimized parameter values with an example. Tab. 7.12 shows the duration of the improvement stage and final objective value of our eight example instances given the three different candidate values for the reaction parameter r. As r is the first parameter that we optimize, the table is identical for both user profile. The best found time and objective value for each instance is highlighted using a bold font. In the last row of Tab. 7.12, we indicate the average deviation in time and quality from the best solution for each r. We now use the weights of the user profiles to determine the aggregated deviation, which is shown in Tab. 7.13. The minima in the rows of Tab. 7.13 reveal the optimized value of the reaction parameter. A time-sensitive user prefers a reaction factor of $r = 0.1$, a quality-sensitive favors $r = 0.3$.

Table 7.12: Solving times and objective values given different reaction factors r

	Solving time [s]			**Final objective value**		
Instance	$r = 0.1$	$r = 0.3$	$r = 0.5$	$r = 0.1$	$r = 0.3$	$r = 0.5$
03-767x	**234**	435	267	1,489,659	**1,307,760**	1,407,920
05-737	**185**	214	298	500,715	485,971	**466,190**
07-757	1,095	**1,008**	1,053	3,933,895	**3,928,002**	3,937,980
28-777	**1,000**	1,377	1,376	1,942,965	1,706,680	**1,706,115**
03-380x^{t}	759	**730**	775	3,512,704	3,520,273	**3,510,846**
05-380x	3,590	**3,327**	3,527	2,723,233	**2,647,667**	2,676,390
10-330/340x^{t}	**1,592**	1,598	1,607	2,396,427	**2,267,670**	2,329,131
10-777x	**1,345**	1,657	1,497	6,331,917	**6,231,717**	6,261,513
Average deviation	0.0256	0.2036	0.1771	0.0569	0.0057	0.0152

Table 7.13: Aggregated deviation for both user profiles

	Aggregated deviation		
	$r = 0.1$	$r = 0.3$	$r = 0.5$
Time-sensitive user	**0,0319**	0,1640	0,1447
Quality-sensitive user	0,0507	**0,0453**	0,0476

□

We present the optimized values for both user profiles in Tab. 7.14. We see that the quality-sensitive user generally exploits the benefits of the adaptive framework more explicitly. The user places a higher weight on the current performance and his rewarding scheme is more pronounced. Especially, a higher reward is granted to methods that find unvisited solutions (σ_1, σ_2), because they diversify the search and promote to find better solutions in the long run. The time-sensitive user pursues in some sense a counterproductive strategy. This user is not interested in placing high weights on successful or diversifying methods. In doing so, the search is terminated earlier, as our stopping criterion sets in when the improvement rate falls below a threshold. Obviously, this behavior does not correspond to our interests. We therefore assume a quality-sensitive user for our computational study in Chap. 8.

Table 7.14: Optimized parameter values for ALNS framework

	Optimized values			
Parameter	r	σ_1	σ_2	σ_3
Time-sensitive user	0.1	0.3	0.1	0.0
Quality-sensitive user	0.3	0.5	0.3	-0.1

During our test runs, we noticed that the parameter tuning is subject to random factors. When running the same settings twice, we did not necessarily obtain the same results. This is due to the stochastic elements in the ALNS algorithm. Nevertheless, the general trend of our conclusions could be confirmed.

8 Analysis and evaluation

Our computational study is divided into two parts. First, we describe how we obtained the data for our test instances. In the second part, we analyze and evaluate the performance of our adaptive large neighborhood search algorithm.

8.1 Generation of test instances

To prove the performance of our approach, the schedules of two major international airlines are at our disposal. As the provided data sets from our airline partners are incomplete, a number of measures are required to obtain evaluable test instances.

The first airline (Carrier One) operates five hubs, from which it serves 275 destinations in 50 countries. On average, 3,450 flights are carried out every day using 710 aircraft of twelve different fleets. The airline provided us their complete Standard Schedule data as defined by the *International Air Transport Association* (IATA), including fleet assignment information. The second airline (Carrier Two) is a fast-growing hub-and-spoke operator. Out of a single hub, 465 daily flights to 125 destinations in 70 countries are offered. The airline owns a total of 211 aircraft of eight different fleets and 15 subfleets. This airline provided us a detailed tail assignment plan including scheduled checks for individual tails for a period of ten days. The plan has been designed using their current best practice. The data also includes information about where and when the aircraft become available. However, the data of both airlines withhold information about the flying and maintenance history of the aircraft as well as tail-dependent restrictions. We also do not know the specific maintenance programs of the airlines.

In total, we have created 220 test instances, 53 are based on the data of Carrier One and 167 on the data of Carrier Two.

8.1.1 Initial positions of the aircraft

To generate test instances, we start by extracting all flights during a selected planning period from the schedule of an airline. The data of Carrier Two already provide us with the information about the initial positions of the aircraft. The data of Carrier One is lacking this information. We also do not know how many aircraft are required to perform the schedule. However, as we assume that operations are according to

plan and all flights are carried out, we can calculate the required number of aircraft at every airport at the beginning of the planning period. We place the smallest number of aircraft at every airport so that during the entire planning horizon the aircraft flow balance never falls below zero.

8.1.2 Maintenance program and capacity

The data from our airline partners does not include any information about their individual maintenance programs. We therefore create a standard program that is based on the regulations of the FAA. We assume that there are three check types to consider, which vary in scope, frequency, duration and cost.

All aircraft are required to perform a *Daily Check* every 48 calendar hours. A more comprehensive *Weekly Check* is due every 168 calendar hours. Finally, we consider an *A-Check*. The interval limits of the A-Check are fleet-dependent and defined in terms of accumulated flight hours and cycles. We find target values in the MRBR of the fleets. For example, the A-Check for a Boeing 757 is due after 500 flight hours or 300 flight cycles [16]. We suppose an explicit check hierarchy, stating that a Daily Check is included in a Weekly Check and a Weekly Check in an A-Check.

The scheduling of less frequent, regular maintenance checks, e.g., C-Check and D-Check, is not considered in tail assignment. We model these checks as preassigned maintenance activities for individual tails. Nevertheless, the maintenance hierarchy still applies, which means that they include all hierarchically lower checks.

We estimate the durations of the checks from the tail assignment plan of Carrier Two and the experience of experts at *INFORM GmbH*. We assume that a Daily Check requires one hour of ground time, a Weekly Check two hours and an A-Check six hours. In addition, we consider buffer times of 20 minutes prior and after a check. Of course, all these parameters can easily be adjusted by the user. The check durations are also used as reference point when assigning a cost parameter to each check.

Carrier Two performs all maintenance activities at its hub. For Carrier One, on the other hand, we only know where A-Checks are performed (hangar bases). We assume that Weekly Checks are performed at the same stations. Daily Checks, in contrast, do not require extensive tools and materials. They can be done at most larger airports. For our tests, we assume that they can be performed at any hangar base but also at stations that are served at least four times a day.

We assume that Daily and Weekly Checks are performed at the ramp of the airport. Only A-Checks and eventually some prescheduled maintenance activities require a hangar. The data of Carrier One includes information on the maximum number of simultaneously performed A-Checks at the maintenance bases. We use this number as reference for available hangar positions. The data of Carrier Two does not include capacity information. We assume that the hangar space at its hub is restricted to five positions. We chose this number, because it equals the capacity at the largest station of Carrier One.

8.1.3 Initial maintenance counters

The data of both carriers withhold information about the initial maintenance counters of the aircraft. We therefore have to generate these parameters randomly. For every aircraft $t \in T$, we first determine the relevant maintenance checks M_t. Then, we create initial counters $init_{tmq}$ for all criteria $q \in Q_m$ of the determined checks $m \in M_t$.

The initial counters are randomly generated integer numbers, which are drawn from different uniform distributions. For the flight hours criterion, we use the uniform distribution $\mathcal{U}(0, max_{mq} - 25)$, for the flight cycles criterion $\mathcal{U}(0, max_{mq} - 6)$, and for the calendar hours criterion $\mathcal{U}(0, max_{mq} - 48)$. We incorporate buffers in the initial counters to guarantee that the aircraft are not stranded at their initial positions.

Also, we make sure that the generated counters are consistent. The initial Daily Check counter of an aircraft has to be equal or lower than the Weekly Check counter. If a maintenance check depends on multiple criteria, like the A-Check, the initial counters for all criteria should represent a similar flying history of the aircraft. For example, it would not make sense if the counter of the flight hours criterion is close to the allowed limit while the flight cycles counter is almost zero.

8.1.4 Preassigned activities and curfews

Creating instances with preassigned activities and curfews is risky, because a random definition might turn the problem infeasible. To guarantee solvable instances, we only use the data of Carrier Two. This airline provided us a feasible tail assignment plan. However, we do not know which activity-aircraft assignments have been obligatory. Our approach is as follows: based on the provided plan, we determine heavy maintenance checks, e.g., D-Checks, and assign them to the corresponding aircraft. We do the same for irregular maintenance activities, e.g., the repair of a seat. All other maintenance activities are not preassigned, because we want them to be scheduled by our optimizer.

The definition of preassigned and forbidden flights is done differently. We randomly select flight-aircraft assignments from the provided plan and declare them to be preassigned. Similarly, we select flight-aircraft assignments that are not part of the plan to generate curfews. We create preassigned and forbidden activities so that no more than 20 percent of the aircraft are affected in a one-week planning horizon.

8.2 Computational results

In this section, we show computational results and evaluate the performance of our ALNS algorithm. We again employ the non-commercial MIP solver SCIP 3.1.1. All coding is done using Python 3.4 programming language. The test runs are carried out on the same system as the evaluation runs described in Chap. 7.

8.2.1 Performance evaluation

We run benchmarks on all 208 instances for which we were able to generate initial solutions within 30 minutes. Tab. 8.1 shows the computational results for our eight example instances. The table shows the time requirement to create the initial solution, the required time for the improvement stage and the total solving time. Due to data handling and preprocessing, the time of the improvement stage can be longer than the time limit of 30 minutes. Tab. 8.1 also shows the effective stopping criterion.

Instead of an inexpressive objective value, we present a parameter that indicates the quality of the final solution. It specifies the timeliness of maintenance activities. If the value equals 100 percent, all checks are done right on time without wasting any of the allowed intervals. A value of 90 percent means that on average ten percent of the allowed intervals are not used. A solution quality of 100 percent is the theoretical limit which is not always achievable, e.g., for some instances, the flight schedule does not allow to perform a Daily Check after exactly 48 hours for all aircraft.

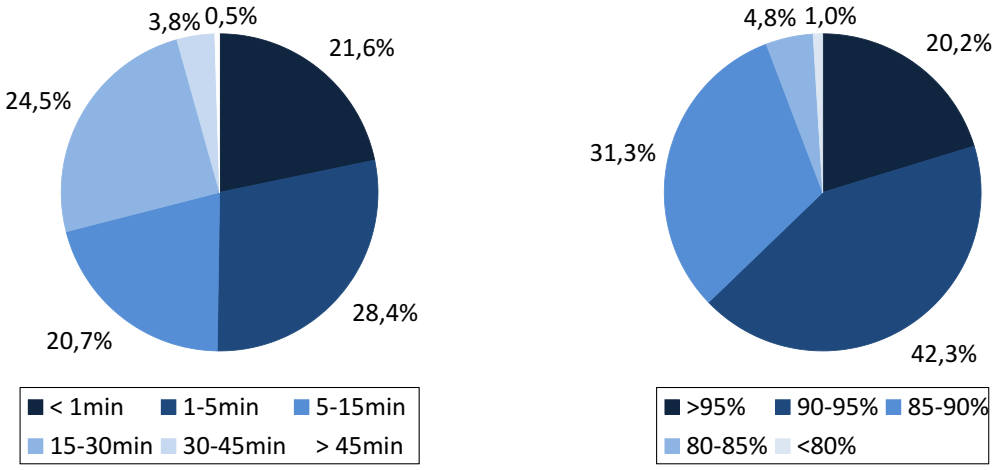
Table 8.1: ALNS solution of example instances

Instance	Init. time [s]	Impr. time [s]	Total time [s]	Stopping criterion[1]	Solution quality
03-767x	0.3	191	191	R	93,9%
05-737	14.0	1222	1,236	R	93,7%
07-757	2.4	893	895	R	91,4%
28-777	1.7	1,537	1,538	R	86,8%
03-380x^{t}	0.3	284	285	R	94,9%
05-380x	7.8	1,399	1,407	T	85,7%
10-330/340x^{t}	5.0	3,784	3,789	T	88,2%
10-777x	18.0	1,659	1,677	T	89,0%

[1] *R: rate of improvement, T: time limit*

Solving time: Our results confirm the good performance of our algorithm. The average solving time across all 208 instances is 9 minutes and 50 seconds. Fig. 8.1a shows the distribution of solving times. Half of the instances is solved in less than five minutes, more than 70 percent in less than 15 minutes and 95 percent in less than half an hour. Only instance 10-330/340x^{t} requires more than 45 minutes to be solved (see Tab. 8.1).

Solution quality: The achieved solution quality is compelling as well. The average value of the solution quality across all 208 instances is 91 percent. As Fig. 8.1b highlights, ALNS achieves solution qualities of more than 90 percent for more than 60 percent of the instances. A solution quality of at least 85 percent is achieved for nearly 95 percent of the instances. Only for two instances, the solution quality is slightly below 80 percent.



(a) Solving time **(b)** Solution quality

Figure 8.1: Performance of ALNS algorithm across 208 instances

Effective stopping criterion: As we have intended, a low rate of improvement is the effective stopping criterion for the large majority of instances (76 %). For the other instances, the iterations are stopped by the time limit criterion (16 %) or when all destructors propose to open assignments that are already included in the tabu list (8 %). The iteration limit is never the effective stopping criterion. Apparently, the limit of 500 iterations has been set too high.

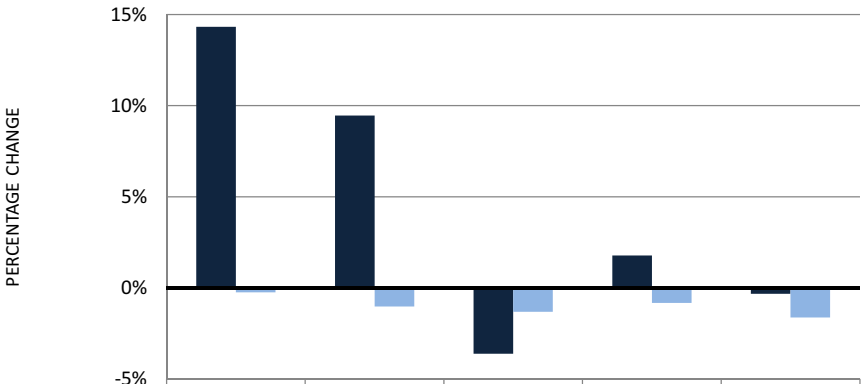
8.2.2 Destructor performance evaluation

We also seek to evaluate the performance of our destructor methods. The evaluation is challenging, because the performance depends on many random factors. The success of a particular method in an iteration is not only subject to its own performance, but also to the preceding selection order and the random choice of released assignments. We therefore can only provide a vague reference about the benefits of the different destructor methods.

We one by one deactivate one destructor method and run benchmarks with the reduced set of destructors. The assessment of the difference in achieved solution time and quality is our indicator for the benefit of a method. Fig. 8.2 illustrates our findings.

Our analysis shows that all destructors are legitimate. When deactivated, the mean solution quality decreases for all destructors. The largest decrease is measured when deactivating the overnight destructor (-1.6 %), the smallest for the time-space destructor (-0.2 %). On the other hand, the time-space destructor is capable of reducing solving times of the ALNS algorithm. When deactivated, the solving times increase by 14.3 percent. A similar benefit is observed for the utilization balancing

and multi route destructors. On average, we measure 9.5 percent higher solving times when deactivating the utilization balancing destructor and 1.8 percent when deactivating the multi route destructor. For the other two destructors, we measure a decrease in solving times when deactivated.



	no time-space	no utilization balancing	no route swap	no multi route	no overnight
Solving time	14,3%	9,5%	-3,6%	1,8%	-0,3%
Solution quality	-0,2%	-1,0%	-1,3%	-0,8%	-1,6%

Figure 8.2: Destructor benchmarking

The time-space, utilization balancing and multi route destructors improve solution quality while reducing solving times. For the other destructors, the overall evaluation is not that clear. It is subject to the preferences of the user, because they improve solution quality, but at the same time increase solving times. As our focus is mainly on solution quality, all destructor methods appear to be reasonable.

8.2.3 Comparison to exact approach

In the last part of our computational analysis, we want to benchmark the performance of our ALNS algorithm in contrast to an exact branch-and-bound approach, i.e., we solve the main MIP as an optimization problem using a MIP solver. The exact approach is capable of finding the globally optimal problem solution. As the solving times are too long to run extensive benchmarks, we focus the comparison on the smallest 24 instances having less than 50 scheduled activities.

The data points in Fig. 8.3 indicate the solving times using the ALNS algorithm and the exact approach in reference to the number of scheduled activities. We use the results of the exact approach to prove optimality of our ALNS results. Filled data points indicate proven optimal solutions. Unfilled data points are either not optimal or the exact approach was not terminated within two hours.

The chart shows that ALNS is capable of finding the optimal solution for many of the analyzed instances. For the larger instances, we are just not able to prove optimality, because the run times of the exact approach are too long.

The dashed/ dotted lines are exponential trendlines. We see that the scatterting of the data points in reference to the trendlines is quite large, especially for the exact approach. We can conclude that the number of scheduled activities is not the only determinant to increase solving times. Other factors to consider are the numbers of fleets and aircraft in the schedule as well as the number of prescheduled activities. Nevertheless, the results prove that the solving times using the exact approach rise drastically with increasing size of the schedule. The solving times using the ALNS algorithm, on the other hand, rise a lot slower. Only for instances with less than 20 scheduled activities, the exact approach is faster than the ALNS algorithm.

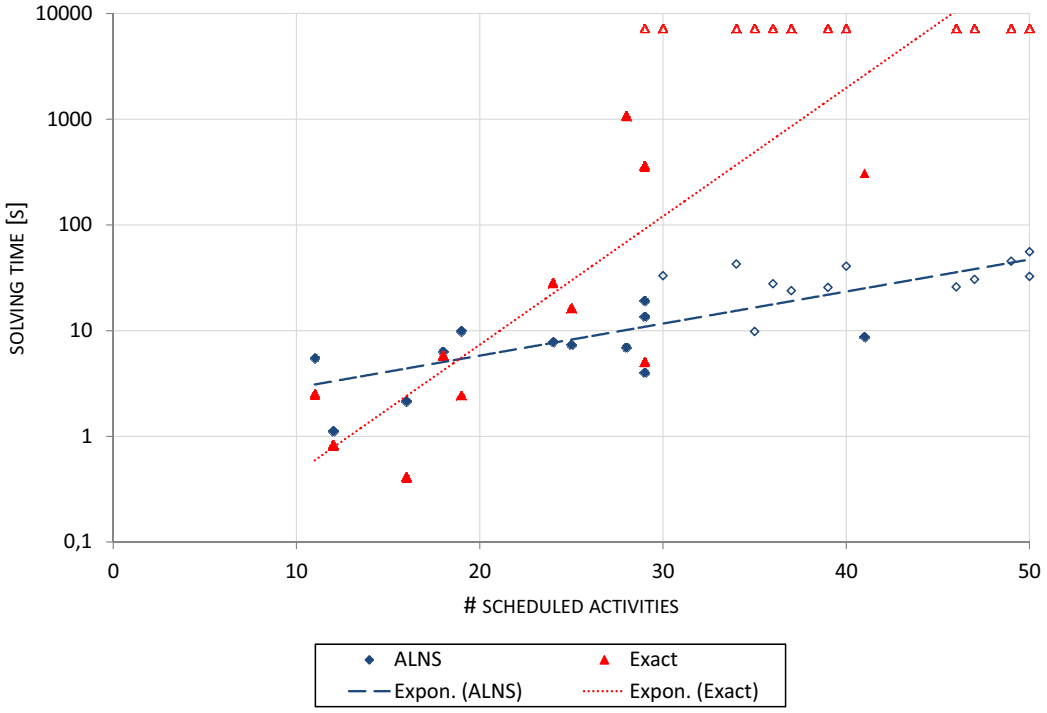


Figure 8.3: Solving times of ALNS algorithm and exact approach

The comparison in Fig. 8.3 is not entirely fair. In contrast to ALNS, the indicated times for the exact approach include the time requirement to prove optimality. Also, the exact approach provides more information, e.g., an assessment of the optimality gap. In Fig. 8.4, we provide a different comparison. We evaluate the solving time of the ALNS algorithm against the time needed for the exact approach to find a feasible solution of at least equal quality. The ratio of the required times is illustrates in the chart. A ratio below 1 means that the exact approach finds a solution that is equal or better than the final solution of ALNS faster than the ALNS algorithm. A ratio above 1 support the theory that ALNS is superior in quickly finding good solutions.

The results show that the exact approach is competitive or even superior on the smallest instances having less than 30 scheduled activities. However, these instances are not very relevant in practice. On larger instances, the ALNS algorithm outperforms the exact approach by several orders of magnitude.

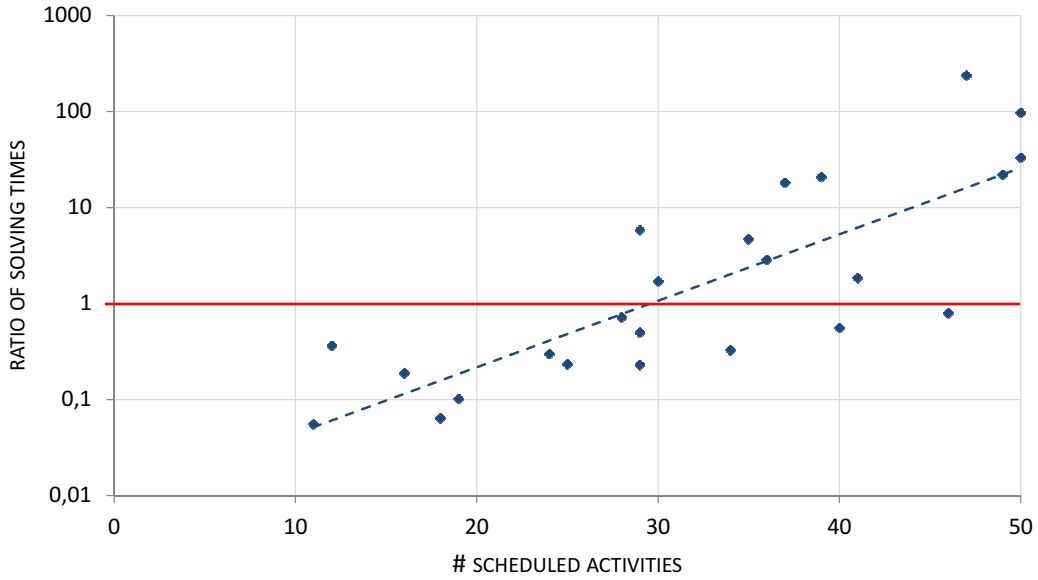


Figure 8.4: Fair comparison of ALNS and exact approach

In Fig. 8.5, we illustrate the typical search progress of the ALNS algorithm and the exact approach. The chart shows the evolution of the best found objective value for the largest of the 24 instances having exactly 50 scheduled activities. For this instance, the final solution of ALNS and the solution of the exact approach after the two-hour time limit are identical. The chart confirms the assumption that ALNS is a reasonable strategy for searching good tail assignment solutions in limited time.

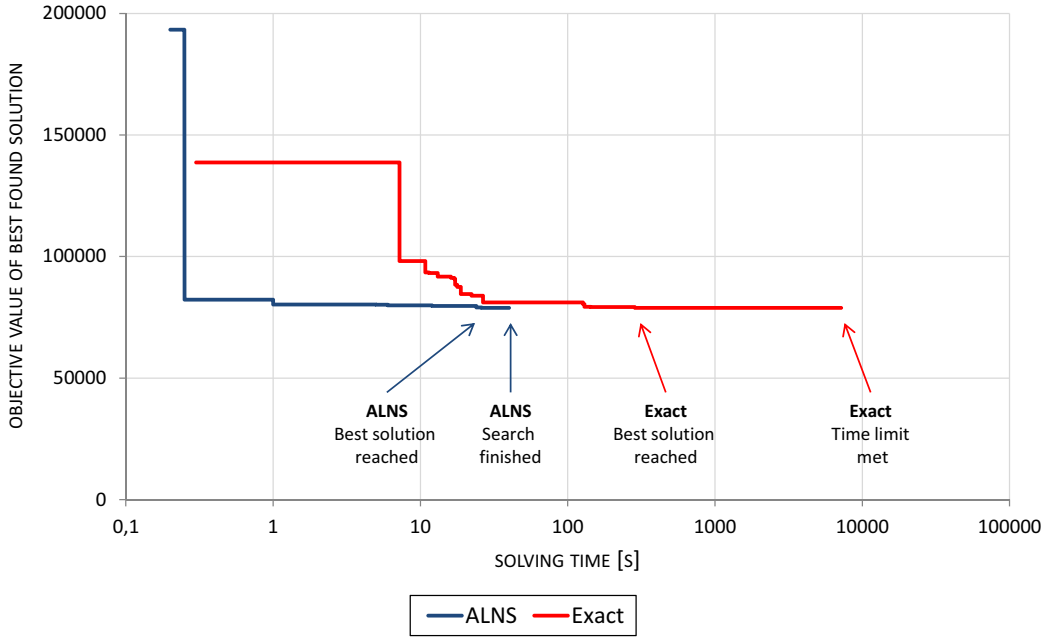


Figure 8.5: Progress of ALNS and exact search on instance with 50 activities

9 Conclusion

In the last chapter of this thesis, we summarize our work (Sec. 9.1) and provide an outlook on future research possibilities (Sec. 9.2).

9.1 Summary

In this research, we have looked in detail into the tail assignment problem of airlines. The problem consists of creating operational feasible routes for individual aircraft to cover all flights in a schedule. We have described the real-world problem as well as its context in the airline planning process and summarized the relevant literature.

The tail assignment process of airlines is not optimized satisfactorily yet. Most related studies focus on the tactical aircraft maintenance routing problem that aims to create generic routes for anonymous aircraft providing a maximum number of maintenance opportunities. A maintenance opportunity does not indicate whether maintenance tasks are actually performed or not. Therefore, these studies are unsuitable for operational maintenance planning and do not contribute to reduce the maintenance effort of airlines. Tail-dependent restrictions, like preassigned activities and curfews, are neglected. Also, only the most frequent maintenance requirements are considered, since the scheduling of less frequent checks depends on the maintenance and flying history of the individual tails.

In contrast, the operational tail assignment problem aims to provide results for real-world implementation. Nevertheless, the majority of existing studies is incapable of capturing in detail the extensive maintenance requirements and their complex interdependencies. Most researchers employ aggregated, simplified maintenance rules that do not reflect sufficiently the real-world situation.

We have developed a comprehensive mathematical model that is able to capture in detail all relevant objectives and constraints of the real-world tail assignment problem. The model is based on a connection network. The objective is to optimize maintenance planning by minimizing unnecessary and untimely maintenance activities. Additionally, the user can set individual cost penalties on connecting any two activities by a particular tail. Our tail assignment model can be solved for several fleets simultaneously and thus enables to consider the limited availability of common resources, e.g., hangar space. By integrating key fleeting decisions, we allow airlines to optimize fleeting and routing decisions simultaneously close to the day of operation.

As solution methodology, we have introduced an adaptive large neighborhood search algorithm. We have shown how our math-heuristic framework can be designed to quickly generate good problem solutions. Our algorithm consists of two stages. First, we create a feasible initial solution for our comprehensive model. Second, we iteratively improve the given solution by alternately releasing and reoptimizing different parts of the assignments using a MIP solver.

We have designed and evaluated different methods to create a feasible initial solution for the tail assignment problem. Besides the obvious approach to solve the comprehensive MIP as pure feasibility problem using a MIP solver, we have investigated heuristic approaches. Furthermore, we have developed a customized row generation framework in conjunction with a specialized heuristic. In this approach, we employ a restricted problem formulation based on a time-space network. The restricted problem is very compact and can be solved quickly using a MIP solver. The provided partial solutions are heuristically completed. We create Benders cuts to rule out partial solutions that cannot be completed to feasible solutions. The cuts are added to the restricted problem before it is resolved. Finally, we obtain a feasible initial solution for the tail assignment problem after running through several iterations.

Extensive benchmarking tests have shown that the last approach is superior to the other ideas of how to generate a feasible initial solution for the tail assignment problem. We have developed several preprocessing and model enhancement methods to further reduce problem sizes and solving times. For 208 out of 220 test instances with up to 180 aircraft and 2,500 flights, we are able to generate start solutions in less than 30 minutes. For 189 of the test instances it took less than ten seconds.

In the improvement stage, we employ several competing destructor methods in an adaptive framework. The probability that a particular method is selected is based on its performance in the past iterations. In doing so, we achieve that promising methods are used more often than unsuccessful methods. The search for improving neighbors is stopped when an acceptable solution quality is achieved, an upper time/ iteration limit met, or when only marginal improvements have been realized in the past iterations. We have shown how the adaptive framework can be tuned according to the preferences of the user.

In our computational study, we have compared the performance of our algorithm to an exact approach on a couple of small-sized instances. The results confirm that our algorithm outperforms the exact approach on instances of practically relevant size by several orders of magnitude. Our algorithm is able to find very good solutions for the large majority of the instances in less than 15 minutes. The timeliness of the scheduled maintenance activities is satisfying. For more than 60 percent of the instances, we plan maintenance so that less than ten percent of the allowed intervals between two consecutive checks are wasted. Even in the optimal solution, a perfect utilization of the allowed intervals is seldom achievable. The good performance proves that our adaptive large neighborhood search algorithm is an appropriate methodology to derive high quality tail assignment solutions in acceptable time frames.

9.2 Future work

The development of our tail assignment optimizer is not finalized yet. There are still many possibilities to improve performance in both solution time and quality.

Initial solution stage

A great potential to speed up solving times in the initial solution stage is the design of the heuristic to derive the routings in multi aircraft groups. When using a more intelligent heuristic, the obtained routings are less likely to be maintenance infeasible. In consequence, the required number of iterations is reduced and the disaggregation of commodities avoided. Our priority heuristic is far more intelligent than the naïve FIFO rule. However, the implemented heuristic could be further enhanced. An improved heuristic could be able to look several activities into the future routing sequence to evaluate the impact of local decisions. Our priority heuristic only tries to take locally reasonable decisions by looking one routing step ahead. It is also imaginable to employ several heuristics, and use information about conflicts as knowledge in subsequent algorithms. As the time requirement of heuristics is generally extremely low, the total iteration time would only marginally increase, even when running multiple heuristics consecutively.

In addition, the communication with the MIP solver during the row generation iterations offers great room for improvements. Currently, we employ a non-incremental communication via LP files. An incremental communication should reduce solving times considerably, as the solver would not need to reconstruct the problem in every iteration, and also it could use the previous solution or LP basis as warm start.

It would also be interesting to develop and evaluate other solution methodologies for the initial solution stage. In particular, this could be a constraint programming approach. Grönkvist reports good results in his studies on using constraint programming to generate a feasible initial solution [29, 30]. A comparison with our row generation framework on an equal set of instances would be a useful analysis.

Improvement stage

Although we have spent the largest proportion of our research effort on optimizing the initial solution stage, the improvement stage features interesting research possibilities as well. Of course, many other, more intelligent destructor methods could be designed. In addition, alternative repair methods could be implemented. A promising and simple idea would be to work with multiple objective functions. There are some intermediate objectives that might support detecting promising areas of the solution space and achieving better results in the long run. For example, we could implement an utilization balancing objective, which is randomly used in some iterations instead of the real objective. Balancing utilization distributes workload evenly on all commodities and in turn might result in better routing and maintenance plans.

The adaptive framework could be reoptimized as well. In our parameter tuning, we only defined a few candidate values per parameter and only two user profiles. It would be interesting to extend the evaluation to larger sets of candidate values given a larger pool of user profiles.

Parallelization of subproblems

Another great potential to increase the performance of our algorithm would be to run several subproblems in parallel on separate processors. A prerequisite for parallelization is that the subproblems are not overlapping. Parallelization only works when all processors work on completely independent subproblems. A promising and obvious possibility to use parallelization is the maintenance reduction stage, in which we call the single route destructor one after another for every tail to remove superfluous checks from the initial routes of the aircraft. As this destructor does not allow the flight sequences to be changed, the subproblems fulfil the requirement to be completely independent. It would be possible to transform the sequential procedure on one processor to a parallelized procedure on multiple processors, each solving the subproblem of an individual tail.

Schedule decomposition

We also discovered that the schedules of some fleet consist of separate subschedules. For example, the AT7 fleet of Carrier One is used to perform short feeder flights to and from the hubs. All aircraft of the AT7 fleet are explicitly assigned to a single hub. The fleet does not perform any connecting flights between the hubs and the served destinations are disjoint sets for every subschedule. Given this structure, we can decompose the flight schedule of a fleet in subschedules for smaller subfleets. By successively solving the decomposed problems for every subfleet, we can achieve better results in smaller time frames without losing any valid solution.

Disruption recovery

A further field of research would be to extend the applicability of our optimizer on the disruption recovery scenario. In this scenario, it is likely that the initial positions and conditions of the aircraft do not allow all activities to be performed. But also in the standard tail assignment scenario, it could be beneficial to allow unassigned activities. For example, when the rules specified by the user are too strict or in case an airplane becomes unavailable for a long time period [28]. To avoid that the entire problem becomes infeasible, we could allow activities to be left unassigned and instead incorporate a high penalty cost for any uncovered activity.

During our research, we already started to study this setting. To allow activities to be left unassigned, our main model requires slight modifications. The arc set O is

enlarged by special arcs in the form of $\langle i, i, r, g \rangle$ allowing an activity to be connected to itself. These arcs guarantee flow balance, even when activities have been left unassigned. But, as we want to minimize the occurrence of unassigned activities, we consider a very high penalty cost C'_{iirg} in our objective function.

Allowing activities to be left unassigned comes along with advantages and disadvantages. A major advantage is that there is now an obvious feasible start solution, namely the situation in which no activity is performed. We therefore could eliminate the entire initial solution stage and directly start the improvement iterations from this solution. But, as not performing any activity is the worst possible solution for an airline, we would have to implement destructor and repair methods that are able to quickly assign as many activities as possible to the routes of the aircraft. A more reasonable approach is probably to keep an initial solution stage, in which a better start solution is generated. The possibility to leave activities unassigned is still an advantage, as it enables us to create feasible solutions from any partial solution.

A major drawback of allowing activities to be left unassigned is that most of our preprocessing methods are not applicable anymore. Since we do not know in advance which activities are performed and which are left unassigned, we cannot compute the aircraft flow a priori. We thus are not able to eliminate connection arcs that violate the aircraft flow balance. Aggregating activities is only possible if we assume that either both activities are performed or left unassigned. In general, problem sizes will be larger than when all activities have to be performed.

Bibliography

[1] Yossiri Adulyasak, Jean-François Cordeau, and Raf Jans. Optimization-based adaptive large neighborhood search for the production routing problem. *Transportation Science*, 48(1):20–45, 2014.

[2] H. Murat Afsar, Marie-Laure Espinouse, and Bernard Penz. A two-step heuristic to build flight and maintenance planning in a rolling-horizon. In *International Conference on Service Systems and Service Management*, pages 1251 – 1256. IEEE, 2006.

[3] H. Murat Afsar, Marie-Laure Espinouse, and Bernard Penz. Building flight planning for an airline company under maintenance constraints. *Journal of Quality in Maintenance Engineering*, 15(4):430–443, 2009.

[4] Ravindra K. Ahuja, Özlem Ergun, James B. Orlin, and Abraham P. Punnen. A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, 123(1-3):75–102, 2002.

[5] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network flows - theory, algorithms and applications*. Prentice Hall, Upper Saddle River, New Jersey, 1993.

[6] Jens Appel. Trends und Perspektiven im Luftverkehr. Presentation at KIT Karlsruhe Institute of Technology, February 2005. `https://www.ise.kit.edu/rd_download/SBT/Kolloquium_SBT_05-02_J._Appel.pdf` [28.05.2015].

[7] Leif H. Appelgren. Integer programming methods for a vessel scheduling problem. *Transportation Science*, 5(1):64–78, 1971.

[8] Michael F. Argüello, Jonathan F. Bard, and Gang Yu. A grasp for aircraft routing in response to groundings and delays. *J. Comb. Optim.*, 1(3):211–228, 1997.

[9] Cynthia Barnhart, Natashia L. Boland, Lloyd W. Clarke, Ellis L. Johnson, George L. Nemhauser, and Rajesh G. Shenoi. Flight string models for aircraft fleeting and routing. *Transportation Science*, 32(3):208–220, 1998.

[10] Cynthia Barnhart, Ellis L. Johnson, George L. Nemhauser, Martin W. P. Savelsbergh, and Pamela H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46(3):316–329, 1998.

[11] Cynthia Barnhart and Kalyan T. Talluri. Airline operations research. In Charles ReVelle and Arthur E. McGarity, editors, *Design and operation of civil and*

environmental engineering systems, pages 435–470. John Wiley & Sons, New York, 1 edition, 1997.

[12] Mehmet Basdere and Ümit Bilge. Operational aircraft maintenance routing problem with remaining time consideration. *European Journal of Operational Research*, 235(1):315–328, 2014.

[13] Massoud Bazargan. *Airline operations and scheduling*. Ashgate, Farnham, Surrey, 2 edition, 2010.

[14] Serge Bisaillon, Jean-François Cordeau, Gilbert Laporte, and Federico Pasin. A large neighbourhood search heuristic for the aircraft and passenger recovery problem. *4OR-Q J Oper Res*, 9(2):139–157, 2010.

[15] Boeing. 737-600/700/800/900 Maintenance Review Board Report, 2007.

[16] Boeing. 757 Maintenance Review Board Report, 2007.

[17] Boeing. 767 Maintenance Review Board Report, 2008.

[18] Boeing. Current market outlook 2014-2033, 2014. `http://www.boeing.com/resources/boeingdotcom/commercial/about-our-market/assets/downloads/Boeing_Current_Market_Outlook_2014.pdf` [28.05.2015].

[19] X. Cai and C.J. Goh. A fast heuristic for the train scheduling problem. *Computers & Operations Research*, 21(5):499–510, 1994.

[20] Lloyd Clarke, Ellis Johnson, George Nemhauser, and Zhongxi Zhu. The aircraft rotation problem. *Annals of Operations Research*, 69(0):33–46, 1997.

[21] Amy Mainville Cohn and Cynthia Barnhart. Improving crew scheduling by incorporating key maintenance routing decisions. *Operations Research*, 51(3):387–396, 2003.

[22] Jean-François Cordeau, Goran Stojkovic, François Soumis, and Jacques Desrosiers. Benders decomposition for simultaneous aircraft routing and crew scheduling. *Transportation Science*, 35(4):375–388, 2001.

[23] Gunter Dueck. New optimization heuristics. *Journal of Computational Physics*, 104(1):86–92, 1993.

[24] Federal Aviation Administration. Advisory Circular 121-22C: Maintenance Review Boards, Maintenance Type Boards, and OEM/TCH Recommended Maintenance Procedures, 2012. `http://www.faa.gov/documentLibrary/media/Advisory_Circular/AC%20121-22C.pdf` [28.05.2015].

[25] M.A. Forbes, J.N. Holt, and A.M. Watts. An exact algorithm for multiple depot bus scheduling. *European Journal of Operational Research*, 72(1):115–124, 1994.

[26] Keivan Ghoseiri, Ferenc Szidarovszky, and Mohammad Jawad Asgharpour. A multi-objective train scheduling model and solution. *Transportation Research Part B: Methodological*, 38(10):927–952, 2004.

[27] Ram Gopalan and Kalyan T. Talluri. The aircraft maintenance routing problem. *Operations Research*, 46(2):260–271, 1998.

[28] Mattias Grönkvist. Tail assignment - a combined column generation and constraint programming approach. Thesis for the degree of licentiate of engineering, Chalmers University of Technology and Göteborg University, 2003.

[29] Mattias Grönkvist. The tail assignment problem. Thesis for the degree of doctor of philosophy, Chalmers University of Technology and Göteborg University, 2005.

[30] Mattias Grönkvist. A constraint programming model for tail assignment. In Jean-Charles Régin and Michel Rueher, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, First International Conference, CPAIOR 2004, Nice, France, April 20-22, 2004, Proceedings*, volume 3011 of *Lecture Notes in Computer Science*, pages 142–156. Springer, 2004.

[31] Christopher A. Hane, Cynthia Barnhart, Ellis L. Johnson, Roy E. Marsten, George L. Nemhauser, and Gabriele Sigismondi. The fleet assignment problem: Solving a large-scale integer program. *Math. Program.*, 70:211–232, 1995.

[32] Mohamed Haouari, Najla Aissaoui, and Farah Zeghal Mansour. Network flow-based approaches for integrated aircraft fleeting and routing. *European Journal of Operational Research*, 193(2):591–599, 2009.

[33] Mohamed Haouari, Shengzhi Shao, and Hanif D. Sherali. A lifted compact formulation for the daily aircraft maintenance routing problem. *Transportation Science*, 47(4):508–525, 2013.

[34] Mohamed Haouari, Hanif D. Sherali, Farah Zeghal Mansour, and Najla Aissaoui. Exact approaches for integrated aircraft fleeting and routing at tunisair. *Comp. Opt. and Appl.*, 49(2):213–239, 2011.

[35] Vera C. Hemmelmayr, Jean-François Cordeau, and Teodor Gabriel Crainic. An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. *Computers & OR*, 39(12):3215–3228, 2012.

[36] Andrew J. Higgins, Erhan Kozan, and Luis Ferreira. Heuristic techniques for single line train scheduling. *J. Heuristics*, 3(1):43–62, 1997.

[37] Frederick S Hillier and Gerald J Lieberman. *Introduction to operations research*. McGraw-Hill Higher Education, New York, 2010.

[38] Martin Hinsch. MSG-3 Eine Einführung in die Bestimmung grundlegender Instandhaltungsmaßnahmen bei Verkehrsflugzeugen. Hamburg, 2011. `http://www.aeroimpulse.de/MSG-3-Einfuehrung.pdf` [28.05.2015].

[39] John N. Hooker and Greger Ottosson. Logic-based benders decomposition. *Mathematical Programming*, 96(1):33–60, 2003.

[40] International Air Transport Association. Annual review 2013, 2013. http://www.iata.org/about/documents/iata-annual-review-2013-en.pdf [28.05.2015].

[41] International Air Transport Association. Passenger demand maintains historic growth rates in 2013. online press release, February 2014. http://www.iata.org/pressroom/pr/pages/2014-02-06-01.aspx [28.05.2015].

[42] Timothy L. Jacobs, Laurie A. Garrow, Manoj Lohatepanont, Frank S. Koppelman, Gregory M. Coldren, and Hadi Purnomo. Airline planning and schedule development. In Cynthia Barnhart and Barry Smith, editors, *Quantitative problem solving methods in the airline industry*. Springer, New York, 1 edition, 2012.

[43] Hai Jiang and Cynthia Barnhart. Dynamic airline scheduling. *Transportation Science*, 43(3):336–354, 2009.

[44] Nader M. Kabbani and Bruce W. Patty. Aircraft routing at american airlines. In *Proceedings of the 32nd Annual Symposium of AGIFORS*. Agifors, 1992.

[45] Diego Klabjan. Large-scale models in the airline industry. In Guy Desaulniers, Jacques Desrosiers, and Marius M. Solomon, editors, *Column generation*, pages 163–195. Springer, New York, 1 edition, 2005.

[46] Natalia Kliewer, Taïeb Mellouli, and Leena Suhl. A time-space network based exact optimization model for multi-depot bus scheduling. *European Journal of Operational Research*, 175(3):1616–1627, 2006.

[47] Jens Koenen. Der Kranich ist noch krank. *Handelsblatt*, 2015. http://www.handelsblatt.com/unternehmen/handel-konsumgueter/lufthansa-analyse-der-kranich-ist-noch-krank/11729532.html [28.05.2015].

[48] Gilbert Laporte, Roberto Musmanno, and Francesca Vocaturo. An adaptive large neighbourhood search heuristic for the capacitated arc-routing problem with stochastic demands. *Transportation Science*, 44(1):125–135, 2010.

[49] Marcial Lapp and Florian Wikenhauser. Incorporating aircraft efficiency measures into the tail assignment problem. *Journal of Air Transport Management*, 19:25–30, 2012.

[50] Zhe Liang and Wanpracha Art Chaovalitwongse. A network-based model for the integrated weekly aircraft maintenance routing and fleet assignment problem. *Transportation Science*, 47(4):493–507, 2013.

[51] Zhe Liang, Wanpracha Art Chaovalitwongse, Huei Chuen Huang, and Ellis L. Johnson. On a new rotation tour network model for aircraft maintenance routing problem. *Transportation Science*, 45(1):109–120, 2011.

[52] Lufthansa Technik. Price List Line Maintenance Services (German Stations only), 2015. http://www.lufthansa-technik.com/documents/100446/101431/Pricelist+German+Stations+2014.pdf [28.05.2015].

[53] Chris Melzer. Vor 100 Jahren startete die kommerzielle Luftfahrt. *Die Welt*, 2014. http://www.welt.de/geschichte/article123412576/Vor-100-Jahren-startete-die-kommerzielle-Luftfahrt.html [28.05.2015].

[54] Laurent Flindt Muller, Simon Spoorendonk, and David Pisinger. A hybrid adaptive large neighborhood search heuristic for lot-sizing with setup times. *European Journal of Operational Research*, 218(3):614–623, 2012.

[55] Aarti Nagraj. Emirates aims to fly 70 million passengers in 2020. *Gulf Business*, 2013. http://gulfbusiness.com/2013/10/emirates-aims-to-fly-70-million-passengers-in-2020/ [28.05.2015].

[56] Nikolaos Papadakos. Integrated airline scheduling. *Computers & OR*, 36(1):176–195, 2009.

[57] Junhyuk Park and Byung-In Kim. The school bus routing problem: A review. *European Journal of Operational Research*, 202(2):311–319, 2010.

[58] David Pisinger and Stefan Ropke. Large neighborhood search. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of Metaheuristics*, pages 399–419. Springer, New York, 2 edition, 2010.

[59] Ted Reed. Airlines, not yet where they want to be, make 21 cents per passenger. *Forbes*, 2013. http://www.forbes.com/sites/tedreed/2013/02/25/airlines-not-yet-where-they-want-to-be-make-21-cents-per-passenger/ [28.05.2015].

[60] David Ronen. Cargo ships routing and scheduling: Survey of models and problems. *European Journal of Operational Research*, 12(2):119–126, 1983.

[61] Stefan Ropke and David Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472, 2006.

[62] Sebastian Ruther, Natashia Boland, Faramroze Engineer, and Ian Evans. Integrated aircraft routing, crew pairing, and tail assignment: branch-and-price with many pricing problems. *Transportation Science*, 2013 (unpublished manuscript).

[63] Nadia Saleem and Mark Potter. Emirates targets at least 8-10 pct profit growth on new aircraft, routes. *Reuters*, 2014. http://reut.rs/1F09gWw [28.05.2015].

[64] Rivi Sandhu and Diego Klabjan. Integrated airline fleeting and crew-pairing decisions. *Operations Research*, 55(3):439–456, 2007.

[65] Abdulkadir Sarac, Rajan Batta, and Christopher M. Rump. A branch-and-price approach for operational aircraft maintenance routing. *European Journal of Operational Research*, 175(3):1850–1869, 2006.

[66] Gerhard Schrimpf, Johannes Schneider, Hermann Stamm-Wilbrandt, and Gunter Dueck. Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, 159(2):139–171, 2000.

[67] Paul Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In Michael J. Maher and Jean-Francois Puget, editors, *Principles and Practice of Constraint Programming - CP98, 4th International Conference, Pisa, Italy, October 26-30, 1998, Proceedings*, volume 1520 of *Lecture Notes in Computer Science*, pages 417–431. Springer, 1998.

[68] Frank Sieren. Chinese aviation sector readies for takeoff. *Deutsche Welle*, 2014. `http://dw.de/p/1DuDP` [28.05.2015].

[69] Eike Stumpf. Systeme der Luft- & Raumfahrt - Unfälle & Zuverlässigkeit. Lecture at RWTH Aachen University, 2012.

[70] Kalyan T. Talluri. The four-day aircraft maintenance routing problem. *Transportation Science*, 32(1):43–53, 1998.

[71] Trefis Team. Airline industry will have to maintain capacity discipline to remain profitable. *Forbes*, 2014. `http://www.forbes.com/sites/greatspeculations/2014/06/20/airline-industry-will-have-to-maintain-capacity-discipline-to-remain-profitable/` [28.05.2015].

[72] Stefan Voß, Ibrahim H. Osman, and Catherine Roucairol. *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*. Kluwer Academic Publishers, Boston, 1 edition, 1999.

[73] Stefan Voß. Meta-heuristics: The state of the art. In Alexander Nareyek, editor, *Local Search for Planning and Scheduling, ECAI 2000 Workshop, Berlin, Germany, August 21, 2000, Revised Papers*, volume 2148 of *Lecture Notes in Computer Science*, pages 1–23. Springer, 2000.

[74] Valdemar Warburg, Troels Gotsæd Hansen, Allan Larsen, Hans Norman, and Erik Andersson. Dynamic airline scheduling: An analysis of the potentials of refleeting and retiming. *Journal of Air Transport Management*, 14(4):163–167, 2008.

[75] Simon Wright. Why airlines make such meagre profits. *The Economist*, 2014. `http://www.economist.com/blogs/economist-explains/2014/02/economist-explains-5` [28.05.2015].

[76] Cheng-Lung Wu. *Airline operations and delay management*. Ashgate, Farnham, Surrey, 1 edition, 2010.

[77] Hans-Jürgen Zimmermann. *Operations research*. Vieweg, Wiesbaden, 1 edition, 2005.