View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by University of Waterloo's Institutional Repository

Pairing Generation for Airline Crew Scheduling

by

Daniel Andreas Bayer

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Management Sciences

Waterloo, Ontario, Canada, 2012

© Daniel Andreas Bayer 2012

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Airline planning is a complex and difficult process. The biggest airlines in the world plan for and operate fleets of over 700 aircraft using tens of thousands of crew members. As such, small percentages in savings translate to millions of dollars.

In this thesis, we study the pairing and duty generation problem in the context of airline crew scheduling, and propose approaches to improve the computational speed and the solution quality. We propose several enumeration algorithms to generate all possible duty periods of a given schedule to improve on the time required to generate duty periods; and present a set of column generation models to improve on the solution quality. When tested on a real test case study, the proposed approaches are found to improve the computational times from 142 seconds down to less than one second, and the cost savings of 13.7%.

Acknowledgements

I would like to thank my supervisor, Professor Samir Elhedhli, for his guidance and help in this research project.

I would like to thank Navtech and Robert Mora from Navtech for the role they played in making this possible. I would also like to thank Professor Fatma Gzara, Da Lu and Ahmed Saif from the University of Waterloo for their participation in our Navtech-Mitacs research project.

I would like to thank the readers of this thesis, Professor Benny Mantin and Professor Fatma Gzara, for their valuable comments.

I would like to thank my parents for giving me the opportunity to pursue my education. And I would like to thank Melissa for her support and patience during the writing of this thesis.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

The airline crew scheduling problem is defined by the minimization of the cost associated with flight crews through the development of a schedule which covers all flight legs over a defined period of time. This problem is a monthly planning process at most airlines, involving the creation of a sequence of flight assignments, starting and ending at a crew-base, subject to constraints imposed by aviation regulations, collective bargaining agreements (CBA) and airline policies. The importance of this problem arises from the expense crews represent in an airline's budget. Next to fuel costs, crews are the largest direct operating cost faced by an airline [2], [3], [4], [5]. This means that the research efforts aimed at reducing this cost, even by small percentages, are easily justifiable.

An airline the size of American Airlines had crew costs in excess of \$1.3 billion per year around 1990 [3], [4], meaning a one percent savings equates to \$13 million per year. United Airlines' annual savings due to crew scheduling optimization is \$16-\$18 million on a total crew payroll of \$600 million. These savings are achieved in part by using integer linear programming over enumeration methods [23], [25]. The magnitude of these figures are not uncommon. Northwest Airlines had a crew payroll of just over \$1 billion [5]. Andersson et al. [5] report that these figures are representative of European airlines as well.

As the size of the economic impact implies, the size of the scheduling problem is large and its solution is difficult. The magnitude of the above figures are a result of the size of operations. Major US airlines use nearly 500 jets (Delta Airlines has a fleet of over 700 aircraft) to serve more than 150 cities with 2,500 daily flights. This requires some 5,000 cockpit crews and 10,000 flight attendants [24].

There are five characteristics defining the airline crew scheduling problem: distinct crew categories, the fleet types which the airline operates, the network structure which the airline operates, all applicable rules and regulations pertaining to crewed operations, structure of the timetable of flight legs and the crew cost structures.

An airline deals with several distinct crew categories to be able to operate their routes. Among the flight crew there are captains and first officers and in some cases flight engineers. Also on board of an aircraft there is a purser, flight attendants and hostesses. All of these crews require separate scheduling, are not interchangeable and are subject to varying rules and regulations, which allows the problem to be broken down by crew type. Captains and first officers are scheduled together but the pilot problem can be broken down by fleet type. Pilots are trained to operate only one type of aircraft and are not interchangeable. The cabin crew problem is subject to different rules and can be subject to their own CBAs. The cabin crew problem at United Airlines, for example, is decomposed by wide-body fleet, narrow-body fleet requiring more than three flight attendants, and narrow-body fleet requiring three or less flight attendants [5].

Airlines operate networks which can have various and specific characteristics. The three main types of networks in use are hub-and-spoke networks, point-to-point networks, and international long-haul networks. Hub-and-spoke networks allow airlines to connect a large amount of cities (nodes) with relatively few flying routes (arcs), see figure 1.1. This means that there must be many connections available at central hubs. As such these networks are designed with short time periods, in which many flights arrive into one hub and then subsequently have a period of many flights departing. This achieves the goal of maximizing the trip origin and destination combinations available, given a limited set of flying resources. One of the drawbacks is that crews then also have hundreds of connection possibilities at hubs, increasing the size of the problem. In addition, many hubs are crew

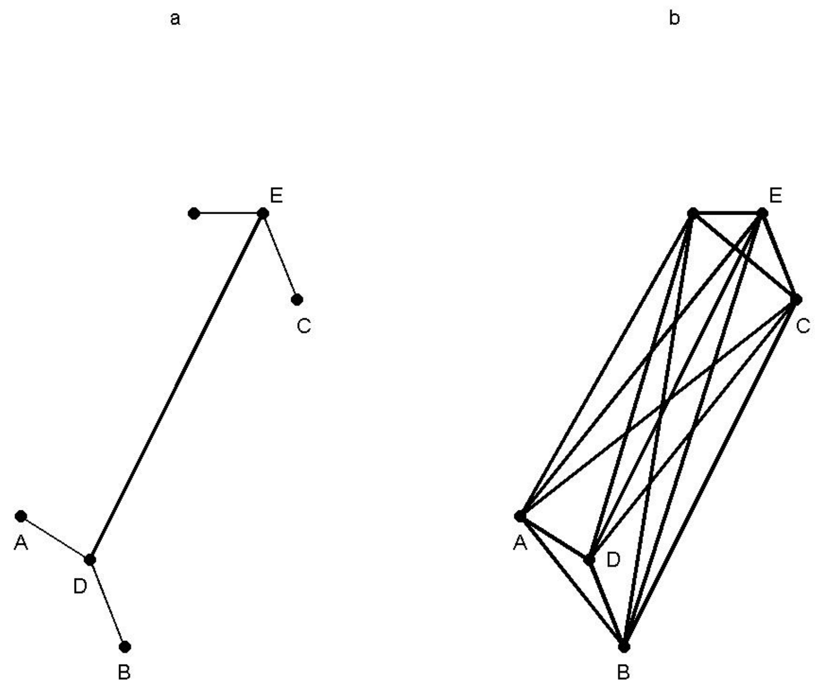

Figure 1.1: Sample networks of a)hub and spoke network and b)point to point network.

bases, meaning that not only are there many possible connections, it is also possible to end a crew's pairing and there are then also many possibilities to start a new crew's pairing. Most major airlines operate multiple hubs, making the problem more challenging. Point-to-point networks tend to be more desirable for the passengers. Airlines attempt to offer the same travel options but with fewer connections. For example, on a hub-and-spoke network, panel (a) in figure 1.1, three connected flights may be required to travel from city A to city C and two flights are required to get from A to B. A point-to-point network tries to offer direct flights. In panel (b) of figure 1.1, a passenger can travel from A to either B or C with one flight. In order to offer this benefit, such a network would need to be fully connected and would require many more flying resources.

International long haul routes often have an aircraft flying to a destination far away and returning to the same origin. In some cases the aircraft may fly back to a different airline hub or base. These types of networks tend to have limited possibilities in terms of feasible flight sequences for air crews. Therefore, long haul crew planning problems have unique solution characteristics. For example, in figure 1.1, these pilots would be flying between D and E and connecting to other hubs in the network.

The crew scheduling problem is generally treated as two separate problems. First anonymous schedules are created and selected to cover all flights, called the crew pairing problem. Second, these schedules are assigned to specific crew members in the crew rostering problem [5].

Duties and pairings are subject to a host of complex rules and constraints imposed by aviation regulations, CBAs, and airline policies. Usually regulations set forth by regulatory bodies such as the Federal Aviation Administration (FAA) are in place for the safety of pilots, passengers and anybody who may be impacted by general aviation, including the public. Collective bargaining agreements are agreements between the airline and the crews which are intended to protect the crews and ensure reasonable working conditions. The airline itself will also set scheduling policies to ensure the operability of the created schedule as well as robustness of the schedule.

Legs in duties and pairings must be sequential in space and time. There are constraints on maximum and minimum permissible time between legs. There are constraints on the length of 'on-duty' time, and on the amount of actual flying time is allowed to be flown in a duty period. Pairings must start and end at a crew-base. Pairings generally are generated to adhere to a prescribed length criteria. This length is typically 1 to 5 days in short to medium haul domestic operations in North America. American Airlines for example limits pairings to three days [4]. International long haul problems generally allow longer pairings [5]. The maximum length of pairings is usually set by airline regulations. For example, some rules could be that each duty period is at most 12 hours long, a layover is at least 9 hours long or 1.5 times the length of the flying time in the preceding duty period, whichever is longer. The maximum layover length is 32 hours long [5].

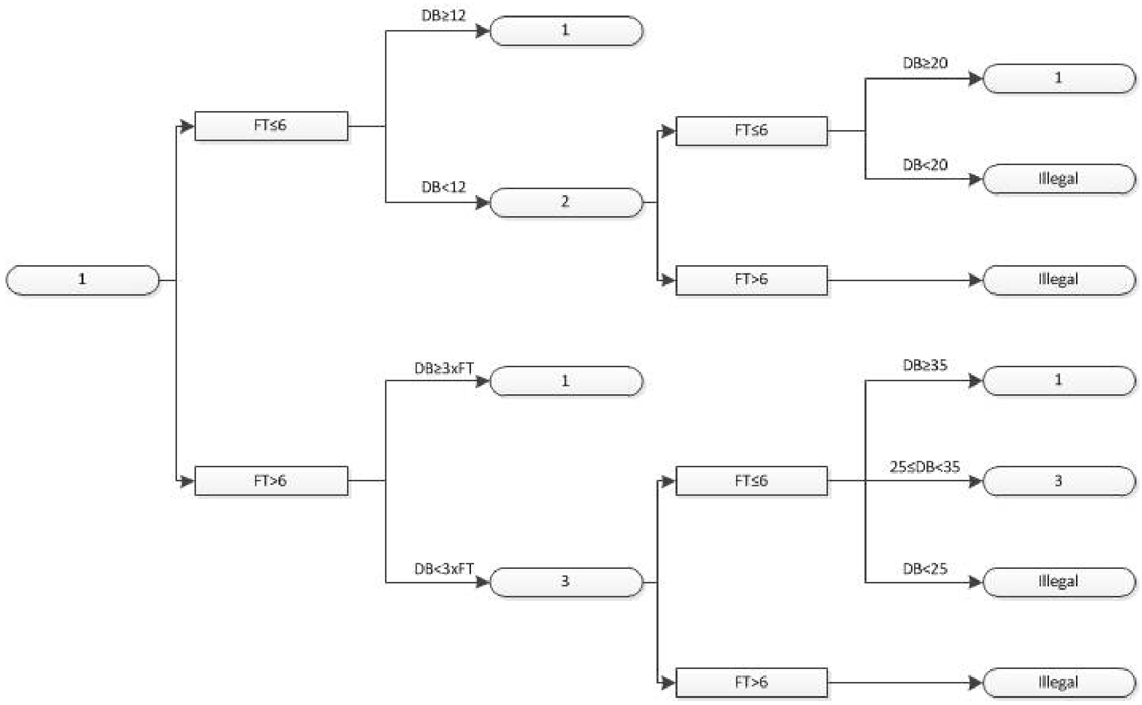

Figure 1.2: Decision tree illustration of the complexity of crew rest rules as demonstrated by Lavoie et al. [28].

Within pairings there are complex rest requirements, beyond strict maximums and minimums, which are a function of the flight time and duty lengths within the pairing. The 8-in-24 rule is one example of these complex rules. The 8-in-24 rule is a regulation imposed by the FAA, requiring that if during any given 24 hour period, the flying time of any crew exceeds 8 hours, that crew must be compensated with extra rest. This 24 hour period is evaluated as a sliding time interval. The violation of the 8 hour limit then requires a crew to receive an intervening rest period to break up the flying times, which is a function of the length of flying, as well as a 14 hour rest opportunity immediately after the flying which violated the rule [24]. Lavoie et al. [28] provide another example of a complex rule and an excellent decision tree which demonstrates the complexity of rules associated with rest time requirements. This case is replicated in figure 1.2.

In figure 1.2 rounded boxes indicate states of the pairings, rectangles indicate duty periods and arcs with captions indicate duty breaks and their corresponding lengths. There are three states which a pairing can find itself in, state 1 is a normal state where there are no rest deficiencies. State 2 is a rest deficiency which occurs when a duty period has more than six hours of flight time and is followed by less than twelve hours of rest time. This deficiency state can only be followed by a duty of less than six hours of flight time and then a compensatory rest period of 20 hours or more. State 3 is a state in which a duty period of more than six hours is followed by a short rest, less than three time the flying time. This deficiency state must be followed by less than six hours of flying time and at least 25 hours of rest time. If the rest time exceeds 35 hours, the deficiency is resolved, but if the rest time falls between 25 and 35 hours the deficiency is propagated. Given these various states and conditions for compensating for deficient rest times, it becomes easy to see the difficulty in mathematically programming such a system.

Costing approaches can also vary from airline to airline. In the case of airlines which pay crews on monthly salaries, duties can be assigned a cost equal to the number of crews required to operate that pairing. If the pairing is repeated day to day, the cost would be equal to the length of the pairing. The solution in this case would be a minimum amount of crews required to operate all pairings. This should yield a set of pairings with high utilization, in other words, the pairings would have a maximized number of flight hours per pairing. This approach is relatively simple but unfortunately is not the predominant method. Airlines with salaried pilots tend to be European airlines. Crews are generally paid by flying time. This is the norm in North America. Crews can also be compensated for excessive ground times and time away from base (TAFB), and may be eligible for per diem when staying the night away from base. The cost function of a duty period can then be described as follows:

$$c_d = \max(f_d \times elapse, fly + dh, mg_d) \tag{1.1}$$

where f_d is a factor indicating the minimum amount of time a crew-member gets paid for each hour of duty time, *elapse* is the duty time, fly is the flight time in the duty period, dh is the amount of deadhead time for which a crew-member gets paid, and mg_d is the minimum guaranteed amount of time for which a pilot gets paid if they begin a duty [9]. As can be seen in the duty cost function, pilots receive pay for what the airline sees as unproductive time, that is, time where the pilot is not flying an aircraft and hence not contributing directly to revenue generation for the airline. This gives rise to the principle of *pay and credit* time. This is a quantity of time defined by the difference between the time for which a pilot gets paid and the amount of time of actual flying. In equation 1.1 the first term discourages duties which have long or frequent sit times in them and the third term discourages extremely short duty periods.

The cost function for a pairing can be expressed as follows:

$$c_p = \max(f_p \times TAFB, ndp \times mg_p, \sum_{d \in p} c_d) + \sum_{\hat{d}, \bar{d} \in p, \hat{d} \to \bar{d}} e(\hat{d}, \bar{d}) \tag{1.2}$$

where f_p is the amount of time a crew-member gets paid for every hour away from base, $TAFB$ is the time away from base in hours, ndp is the number of duty periods in the pairing, mg_p is the minimum guaranteed pay per duty period in the pairing and c_d is as defined in equation 1.1. And where $\hat{d}$ and $\bar{d}$ and duty periods in p, $\hat{d} \to \bar{d}$ indicating that $\bar{d}$ is the duty period which immediately follows duty $\hat{d}$, and $e(\hat{d}, \bar{d})$ is the cost of the duty break between duties $\hat{d}$ and $\bar{d}$. This cost can include many things such as hotel costs, per diem costs, meal costs or anything else associated with an overnight duty break.

As can be deduced from the above equations the sources of expensive pairings are long and frequent sit times, deadheads, and overnight rests. Frequently, pairings balance a tradeoff between deadheads and over-night stays.

In broad terms, airlines which are based in geographically similar locations use similar approaches to their operations. As such, traditionally the North American and European characteristics of the airline crew scheduling problem have been compared. A summary of the above outlined characteristics and a comparison of European and North American traits is provided in table 1.1.

	North America	Europe
Crew Category	Decomposes by crew category	
Fleet	The pilot problem decomposes by fleet type but cabin crew does not.	
Network Structure	Predominant hub-and-spoke structure. Designed to make many varying connections possible.	Less structure, more point-to-point flying.
Rules and Regulations	FAA regulations are most important and constraining.	Strong and complicated collective bargaining agreements.
Regularity of the Timetable	Repeating weekday services with reduced weekends.	Less regularity in day to day operations.
Cost Structure	Credit hour cost structure.	Crews receive fixed salaries.

Table 1.1: Comparison of the properties of the North American and European Crew Scheduling problems [5]

1.2 Terminology

As noted in section 1.1, one method of decomposing the problem is by aircraft type. A *Wide-body Aircraft* is one with two passenger aisles in the seating area. In contrast to this a *Narrow-body Aircraft* is one with only one aisle. The number of aisles is dependent on the fuselage diameter.

The basis for the scheduling of crews is the schedule of flights the airline has chosen to operate. These services are transportation to destinations given in the *Flight Schedule* or *Time Table*. This is usually set monthly and contains the particulars of the complete set of flights to be operated over the given time period. The flights themselves can be referred to as *Flight Legs*, *Flight Segments* or *Sectors*. A flight leg connects two airports referred to as *Stations*. A *Coterminal* is a location where multiple stations are in close proximity which and connectable by ground transport. This allows crews to fly into one station and then connect to another airport. This can happen in places like New York where John F. Kennedy International (JFK), Newark Liberty International (EWR), LaGuardia (LGA) and Long Island MacArthur (ISP) Airports are within relatively close proximity.

Additional examples are London, with London Heathrow (LHR), Gatwick (LGW), London Luton (LTN), London Stansted (STN), London City (LCY) airports and in Paris there are Paris-Charles de Gaulle (CDG) and Paris-Orly (ORY) airports. These airports could be connected efficiently by some other means of transportation.

From the flight schedule the *Aircraft Routing Plan* is created. This is the plan for the allocation of available aircraft to execute the flights outlined in the flight schedule.

In scheduling, a crew might need to be relocated in order to operate a specific flight. This is done by transporting them as passengers on another flight. This practice is referred to as *Deadheading*. This can be done on the airline's own flights, called *Online Deadheading*, or another carrier can be used, which is called *Offline Deadheading*. Offline deadheading can be through alliance partners or tickets may be bought on another airline through alternate arrangements. Crews can also be moved by other means if airports are sufficiently close together or otherwise easily connected. This type of crew positioning uses what is referred to as a *Ground Transfer*. The modes of transport used for ground transport include car, bus, water or train. In contrast to deadheading segments which are being operated by the crew are called *Live Segments*.

One of the building blocks of a solution the airline crew planning problem is a *Duty Period* or *Flight Service*. This is a legal work day composed of at least one flight leg or deadhead that can be executed without the requirement for a rest period. Duty periods are restricted by maximum limits on flight time, landings, time duration. There can be a large variety of duties that do not involve any flying as an operating crew member, such as training or other ground duties. Duties that do involve flying as part of an operational crew are sometimes called *Flight Duty Periods*. Duty periods and flight duty periods can be classified into a variety of categories. This may help with, or be important in things like costing or extra time requirements for things like clearing customs. These categories can include: domestic duty, international duty, night duty and redeye duty.

Within a duty, there can be many flight legs. Between those flight legs, aircraft need to be unloaded and reloaded, passengers need to deplane and board again and crew may need some time. The time between live flight legs is called the *Turnaround Time*, *Connection*

Time or *Sit Time*.

A series of one or more duty periods is called a *Pairing*, *Trip*, *Rotation*, *Sling* or a *Tour*. A pairing must start and end at an airport where crews are stationed, called a *Crew Base*. Within the pairing duty periods are separated by legal *Dutybreaks*, also called *Layovers*, *Rest Periods*, *Night Stops* or *Stop Time in Call*.

In scheduling pilots an important property of a flight leg is the *Block Time*. This is the scheduled duration of a flying segment from gate to gate. In general the *Segment Credit Time*, the amount of time awarded to a pilot for flying a given leg, is then the same as the block time however it may vary depending on the airlines and specific legs. The *Synthetic Time* or *Penalty Time* is the time for which a pilot gets paid but does not do any actual flying. Depending on contract rules this could include deadheading time, or just time flown below minimum guarantees. Where a *Minimum Guarantee* is a guaranteed minimum threshold amount of time paid whenever the pilot goes on duty. This may take the form of a hard minimum, imposed on each duty period, or a soft minimum, applied to the average duty period, depending on airline or contract rules. One way of stating synthetic time is

$$syn\_time = \max(0, guaranteed\_minimum - block\_time + deadhead\_time) \tag{1.3}$$

The total *Credit Time* is the sum of the segment credit times and the synthetic time, this is what dictates the pilot's pay and as seen in equation 1.3 cannot be less than the guaranteed minimum. Additional terms which can influence a pilot's pay included *Duty Rigs* and *Trip Rigs*. A duty rig is a factor which states that a pilot must receive a minimum amount of credit time as a function of the length of the duty period. In the statement 'For every x hours on duty, the crew-member is guaranteed one hour of flight pay', x is the parameter referred to as the duty rig. If the guaranteed amount is greater than the actual total segment credit time in the duty period the crew-member is credited with the difference. The trip rig is the same principle applied to an entire pairing.

1.3 Challenges

Aside from sheer size, there are two main challenges in the airline crew pairing problem. The first is the generation of pairings. Due to the complex non-linear cost structure and the various rules which dictate feasibility, it is not easily formulated mathematically and characterization of good or even all feasible pairings mathematically is extremely difficult. Dealing with all rules, safety regulations, union agreements and company policies while adhering to sound economics is a large problem. Recall the rules described in section 1.1, such as the 8-in-24 rule.

The second challenge is the selection of an optimal set of pairings. This is formulated as a set partitioning problem. The set partitioning problem is a hard problem to solve, and in the context of the airline crew pairing problem, the number of variables is huge as well as a considerable number of constraints. Aside from the wide array of rules and regulations airlines have multiple crew-bases and need to adhere to agreements placing specific fractions of flying time at each crew base. Crew-base constraints are in place as a result of collective agreements and crew distributions. This means that the flying time must be distributed over a set of crew bases within prescribed guidelines. These rules add one or more constraints per crew-base to the set partitioning problem. Not only does it make the set partitioning problem more difficult to solve by deviating from the pure set partitioning problem and adding more constraints, it makes the optimal solution more complex. These factors further complicate the enormous number of decision variables (from pairings) that are generated and the large amount of constraints (legs) that make this difficult to solve. The difficulty of a set partitioning problem appears to increase more with increased constraints (number of legs) rather than increased variables (columns) [5]. The solution which comes out of this problem also requires to be an integer solution so integer programming techniques need to be used rather than linear programming techniques.

1.4 Industry

The schedule is relatively firm and does not frequently change drastically from one planning period to the next. As such the crew planning problem receives a lot of attention by dedicating many man-hours and resources to finding the best possible solution. The resource planning problem for airlines is an extremely large problem and extends beyond just crew scheduling. Large airlines operate around 100 flights per day using their small fleets and anywhere between 500 and 1000 flights for their larger fleets [5]. This is why the airline resource planning problem is generally broken down into stages. At each stage different considerations are applied and usually an entire department within an airline works to achieve an 'optimal' solution. A common way in which tasks are divided is shown in figure 1.3.

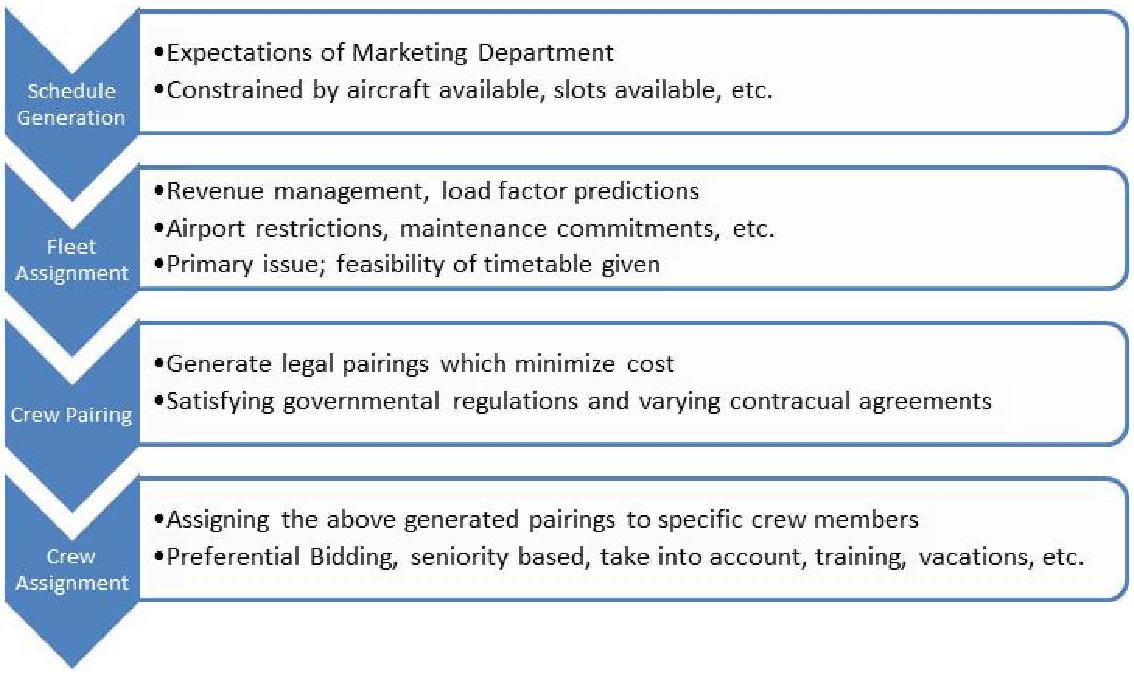
Schedule Generation
•Expectations of Marketing Department
•Constrained by aircraft available, slots available, etc.
Fleet Assignment
•Revenue management, load factor predictions
•Airport restrictions, maintenance commitments, etc.
•Primary issue; feasibility of timetable given
Crew Pairing
•Generate legal pairings which minimize cost
•Satisfying governmental regulations and varying contracual agreements
Crew Assignment
•Assigning the above generated pairings to specific crew members
•Preferential Bidding, seniority based, take into account, training, vacations, etc.

Figure 1.3: The four stages of the airline resource planning process

As illustrated in figure 1.3 four stages of the planning process include: timetable construction, fleet assignment, pairing generation and crew assignment [5].

A department in charge of timetabling makes decisions regarding what destinations to serve, which days and what time flights are to be operated. In these decisions marketing demand forecasts, flying resources, slot and gate availability, landing rights and network constraints are considered. The goal is to serve markets which are most in demand. The output from this department is a dated list of flights to be operated.

After the timetable has been set the fleet has to be assigned to flying the proposed timetable. Here specific fin numbers (aircraft identifiers) are assigned to legs in the timetable. In this stage it is important to consider the feasibility of flying the timetable, the projected demand for flights, the maintenance commitments and requirements of aircraft and operational limitations. At this stage these first two departments are required to have maximized the expected revenue of the given timetable with set aircraft routing plans.

Pairing Optimization then takes the firm timetable and aircraft flows, and applies the crew rules; CBAs, regulatory and airline. It tries to optimize costs such as the cost of crews operating flights, hotels and deadheads. This step requires detailed information on legs, aircraft assignments, hotel costs, deadhead costs and ground transportation costs and options. The output of this department is a set of anonymous pairings which cover all legs in the schedule at lowest cost.

Pairing Assignment involves taking the generated pairings and assigning them to individual crew members. This is generally done through preferential biding, where crew members can bid their preferences and the schedulers optimize over these bids. This takes into account set pairings, more crew rules, crew vacation schedules, training activities, seniority and crew preference bids. This department puts out a set of customized lines for each crew member outlining a legal combination of flying and training activities for that month.

Some airlines may break down the resource planning process differently. Barnhart et al. [9] for example break the problem down to schedule generation, fleet assignment, maintenance routing and crew scheduling. This scheme covers the same tasks and factors, the only difference is the grouping of tasks. Regardless of how they are grouped, the stages are solved sequentially due to their respective sizes, complexities and unique challenges [9]. There is some communication up the process, but it's limited. For example, if a particular sequence of flights cause consistent crewing and aircraft problems, this may be communicated to the timetabling department to try to avoid these situations. Some relatively small amendments may be made later on in the planning cycle which then have to be accounted for in fleet assignment and crew solutions. It is therefore desirable to be able to complete the crew planning aspect in as little as time possible in order to have the solution cover the most recent timetable and aircraft assignment. Amendments can be costly due to the difficulty of updating a current solution and crew assignments.

Some of the commercial systems and approaches used by airlines include the Carmen Crew Pairing System which was taken over by Jeppesen which is owned by the major airframer Boeing, Sabre Airline Solutions, AD-OPT by Kronos and Unisys. Some airlines have their own crew planning implementations, such as Lufthansa Systems by Lufthansa and American Airlines had American Airlines Decision Technologies (AADT). Some airlines still create schedules manually but they tend to be worse than schedules created by optimizers and this approach is only practical for extremely small fleets.

The solution to the crew pairing optimization problem is characterized by a set of pairings which is practically implementable, operationally robust and economically sound, covering all legs. From the perspective of airlines the most important goals of implementing advanced crew scheduling procedures are: to find cost savings, to find solutions quickly, to be able to implement and operate the schedule effectively and to provide crews with enough flying hours.

As previously mentioned speed is another important parameter. This would allow an airline to move the planning/scheduling cycle closer to execution dates which allows for the capture of most recent changes by other departments. Schedules are not always static so if the pairings can be generated as close as possible to the operational period, the changes to the schedule can be dealt with in the best possible manner. Performing last minute changes can be expensive because these extra trips are rarely efficient and often things like deadheads are required to make them feasible.

For robustness reasons, it is preferable to keep the crew on the same plane. If plane changes are required it is generally desirable to increase the amount of time of ground time between legs but care must be taken not to increase it so much that the pairing becomes extremely expensive.

1.5 Focus of This Thesis

This thesis deals with the pairing optimization for the airline crew pairing problem. The goal is to improve the tradeoff between computation time and solution quality. This allows for the application of this work to save time and money for airline crew schedulers, creating better schedules and operations, which will improve the passenger's flying experience. We propose an enumeration approach for the generation of duty periods, as well as a hybrid enumeration-column generation scheme. The enumeration approach is improved through the application of a different search techniques in combination with pre-allocation and pre-processing. The column generation uses various different sets of previously generated pairings to solve the restricted master problem and generate better pairings, yielding better solutions. Various rules and criteria are applied and tested for the selection of enumerated pairing sets. In addition the column generation approach is modeled using two different cost functions on the arcs of the sub-problem. The proposed approaches are all tested on a real case study in order to compare their performance.

This thesis is organized as follows: Chapter 2 presents a review of relevant literature related to solving the airline crew scheduling problem. Chapter 3 describes the formulation and the proposed approaches. Chapter 4 outlines the computational results achieved through the application of the proposed approaches. And chapter 4 provides conclusions based on the work presented in previous chapters and suggests some areas of future research in the field.

Chapter 2

Literature Review

There are several classes of literature in the field of airline crew scheduling. Some of the literature studied includes general information literature, academic surveys, and literature on the two main approaches used to solve the crew pairing problem, subset approaches and global approaches. Some of the work within these categories covers subjects such as column generation, network approaches, and heuristics approaches. There are two pertinent areas to solving the crew pairing optimization problem, they are pairing generation and pairing selection (optimization). Pairing selection is a set partitioning problem which is a field of study on its own. In this section set partitioning will only be covered in the context of airline crew scheduling and will not cover set partitioning literature itself.

2.1 General Literature and Surveys

Arabeyre et al. [6] are frequently cited as one of the first survey papers on the subject of airline crew scheduling. Arabeyre et al. [6] survey industry approaches used at that time, they contrast various airlines documenting the different characteristics of problems faced by individual airlines. Differences in constraints, requirements, rules, network structures, etc. They discuss methods airlines used to generate their duties and pairings, how each airline costs pairings, matrix size reduction techniques, optimization techniques applied and rostering linkage procedures. These techniques are all varying heuristics using rules of thumb from industry experience and some predetermined logic algorithms. All airlines used heuristics to generate duties and pairings and optimization methods of the day were sometimes applied in the set partitioning or set covering solution. Airlines then also faced size issues and had to reduce the matrix size before attempting to optimize, this also was done heuristically. If the set partitioning or set covering problem was solved heuristically this reduction step can be skipped. In this era, pilots in Europe tended to be paid a salary, compared to North America where pilots tended to be paid based on flight hours. Arabeyre et al. [6] show us that in its infancy, the crew scheduling problem is solved either through heuristic approaches or through exact approaches of restricted problems, not much has changed with regard to this fact. This is seen in more recent pieces of general literature including Andersson et al. [5], Barnhart et al. [9] and Gopalakrishnan and Johnson[24]. Gopalakrishnan and Johnson [24] provide an especially good survey of current methods used for the problem itself and for algorithms which support the approaches applied.

Andersson et al. [5] focus on the crew pairing problem. It reviews the crew pairing problem and the formulation and discusses optimization methods previously applied. They also discuss a commercial pairing construction system called the Carmen Crew Pairing System.

Ernst et al. [20] provide a review article covering staff scheduling problems from optimization to rostering. It covers many areas in transportation systems including airlines, railways, busses and mass-transit systems, and many other areas as well. It points out general solution approaches as outlined in close to 200 papers. Ernst et al. [19] also provide an annotated bibliography of personnel scheduling and rostering, which references over 700 pieces of literature. These two papers provide an excellent source of a wide range of papers on the subject.

2.2 Structure of Solution Approach

Because the pairing optimization problem is such a large one, containing thousands of flights over a planning horizon, it can be tackled by breaking it down into three stages. These three stages, the daily, weekly and fully dated problem, are solved sequentially.

2.2.1 Solving the Daily Problem

In solving the daily problem the entire schedule is considered but it is reduced to fit the assumption that the same flights are operated every day. As such, only a solution of a single day is required and only pairings as long as the maximum pairing length would need to be generated. The repetition of the solution to this daily problem would cover the entire monthly schedule. This approach makes the problem more tractable because it only considers the number of legs which are operated on a single day and adds regularity to the pairings which is operationally desirable [4].

To convert the fully dated problem to a daily problem, the flight legs which are repeated less than four days per week are removed from the problem. A great deal of research focuses on the solution of the daily problem because it is assumed that this is the basis of a good solution and this is the most difficult problem to solve [5]. The solution of the daily problem is especially useful for solving a largely regular schedule, however in the case of an irregular schedule, the benefits are greatly reduced.

In the daily problem the cost of a pairing, assuming salaried pilots, is the number of days which the pairing covers, because you would require that many crews to operate that pairing every day. In the case of pilots which are paid on credit hours, the cost is the number of credit hours of the pairing as calculated by equation 1.2 on page 7.

Solving the set covering problem of this problem yields an optimal solution which executes the same pairings every day. However, these solutions are not always optimal because the optimal solution may include legs being covered by different pairings on different days. The linear relaxation may give such a solution to this problem. In the case presented by

Andersson et al.[5] an example of this is presented. This optimal solution can be found by solving the linear relaxation of the set partitioning problem. The linear relaxation gives a lower bound on the non-daily solution but there is no guarantees about it being a good bound [5]. The impact of using one or the other solution for the next step is not mentioned.

The daily problem can be solved as a standalone problem or it can have a subset approach applied, such as in section 2.3.

2.2.2 The Weekly Problem

The weekly problem assumes the schedule is repeated on a weekly basis to make up the full month's schedule. The aim is to utilize the daily solution as a 'warm starting point'. The daily problem solution is taken and modified to account for weekly exemptions to the assumed daily schedule. Again, the repetition of the weekly problem will cover the assumed fully dated problem. In many North American airline schedules, the daily problem is relatively consistent throughout the week but only varies on the weekends so the changes required to go from a daily solution to a weekly solution are minimal. In a much more irregular schedule, the changes will be much more drastic reducing the value of the daily solution [5].

2.2.3 The Monthly or Fully Dated Problem

For the fully dated problem, the repeated weekly problem is modified to fit the fully dated schedule. For North American carriers this generally means adjusting for changes in timetables and holiday schedules [5].

2.3 The Subset Approach

Traditionally, commercial solvers have used local optimization techniques, employing various subset approaches. The subset approach is the most commonly used approach to solve the airline crew pairing problem. It is employed by the TRIP (Trip Reevaluation and Improvement Program) system used by American Airlines and Continental Airlines, and the TPACS system used by United Airlines.

It uses an iterative local improvement algorithm which begins by using an initial workable solution. This solution must be feasible to the set partitioning problem. The initial solution can come from a similar historical solution, in some cases a specialized initial solution generator is used, or a solution is manually constructed [4], [9]. Most approaches to initial solution generation use a previous solution as a starting point, as usually schedules only see minor changes from one planning period to the next. From the initial solution a sub-problem is selected. Gershkoff [23] just uses 'some heuristic' because intelligent selection is surprisingly difficult and suggests using random selection is the best approach. Simply selecting high cost pairings to improve upon doesn't work [23]. The size of the sub-problem which is selected can vary and is usually determined by the experience of a scheduler, and can depend on the stage of optimization. For example, if improvements become scarcer, the sub-problem size may be increased in order to make larger changes to the solution possible such that the local minima can be escaped. A system such as the TRIP system, selects five to ten pairings per iteration [4] and the Trip Pairing for Airline Crew Scheduling System (TPACS) selects only two or three pairings [25]. The selected pairings are then broken down, and all the legs which are covered by these pairings are then used to generate pairings.

The methods applied for the generation of pairings are discussed in section 2.4.

In cases where five to ten pairings are selected, such as with the TPACS system, it is possible to totally enumerate all pairings. Once generated, all pairings must either already be known to comply with all rules or they must be checked. In addition, cost can only be calculated once the pairings are completed because the nature of the cost structure makes it such that small changes can determine which cost factor dominates (see equations 1.1 and 1.2). Once all pairings are confirmed to be legal and the costs are known, they are entered into the set partitioning problem. Because the original pairings are also in the set partitioning problem, and are a feasible solution, they are an upper bound on the optimal solution. The optimal solution must then be found and entered into the overall solution. Because of this upper bound, the overall solution's cost can only improve with each iteration.

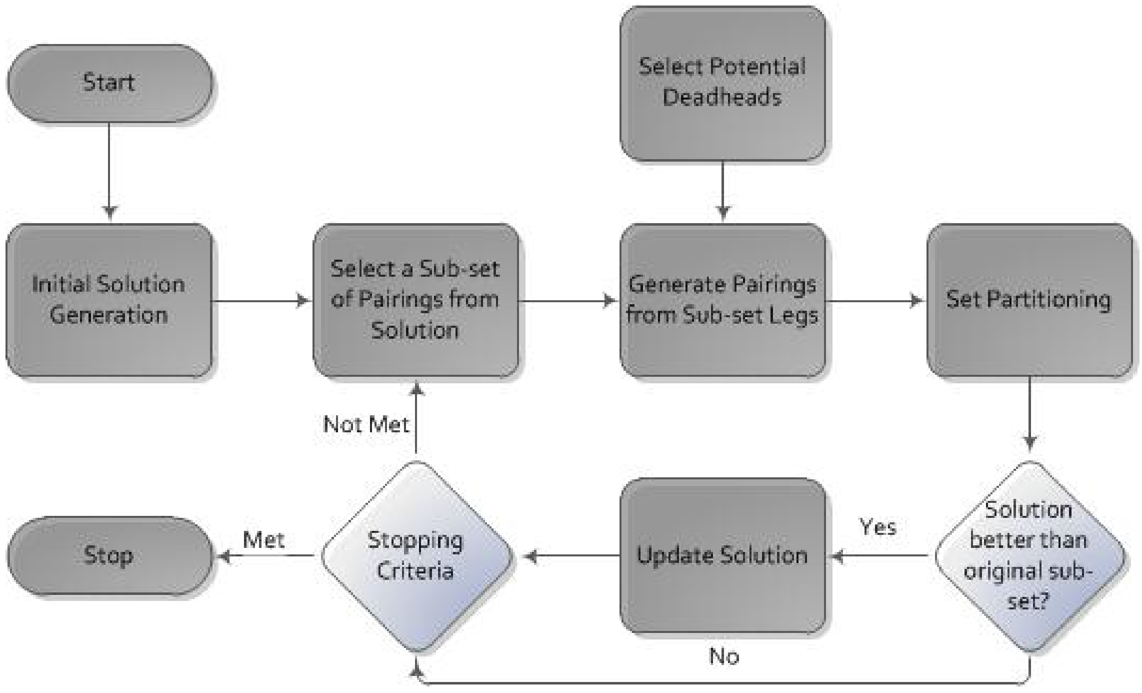
Start
Select Potential Deadheads
Initial Solution Generation
Select a Sub-set of Pairings from Solution
Generate Pairings from Sub-set Legs
Set Partitioning
Not Met
Stop
Met
Stopping Criteria
Update Solution
Yes
Solution better than original sub-set?
No

Figure 2.1: Block diagram of the subset optimization approach

As previously mentioned one of the systems using this approach is the TRIP system of American Airlines and Continental Airlines [3], [25]. TRIP is a program which is based on the TPACS program. Anbil et al. [3] report work based on TRIP and report major improvements. TRIP solves the set partitioning problem by using a hybrid algorithm combining linear programming techniques and Lagrangian relaxation. This approach calculates a set of Lagrangian multipliers which are then used in order to efficiently find the solution to the LP [24]. ALLPS by Unisys is another program which has been likened to TRIP [24]. This system was used at Northwest Airlines and USAir.

TRIP did not include base constraints, but TPACS did. It wasn't until Anbil et al. [3] that it considered these types of constraints. The difference is the formulation of the set partitioning problem. The TRIP set partitioning problem is:

$$\begin{aligned}
\min \quad & \sum_{p \in P} c_p x_p \\
\text{st} \quad & \sum_{p \in P} a_{ip} x_p = 1 \quad \forall i \in I \\
& x_p \in \{0, 1\}
\end{aligned} \tag{2.1}$$

where c_p is the cost of pairing p in the set of pairings P, a_{ip} is 1 if leg i is covered by pairing p, and x_p is a binary decision variable taking the value of 1 if pairing p is in the optimal solution and 0 if it is not. This formulation can be turned into a base constrained set partitioning problem by adding the following constraint.

$$L_j \leq \sum_{p \in P} b_{jp} x_p \leq U_j \quad \forall j \in J \tag{2.2}$$

where b_{jp} is the value of resources required from base j in order to execute pairing p, L_j is the lower limit of resources that must be used at base j and U_j is the upper limit or resources which can be used from base j. TRIP has been developed to use a heuristic to adjust weighting of the costs of the pairings to steer solutions to satisfy base constraints. The formulation which TPACS uses is an elastic set partitioning model [25]. This model is formulated as follows:

$$\begin{aligned}
\min \quad & rx + ps^- + ps^+ + ev^+ \\
\text{st} \quad & Ax + Is^- - Is^+ = 1 \\
& Dx - Iv^+ \leq M \\
& x_p \in \{0, 1\} \\
& s^-, s^+, v^+ \geq 0
\end{aligned} \tag{2.3}$$

Where A is the matrix indicating if given legs are in specific pairings, exactly like a_{ip} above, r contains the costs of the pairings, D is the parameter containing base resource constraint data, where there is a row for each base in the problem, M is the upper limit on base resources. As above x is the binary decision variable indicating the selection of optimal pairings, s^- and s^+ permit the violation of the set partitioning and base resource constraint at a penalty cost of p. The last term in the formulation, v^+, allows for the violation of the base resource constraint at penalty cost e.

The approach by Graves et al. [25] is also a sub-problem approach but overlaps somewhat with column generation. They solve the above flexible set partitioning model to obtain data to generate a limited set of pairings to add to the eventual integer program. They begin by applying a concept called interior enumeration, as described in section 2.4.2, to each leg, to obtain an initial solution. The problem is solved, likely yielding a solution which covers some legs more than once. The set of pairings selected are then reduced to a set which offers disjoint coverage of legs. The legs which are then not covered are used as seed legs to generate more pairings. These pairings are then added to the flexible set partitioning problem, which is solved again. This process is repeated until a disjoint solution is found. The generation of pairings in this approach is already limited, but can be limited even further using system parameters they offer. At this stage they perform local optimization (2-OPT and 3-OPT) taking two or three pairings at a time and performing total enumerations

Even though a solver such as TRIP can find a global optimal solution for smaller instances, one of the problems which can be experienced with this approach is that the solution may get stuck in a local optima. To get out of this local optima larger changes to the solution may be needed. As alluded to before, increasing the size of the sub-problems would be one way to exit these local optima, and this technique may be employed once the solution remains unchanged for a period of time. In the case of the TRIP system, the sub-problem optimizer can only handle problems up to 100 segments and 10,000 columns [4]. Another way to avoid local minima is to accept solutions which do not improve the solution. Anbil et al. [3] have employed two of the techniques mentioned here, they have implemented increasing the size of sub-problems, and a customized approach to accept solutions which do not improve the solution. They have also found that often there are alternative optimal pairings. In simultaneously considering multiple solution paths they are able to minimize the impact of local minima. Unfortunately, the do not provide very many details on these approaches.

Some additional previous work using this iterative scheme are presented by Housos and Elmroth [26], the authors present an iterative scheme which is reported to be very successful, Ball and Roberts [8] and Etschmaier and Mathaisel [21].

2.4 Pairing Generation

Literature rarely supplies specifics in this area. Pairing generation is mostly done through some version of an enumeration technique. There are four options for exhaustive pairing generation; total generation of all pairings covering all legs, total generation for a limited set of legs, partial generation for the total problem and partial generation for a partial problem. Total enumeration is only possible for small problems, and is hence used in local optimization approaches in section 2.3.

Gershkoff's [23] approach is to order legs by station and time, beginning with the earliest departure from crew base station, the algorithm searches for the earliest legal connection at the arrival station of the last segment added. If the arrival station of the last segment is the same as the first departure station a pairing has been created, and is written to file. If the addition of a segment leads to an illegality the branch is terminated at that segment. This strategy was improved by Minoux [30] with a heuristic pruning algorithm.

2.4.1 Partial Enumeration

Partial Generation is another approach and it aims to generate only good pairings for all legs. This is difficult because it is difficult to identify what makes a good pairing and what pairing is going to help the overall problem. This is where column generation and global approaches comes in. These approaches attempt to remedy this.

One partial enumeration approach which is relatively straight forward is described by Baker et al. [7]. First legs departing at a crew base are identified to be used to seed pairings. These legs are now considered partially completed pairings. The set of legs which do not begin at a crew base is ordered by departure time, the first leg from this set is selected and a search for a partially completed pairing to attach it to begins. Usually several partially completed legs can be identified and five different criteria for selecting which one to use are proposed. These criteria are selecting the first feasible pairing, the first feasible pairing with the same aircraft flight number, the first crew in takes the first flight out, last crew in takes the first flight out or minimum cost assignment of adding that leg to the partial pairing. The criteria specifying first or last crew in takes first flight out, maximize and minimize the difference between departure time of the leg and the arrival time of the partial pairing respectively. Baker et al. [7] reports that the last crew in takes first flight out and minimum cost assignments yield the best quality solutions.

2.4.2 Interior Enumeration

Interior enumeration is rarely seen in literature but one instance is shown by Graves et al. [25] and is worth mentioning. A leg is taken and then connections are made backwards until a crew domicile is reached. To do this one could take the approach by Gershkoff [23] mentioned above and modify it slightly. Combining with a forward enumeration you can create pairings. This is designed to create pairings to include specific legs. This feature may be very useful in improving solutions which have legs that are difficult to cover.

2.5 Global Approach

Anbil et al. [4] describe a global approach which is motivated by the desire to overcome the local optima problem of a sub-problem approach. They consider the entire problem instead of focusing on smaller pieces of the problem. The approach does not guarantee finding the global optimum but takes into consideration the entire problem and constructs a solution without using the initial solution.

The approach begins by exhaustively generating millions of pairings. Anbil et al. [1] begin with the generation of 2 million columns. These pairings are then entered into the set partitioning problem. Solving this integer problem would be impossible. Instead the LP relaxation is solved, with the objective function minimizing the excess cost of all pairings.

The LP relaxation may still be difficult to solve. In the case described by Anbil et al. [4] 12 million columns were generated and then reduced to 5 million before the set partitioning LP relaxation was attempted. The approach which was used to find the optimal solution was the SPRINT approach by John Forrest. Once the LP solution has been found the best 10,000 to 15,000 columns are taken and entered into the integer set partitioning problem. These columns are selected based on their reduced costs. In solving the integer problem, for small set partitioning problems, simply rounding linear variables which were closest to one yielded good solutions. However, for larger problems, even a regular branch-and-bound approach failed. A special code which works with branching on follow-ons was developed to reduce the up to 15,000 columns down to 3,000. When the number of columns is reduced to this stage, it is solved using an MIP solver. The approach presented by Anbil et al. [4] was implemented for the schedules of two aircraft types at American Airlines and realized 1.2 million dollars per year in savings over the solutions from the TRIP system.

2.6 Column Generation

In the column generation approach to the airline crew scheduling problem, the master problem is the set partitioning problem. This ensures that all legs are covered and in certain cases that base constraints are met. The sub-problem is a typically a shortest path problem. There are different ways the sub-problem network is modeled. Two possible network structures are the leg-on-node network, and the leg-on-arc network [17]. Minoux [30] and Desrosiers et al. [18] use leg-on-arc networks. However, networks can also be created using duty periods. For instance Barnhart et al. [11], Lavoie et al. [28] and Vance et al. [33] use duty periods instead of legs to create networks. The process begins by exhaustively generating all legal duty periods. These duty periods become the nodes of the network. The arcs on the network are legal overnight rest connections between duties, which could make up a pairing. Once this has been established, a set of pairings must be generated which constitute a feasible solution to the set partitioning master problem. A good way to determine a good starting solution is to have an experienced scheduler create one by hand, using a previous solution.

$$
\begin{aligned}
\min \quad & \sum_{p \in P} c_p x_p \\
\text{st} \quad & \sum_{p \in P} a_{ip} x_p = 1 \quad \forall i \in I \\
& 0 \geq x_p \leq 1
\end{aligned}
\tag{2.4}
$$

This allows for the first solution to the restricted master problem (equation 2.4) to be found by applying the revised simplex method to the continuous linear relaxation of the set partitioning problem. The dual variables of the optimal solution are then applied to the cost of the nodes of the sub-problem network. The dual variables correspond to all of the flight segments, as such the cost of the node is set to be the negative sum of the dual variables which are in the duty period represented by the node. A shortest path problem on the network is then solved to get negative reduced cost pairings. A heuristic solution to the constrained shortest path problem is discussed in Anbil et al. [3], which was then adopted by IBM and Sabre Decision Technologies for use at American Airlines and USAir. A subset of these pairings are then added to the restricted master problem, and the process is repeated until an optimality criteria is reached. This criteria is that no more negative reduced cost pairings can be found. At this stage the binary integer solution to the set partitioning problem is found by branch and bound techniques. Lavoie et al. [28] provide computational results of problems up to 329 legs, which was solved in 1250 seconds finding a 12% improvement over manual methods, Vance et al. [33] provide results of problems of up to 174 legs. Desrosiers et al. [18] report a 5% savings over experienced manual schedulers in cases spanning one to two weeks, with 300 to 500 legs and sometimes 1000 legs. In a case of 313 legs, Desrosiers et al. [18] report a 2.7% savings and in a case of 282 legs a 9.1% savings over other optimizers and claim optimal solutions using the GENCOL software.

Barnhart et al. [11] apply the column generation technique to the long-haul crew pairing problem. Their article shows that these problems are solved well by this approach. The long-haul problem has fewer options for connecting flights and duty periods and even though the flights are long the network size is small when compared to short-haul problems. This allows for a sub-problem network to be made of legs instead of duties, and often a duty period will only be one leg long. In this class of problem TAFB is a good approximation for the cost of a pairing.

Graves et al. [25] present a method which is a form of column generation. Along with local optimization procedures, it incorporates an algorithm to heuristically identify legs which should be used to generate more pairings. This is done iteratively much like the approach described in section 2.3.

It is reported that few commercial solvers use the column generation approach. It is used in ALTITUDE by Ad Opt [16] and has been used by Air France [28]. Commercial programs favor large-scale heuristics or the subset approach.

Some additional literature which covers column generation include articles such as Anbil et al. [2] , Chu et al. [14], Crainic and Rousseau [15], Gamache et al. [22] and Lübbecke and Desrosiers [29].

Chapter 3

Formulation and Solution Methodology

This thesis focuses on exploring efficient approaches to generate pairings. The efficiency sought is characterized by lower cost and solution time. The starting point is a brute force approach to generation which is developed and improved upon. In brute force enumeration techniques, the literature suggests four options; total enumeration of the entire problem, total enumeration of a limited part of the problem, partial enumeration for the total problem and partial enumeration for a partial problem. These four options are really only two different approaches: total enumeration and partial enumeration. The only reason they are stated as four is that they can be applied to different sized problems. In the following work, the test cases can be interpreted either as an entire small problem or a subset of a larger problem, but deals with total enumeration. Techniques encompassed by the genre of partial enumeration will only be dealt with in the context of column generation.

Partial enumeration techniques require a certain level of expertise in airline crew scheduling. This enumeration approach requires rules-of-thumb to limit which areas of the feasible pairings space should be explored and which can be omitted. To obtain good results with this technique, familiarity with such factors as network structure, historically difficult legs to cover, specific crewing cost structures and what makes a poor pairing structure is required. Because airlines can vary drastically in network, pay and cost structures, as well as operational desires, it is difficult to characterize strictly good pairings. The only generalization that can be made is that it is usually best to have the flying time exceed the minimum guarantee, and to have the flying spaced close enough together to avoid having to pay on the duty or trip rig. In addition, some known poor pairings may be required to allow the set partitioning problem to achieve a better global solution. As such this work deals with total enumeration methods for the generation of duties, and for pairings total enumeration is attempted as well as a hybrid column generation approach.

3.1 Duty Periods

A generic search algorithm (algorithm 1) was taken as a basis for the development of a Depth First Search (DFS) strategy. With a small modification, in its most basic form it achieves the goal of total enumeration of all duty periods.[1] The criteria for creating the set, $LIST$, and for selecting node i in algorithm 1 determines the nature of the algorithm. In both cases nodes are added to the end of the set $LIST$. In Breadth First Search (BFS) algorithms node i is always selected from the front of $LIST$ and in DFS algorithms, i is always selected from the end of $LIST$.

In the duty period enumeration algorithm a graph is created in which the node set is defined by all legs in the schedule and the arc set is defined by possible connections between legs subject to congruency in time and space. The algorithm is repeated using each leg as a source node in order to create all possible duties. This must be done because duties do not have to start at a crew-base, there are no restrictions on the starting leg, therefore any leg can be used to begin a duty period. The algorithm begins at a source node and seeks to make a connection to the earliest available leg occurring after the current leg in the algorithm. Parameters such as duty starting station, duty ending station, duty start time, duty end time, total block time, total TAFB, operating day and the number of legs are tracked throughout the generation.

[1]Partial enumeration would require complicating rules and criteria.

Algorithm 1 Basic search algorithm

```
1: begin
2: unmark all nodes
3: mark node $s$;
4: $pred(s) := 0$;
5: $next := 1$;
6: $order(s) := next$;
7: $LIST := \{s\}$;
8: while LIST$\neq\emptyset$ do
9:     begin
10:    select a node $i$ in $LIST$;
11:    if Node $i$ is incident to an $arc(i, j)$ where $j$ is uncovered then
12:        begin
13:        mark node $j$;
14:        $pred(j) := i$;
15:        $next = next + 1$;
16:        $order(j) := next$;
17:        add node $j$ to end of $LIST$
18:        end
19:    else
20:        delete node $i$ from $LIST$;
21:    end
22: end
```

Updating duty ending stations and duty arrival time allows the search algorithm to determine candidate connections for potential future legs. Maximums on the total block time, TAFB and number of legs can be used for the termination of the search by checking if the current duty has reached its maximum limit on these parameters. Duty start and end station, start and end times, and operating day are parameters which are required to build pairings after the duty generation.

When searching for connecting legs, viable legs are characterized as having a departure time greater than the arrival time of the current leg plus the minimum sit time and less than the arrival time plus the maximum sit time. Naturally, the next leg must depart the same station as the previous leg arrived at.

A general pseudo-code for the DFS duty enumeration algorithm described above is shown in algorithm 2 on page 33.

The two criteria for stepping back in the search tree are, if the duty time reaches its maximum allowable limit in any of the aforementioned duty termination parameters, or if no more legal connections are available in the schedule after the current node. The deviation from a strict DFS comes in the recording of duties. A DFS search is intended to find all reachable nodes given a starting node. In this case the search is designed to determine not only all reachable nodes, but all paths to reachable nodes as well. Every time a node is added to the sequence, the whole sequences is recorded as a unique duty. This means that duties which are subsets of other duties are generated as well. For example, in a case where legs 1, 2 and 3 are a legal flying sequence, we must find a duty flying legs 1 and 2 as well as a duty with flying sequence 1, 2 and 3. Algorithm 2 adds one duty at a time as the algorithm stepped through the coding, causing many iterations of a loop to occur.

In order to save computational time the algorithm was furthered by switching to a BFS like approach. The BFS algorithm was programmed to record all of the potential next legs at once, significantly reducing the number of loops required. Similar to the criteria for adding entries to $LIST$ in the generic BFS, the newly created pairings are added to the end of the list of created duties. The $LIST$ analogue in algorithm 4 is a series of duties, not

Algorithm 2 Depth First Search Duty Generation Algorithm (Part 1, cont'd on page 34)

```
1: begin
2: $LEGS$ :=all legs
3: sort $LEGS$ in order of $departure\_time$ and assign $indices$
4: while LEGS≠∅ do
5:     begin
6:     select $s$ from $LEGS$;
7:     $start\_time$ := departure time of leg $s$ - briefing time required
8:     $current\_station$ := arrival station of leg $s$
9:     $current\_time$ := arrival time of leg $s$
10:    $max\_time := start\_time + max\_duty\_time$
11:    $legs := 1$;
12:    $LIST := \{s\}$;
13:    write $LIST$ and vital statistics to $Duties$
14:    $possible\_legs := \{LEGS : arrival\_time \leq max\_time - debrief\_time \cap departure\_time \geq current\_time + min\_sit\_time\}$
15:    $Last = \emptyset$
16:    while LIST≠∅ do
17:        begin
18:        $conenctions := \{possible\_legs : departure\_station = current\_station\}$
19:        $conenctions := \{connections : departure\_time \geq current\_time + min\_sit\_time\}$
20:        $conenctions := \{connections : departure\_time \leq current\_time + max\_sit\_time\}$
21:        if Last≠∅ then
22:            begin
23:            $connections := \{connections : indeces \geq Last\}$
24:            end
25:        if connections≠∅ then
26:            begin
27:            add first leg in $connections$ to end of $LIST$
28:            update $current\_station$
29:            update $current\_time$
30:            $legs := legs + 1$
31:            write $LIST$ and vital statistics to $Duties$
32:            $Last = \emptyset$
33:            end
```

Algorithm 3 Depth First Search Duty Generation Algorithm Continued (Part 2)

```
34:             else
35:                 begin
36:                 $Last$ :=index of last entry in $LIST$
37:                 remove last entry from $LIST$
38:                 update $current\_station$
39:                 update $current\_time$
40:                 $legs := legs - 1$
41:                 end
42:         end
43:     end
44:     remove $s$ from $LEGS$
45: end
```

just recorded nodes. The algorithm works its way through duties, to check for all possible connections. If there are $n > 0$ connections available the current duty is then copied $n + 1$ times, recording the unaltered duty and adding n duties, each with a unique connection. The new connections are added to the end of the set, and the algorithm moves to the next duty repeating the process. This property also simplifies the back-stepping element of algorithm 2 by eliminating the need for backtracking and recording which nodes have already been connected to specific nodes.

Algorithm 4 is not a strict BFS, because a BFS would have $LIST$ in algorithm 1 grow at the end of the vector, and selecting node i from the front and then removing elements from $LIST$, in a First In First Out (FIFO) manner. In algorithm 4 the analogue to $LIST$ is not reduced in size but just stepped through with an index. The looping criteria therefore states as long as this index is less than or equal to the number of entries in the list continue the search.

The area in which algorithm 4 is to save time over algorithm 2, is that the BFS algorithm records all possible connections at once. In an effort to make this even faster all legs were preprocessed to find legal connections *a priori*. This created a list of all legs, with corresponding indices of possible connections associated to it. This allowed the algorithm to simply access the current leg in this data set and return all predetermined connections in

Algorithm 4 Breadth First Search Duty Generation Algorithm

```
1: begin
2: $LEGS$ :=all legs
3: sort $LEGS$ in order of $departure_time$ and assign $indices$
4: $index\_current := 1$
5: $index\_duties := 0$
6: while LEGS$\neq\emptyset$ do
7:     begin
8:     $current := \{s : s \in LEGS\}$;
9:     $start\_time :=$ departure time of leg $s - briefingtime$
10:    $current\_station :=$ arrival station of leg $s$
11:    $current\_time :=$ arrival time of leg $s$
12:    $max\_time := start\_time + max\_duty\_time$
13:    write $current$ and vital statistics to $Duties$
14:    $index\_duties := index\_duties + 1$
15:    $possible\_legs := \{LEGS : arrival\_time \leq max\_time - debrief\_time \cap departure\_time \geq current\_time + min\_sit\_time\}$
16:    while index_current$\leq$index_duties do
17:        begin
18:        $current :=$ duty in position $index\_current$ in $Duties$
19:        update $current\_station$
20:        update $current\_time$
21:        $conenctions := \{possible\_legs : departure\_station = current\_station\}$
22:        $conenctions := \{connections : departure\_time \geq current\_time + min\_sit\_time\}$
23:        $conenctions := \{connections : departure\_time \leq current\_time + max\_sit\_time\}$
24:        $numb\_connects :=$ number of connections found
25:        write $current$, $numb\_conencts$ times, each instance with one element of $connections$ added to $Duties$
26:        $index\_duties := index\_duties + numb\_connects$
27:        $index\_current := index\_current + 1$
28:        end
29:    end
30: remove $s$ from $LEGS$
31: end
```

one step, instead of creating the *connections* set and checking for connections repeatedly. Additionally, pre-allocation of a matrix for the recording of the created duties was observed to have an impact on the computation speed. As such the algorithms were attempted using various sizes of pre-allocated matrices. The impact of these changes is studied later in table 4.2.

3.2 Pairings

3.2.1 Enumeration

The same search approach was attempted to generate pairings. There is an important distinction in pairing generation when compared to duty generation. That is the requirement that pairings must begin and end at a crew base and the building blocks of pairings are duty periods as opposed to legs. In the case study duty periods are assumed to be operating on distinct days. This assumption holds true due to the structure of the sample problem, but in reality it is possible for duties span days and two distinct duties can both end and the next one begin on the same calendar day. This feature would require more complicated rules with respect to continuous duty over-nights, night and red-eye flying.

Because of the exponentially many possible pairings which can be made, this looping approach was found to be impractical. This approach was abandoned in favor of an approach exploiting the structure of the data created in the duty generation phase. Matrices containing indices representing all duties are copied and matched with all other duties. This generates every combination possible for pairings. However, this includes bad pairings, infeasible pairings and physically impossible pairings. This giant set of pairings is then reduced by applying filters. These filters check for station continuity and apply duty break rules, leaving only pairings with legally connecting duties.

The first step in the approaches attempted is to organize the available duties into separate, distinct sets, based on their starting and ending base, and operating day. This first step is illustrated in table 3.1.

	One Day Pairings	Starting Duties	Ending Duties	Middle Duties
Duty Start Station	Base	Base	Non-base	Non-base
Duty Ending Station	Base	Non-Base	Base	Non-base
Operating Day	Any	Any except last	Any except first	Any except first and last

Table 3.1: Criteria for the division of generated duties to initiate the generation of pairings

One day pairings are already identified through this process and don't require any further manipulation or attention. It is important to note that the union of these new sets creates a subset of all the duties which are available. Some duty periods which for example start at an out-station, and end at a crew base on the first day of the planning period, require to be part of a pairing beginning before the current planning period in order to be used. Similarly, duty periods not ending at a crew-base on the last day of the planning period require to be part of a duty ending at a crew base outside of the planning period if they are to be used. These duties are useless for generating pairings over the specified time period and can be discarded.

This approach of creating all possible combinations is made possible because a pairing can only consist of a limited number of duties. In contrast to duty periods which can contain many legs, for example in the schedule used, there are no limitations on the number of legs permitted in a duty period, where in pairings the number of duty periods cannot exceed four. In the sample schedule, the legs are operated over four days, as such the number of duties allowed in a pairing was set to four.

For this matrix concatenation method sets of duties are created and manipulated by applying a set of filters and generating pairings in order of length, starting with one day duties and then two day duties and so on. This is largely the purpose of the procedure described above using the sets in table 3.1. The manner in which these sets are put together and manipulated in order to generate 1 to 4 day pairings is illustrated in figure 3.2. By starting with two day pairings, which can cover days 1 and 2 or 2 and 3 or 3 and 4, we can simply take sets of duties and further reduce them to be used in three day pairings. Three day pairings can cover days 1, 2 and 3 or 2, 3 and 4 and four day pairings must cover days 1, 2, 3 and 4. So two day pairings need duties which start at a base on days 1, 2 and 3, for three day pairings we can reduce this set to those duties which operate in days 1 and 2, and furthermore for four day pairings starting duties on day 1. Similar steps can be applied to all sets generated by the criteria in table 3.1. This process is illustrated in figure 3.1.

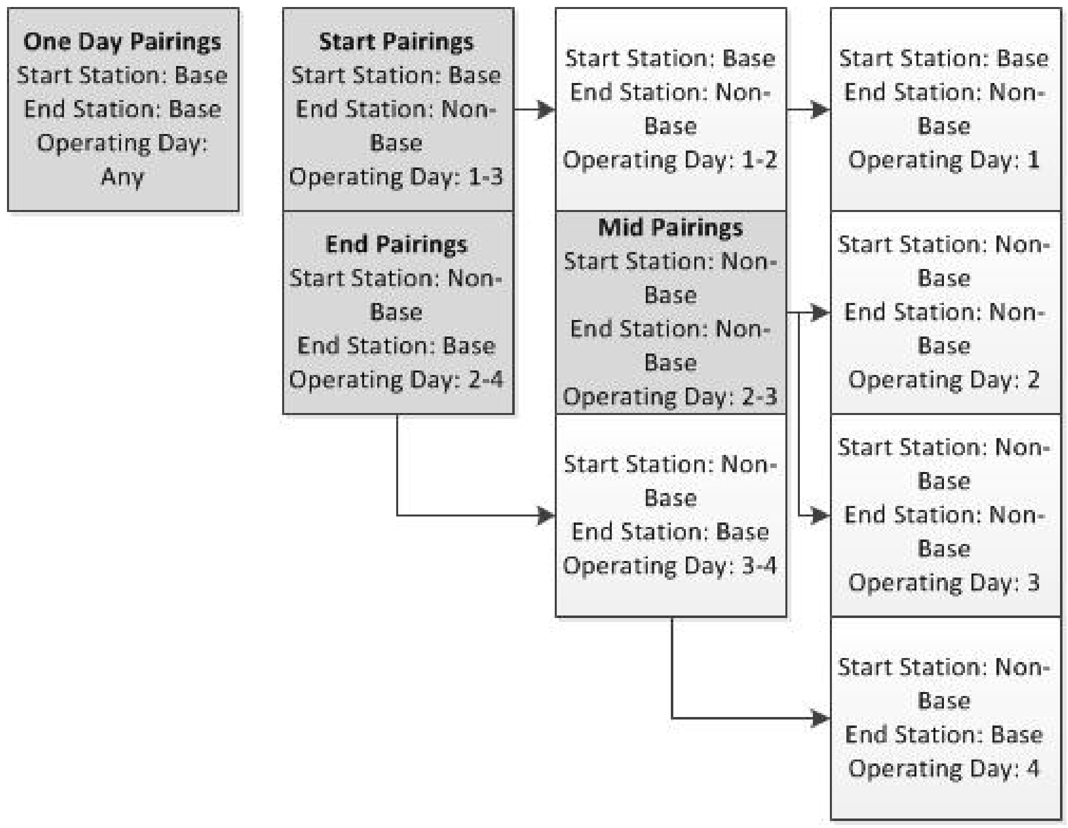

Figure 3.1: Manipulations and sourcing of matrices for pairing generation

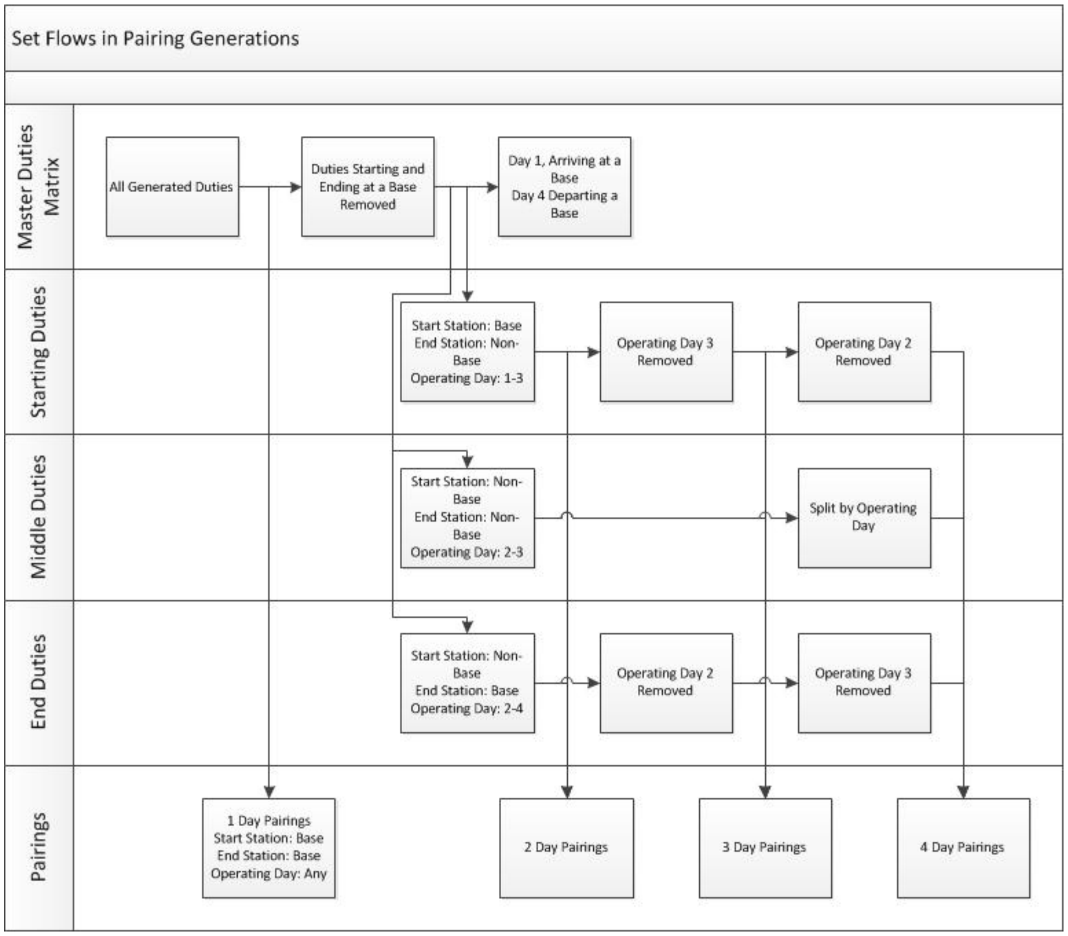
Set Flows in Pairing Generations
Master Duties Matrix
Starting Duties
Middle Duties
End Duties
Pairings
All Generated Duties
Duties Starting and Ending at a Base Removed
Day 1, Arriving at a Base Day 4 Departing a Base
Start Station: Base End Station: Non-Base Operating Day: 1-3
Operating Day 3 Removed
Operating Day 2 Removed
Start Station: Non-Base End Station: Non-Base Operating Day: 2-3
Split by Operating Day
Start Station: Non-Base End Station: Base Operating Day: 2-4
Operating Day 2 Removed
Operating Day 3 Removed
1 Day Pairings Start Station: Base End Station: Base Operating Day: Any
2 Day Pairings
3 Day Pairings
4 Day Pairings

Figure 3.2: Swim-lane representation of set flows in the generation of four day pairings

3.2.2 Column Generation

This approach was attempted in this manner because of the exponential computation time and size for pairings longer than a day or two. The computational result from the previous section demonstrated the need for longer pairings in order to get proper coverage of the schedule, for proper implementation, and likely a better objective function in the presence of penalty costs. The four day pairings generation attempt by previous approaches created too many pairings to be able to handle and manipulate the matrices. Matrices with 100 million columns for four day pairings alone were not uncommon. These matrices were so large that even the cost of pairings couldn't be calculated. One and two day pairings can be generated in an acceptable amount of time and can be handled without problem. Techniques to limit the generation of three and four day pairings without compromising reasonable solutions are sought.

The master problem is the set covering problem and the sub-problem becomes a shortest path problem. First an initial linear relaxation of the set covering problem is solved using a small set of generated columns, by the algorithms discussed above, this is the restricted master problem. Several different approaches were attempted to select these starting columns as shown later in tables 4.4 and 4.5. The restricted master problem is ensured to be feasible by adding penalized pairings containing each leg individually. Equation 3.1 shows the formulation of the problem solved here.

The solution is obtained using the CPLEX plug-in in MATLAB.

$$\begin{aligned} \min \quad & \sum_{p \in P} c_p x_p \\ \text{st} \quad & \sum_{p \in P} a_{ip} x_p \geq 1 \quad \forall i \in I \\ & 0 \leq x_p \leq 1 \end{aligned} \tag{3.1}$$

where c_p is the cost of pairing p in the set of pairings P, a_{ip} is 1 if leg i is covered by pairing p, and x_p is a binary decision variable taking the value of 1 if pairing p is in the optimal solution and 0 if it is not.The dual of this problem is;

$$\begin{aligned} \max \quad & \sum_{i \in I} y_i \\ \text{st} \quad & \sum_{i \in I} b_{pi} y_i \leq c \quad \forall p \in P \\ & y_i \geq 0 \end{aligned} \tag{3.2}$$

The optimal dual variables, y_i, correspond to all of the legs, i, which are in the schedule. These are then used to calculate the costs on the network of the sub-problem. The network is created out of duty periods generated in the earlier steps from section 3.1.

The sub-problem uses a network in which the arcs represent the generated duty periods and legal overnight rests the nodes are the starting legs and the ending legs of duty periods. This allows for the number of nodes to be limited to a maximum number equal to all the legs being considered, and the arcs to a maximum of the duties generated plus the number of legal overnight duty breaks. The overnight rests are to be determined by evaluating which legs which can be connected by legal duty breaks.

Because computation time is an important aspect in the generation procedure, a pre-processing step is taken to reduce the size of the sub-problem. The network created contains many arcs which share both first and last legs. Hence, subsets of duties which are more expensive paths starting and ending at the same legs can be removed from the network without effecting the optimal solution. For example, if there are four legs as described in table 3.2, which are used to create two unique duties as described by the sequences below.

Leg	Departing Station	Arriving Station
1	A	B
2	B	D
3	D	B
4	B	C

Table 3.2: Sample legs

1. leg 1 – sit – leg 4
2. leg 1 – sit – leg 2 – sit – leg 3 – sit – leg 4

These duties have the same start and end legs, and in a shortest path algorithm network create the same arc. Because of this if a duty period connecting stations A and C is part of the optimal solution, the lowest cost duty of these two duties will always be chosen. So this decision is made by pre-processing and treating this as one arc, equal to the lowest cost option. These duties are grouped and at each iteration, and only the lowest cost duty is considered in the network.

The sub-problem representing two day pairings contains 409 nodes and 2,715 arcs. When scaled up to three and four day duties the networks have 566 nodes and 5,674 arcs and 298 node and 4,810 arcs, respectively. When reduced, the networks have 2,292, 4,240 and 3,517 arcs, reducing the number of decision variables 16%, 25% and 27% respectively.

The costing of the arcs in a duty period network would simply be the cost as calculated by the duty cost calculation in the duty generation phase. The goal of the sub-problem is to use the aforementioned optimal dual variables of the reduced master problem to find the most negative reduced cost pairings. The reduced cost of a pairing is;

$$\bar{c}_p = c_p - \sum_{i \in p} y_i \tag{3.3}$$

To calculate the cost values of the arcs in the sub-problem, some literature suggests using the following reduced cost;

$$\bar{c}_d = -\sum_{i \in d} y_i \tag{3.4}$$

This approach simply takes the negative sum of all dual variables corresponding to all the legs in the given duty and assigning that as the cost on the arc. This cost function may give a network which does not penalize for duties which have flying times falling below the minimum guarantee and have flying spread out such that duty rigs are cost factors. In order to adjust for this, the above cost function was compared to the following cost function;

$$\bar{c}_d = c_d - \sum_{i \in d} y_i \tag{3.5}$$

This cost function uses the previously calculated duty cost and adjusts it using the optimal dual variables from the restricted master problem. In this way duties with many legs are favored, but pairings with minimum guarantee penalties or duty rig penalties keep their penalized costs.

The cost $\bar{c}_d$ needs to be interpreted as $\bar{c}_{ij}$ where i is the starting leg of duty d and j is the ending leg of duty d. This will lead to multiple identical i, j combinations, in which case, the $\bar{c}_{ij}$ is taken to be the lowest cost of the set available and the corresponding duty must be noted in case it is part of the optimal solution. Adding the costs of the overnight rest arcs equal to their length we are able to solve the full formulation of the sub-problem:

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in A} \bar{c}_{ij} x_{ij} \\
\text{st} \quad & \sum_{j:(s,j)\in A} x_{sj} = 1 \\
& \sum_{i:(i,t)\in A} x_{it} = 1 \\
& \sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = 0 \quad \forall i \\
& x_{ij} \in \{0,1\}
\end{aligned}
\tag{3.6}
$$

where x_{ij} is the binary decision variable indicating the inclusion of a duty represented by the starting and ending legs i, j in the optimal solution, A is the arc set of the network and s and t are the source and sink nodes respectively.

After solving the sub-problem, the optimal pairings are entered into the restricted master problem and new optimal dual variables are found. This process is stopped once no more negative reduced cost pairings are found or the solution does not improve after several iterations.

Chapter 4

Implementation and Results

4.1 Implementation Details

All models were programmed in MATLAB. The LP's and IP's are solved using CPLEX functions called directly from MATLAB. Because of the nature of set partitioning and covering problems, the matrix structure used can be described as having a row for each flight to be covered and a column for each duty or pairing. This matrix is then populated with a parameter limited to zeros and ones, a_{ij}, with $a_{ij} = 1$ indicating that leg i is covered by pairing j, and takes the value of zero if not. At times, to limit the size of matrices generated, indices are used instead. For example the composition of a two day pairing is two rows, each contains the index of a duty. These indices can then be cross referenced with a matrix as described before, and the addition of the two columns would create one column indicating all legs covered by the pairing. All time reported in the following sections are clock times of running MATLAB scripts.

The test case is a schedule taken from real industry data containing 440 legs spread out over 4 days. There are 14 stations in the network, one of which is taken as the crew base. All flight legs are between one and three hours long. Aircraft flows are not considered in these models. Duty periods must allow for a minimum connection time of 30 minutes between legs up to a maximum of three hours. The minimum duty break given to crew is nine hours up to a maximum of 24 hours. The pre-allocation runs were performed with several different pre-allocation sizes because it is not possible to know *a priori* how many duties will be generated.

4.2 Duty Periods

The computational performance of the algorithms described in section 3.1 is studied in table 4.2. The computation time for each of the algorithms is reported as well as the effect of pre-allocation (PA) and in the case of the BFS search, pre-processing (PP).

PA Matrix Size	Clock Time (s)			PA Savings (%)			Savings (%)		
	DFS	BFS	BFS-PP	DFS	BFS	BFS-PP	Avg. by PA	by BFS	by PP
1	142.5	64.43	63.93					54.8	0.78
$4,400$	107.9	48.75	48.64	24.3	24.3	23.9	24.2	54.8	0.23
$7,500$	39.98	17.41	17.34	72.0	73.0	72.9	72.6	56.5	0.40
$8,000$	29.05	11.80	11.80	79.6	81.7	81.5	80.9	59.4	0.00
$9,000$	0.944	0.584	0.579	99.3	99.1	99.1	99.2	38.1	0.86
$10,000$	0.897	0.601	0.565	99.4	99.1	99.1	99.2	33.0	5.99
$44,000$	0.998	0.663	0.652	99.3	99.0	99.0	99.1	33.6	1.66
$440,000$	1.89	1.5	1.38	98.7	97.7	97.8	98.1	20.6	8.00

Table 4.1: Numerical Results of Duty Generation Tests

The results in table 4.2 are plotted in figures 4.1 and 4.2. Figure 4.1 illustrates that the performance of the DFS search algorithm is easily outperformed by both of the BFS based search algorithms, regardless of pre-allocation policies. The effect of the pre-processing is far less than the effect of switching to the BFS approach. Pre-processing appears to only improve the performance when the pre-allocation overestimates the final number of duties. This can be seen by the lines for the BFS and the BFS-PP data in panel (a) of figure 4.1 running very close together, and in panel (b) they diverge as the pre-allocation increases.

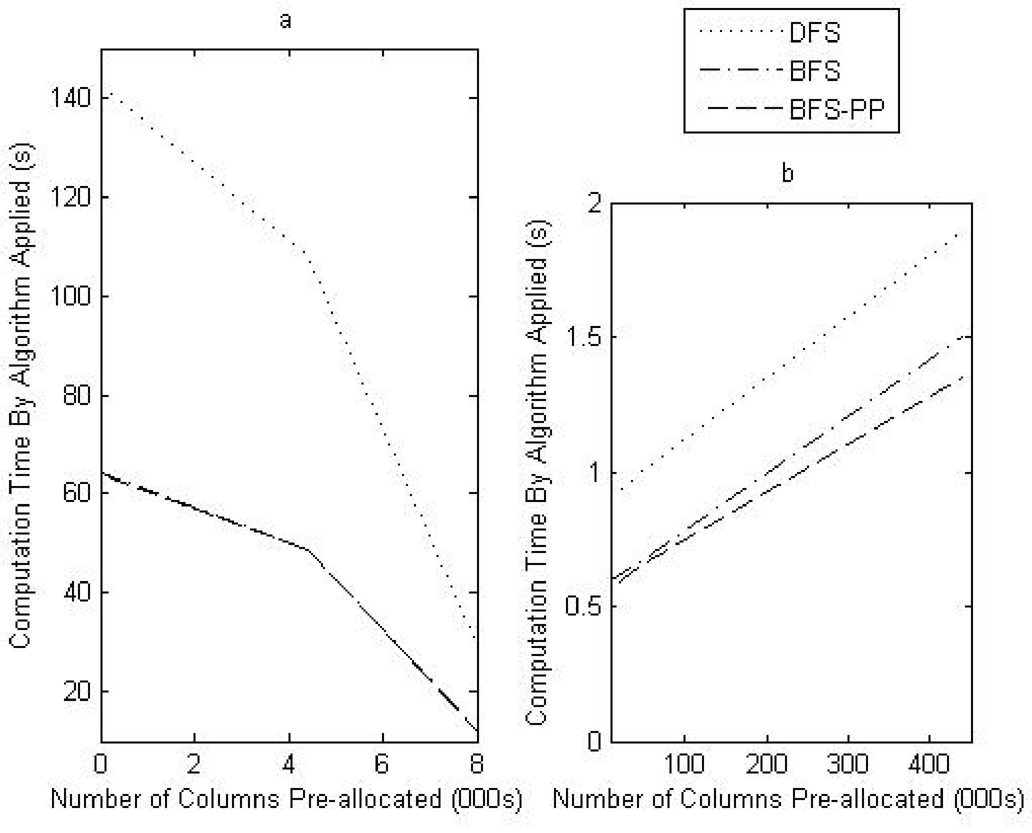

Figure 4.1: Comparison of computation time of DFS, BFS and BFS-PP with pre-processed matrix sizes below (a) and above (b) final matrix size.

Figure 4.2 shows that the loss in performance of the algorithms when overestimating the final number of duties is small relative to the gains of accuracy when it is underestimated. This can be seen by the highly positive slope of the line in panel (a) of figure 4.2 compared to a much smaller negative slope in panel (b).

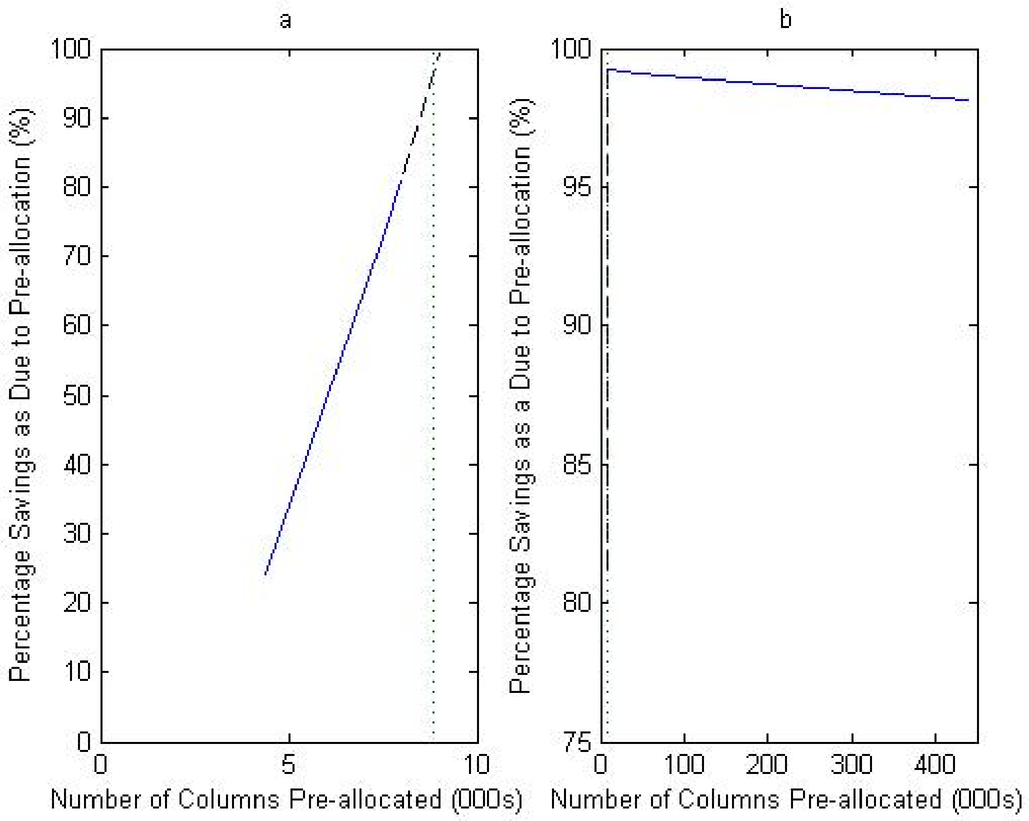

Figure 4.2: Plot of pre-allocation results. a)Effects when allocation falls below final number of duties b)when allocation falls above the number of duties. With the final number of duties indicated by the vertical dotted line.

4.3 Pairings

4.3.1 Enumeration

Applying the enumeration to the sample case, using the duty periods generated in section 3.1, 412 one day pairings, 34, 292 two day pairings and 3, 382, 519 three day pairings were successfully generated. Four day pairings were attempted, however in excess of 2×10^9 combinations can be made after applying preliminary filters. This many combinations is too much to handle with MATLAB, as MATLAB frequently runs out of memory when attempting to create such large matrices. Even the final number of four day pairings is too large. It falls in the 100 million range. The matrices that were created contain one row per day of pairing length. Tables 4.2 and 4.3 show the number of combinations which can be made in the attempt to generate 4 day pairings using two different approaches. Each of these combinations requires one column in a matrix in MATLAB. As illustrated, the number of combinations is very large and impossible to work with using MATLAB.

Start Duties	Mid Duties (Day 2)	Mid Duties (Day 3)	End Duties
251	1,469	1,469	382
368,719		561,158	
Filter		Filter	
37,841		65,447	
2.477×10^9 (Out of Memory)			

Table 4.2: Number of columns in matrices when generating four day pairings

Start Duties	Mid Duties (Day 2)	Mid Duties (Day 3)	End Duties
251	1,469	1,469	382
368,719			
Filter			
37,841			
55,588,429			
Filter			
5,997,831			
2.291×10^9 (Out of Memory)			

Table 4.3: Lengths of matrices when generating four day pairings combining matrices in a different order

The number of four day pairings which would be generated is in excess of 100 million pairings. Even this final size matrix is too large to manipulate and use.

The pairings which were enumerated (up to three days) were then used in a set covering problem to establish an upper bound on the optimal solution to this problem. Because the set of three day pairings, when put in a set covering matrix form, was too large for MATLAB, a simple heuristic was deployed to attempt a set covering solution. There are 165 legs which are not covered by one day pairings which are covered by two day pairings. Similarly, there are 18 legs which are covered by three day pairings but not covered by two day pairings. A further 8 legs require four day pairings to be covered. This phenomenon was used to select a series of three day pairings which are then known to be in the optimal solution (of the problem with pairings up to three days long). The duties which cover the legs which are not covered by shorter pairings are selected and added to the set covering problem.

The heuristic enumeration bound, using the above heuristic method, is found between 14 and 160 seconds, and has a solution value of $50,769$ minutes. The computation time was very erratic and unpredictable. This solution leaves 8 legs uncovered and has not been adjusted with penalty costs for uncovered legs.

4.3.2 Column Generation

Experiments were performed on the same sample schedule as used in previous sections, using the duty periods generated before. Because short pairings are exhaustively generated quite quickly the impact of generating various different length pairings while using the generated sets of shorter sets to start the restricted master problem was studied. These results are found in table 4.4. In these experiments the sub-problem used a different network for each of the different lengths pairings which were generated. In the second column of table 4.4 the length of the generated pairings is indicated, any pairing lengths which are not mentioned are fully enumerated and added to the restricted master problem at the beginning. An integer program was run only once at the end to determine the upper bound on the integer solution.

It can be seen in table 4.4 that this approach slows down significantly when the enumerated two day pairing set is added to the restricted master problem. And given the observation that shorter pairings do not cover the same legs as longer pairings mentioned on page 48, the same heuristic was applied to reduce the number of columns added to

SP Cost Function (equation)	Pairings Gen'd (days)	Z_{LP} (mins)	Z_{IP} (mins)	Average Iter'n Time (s)	Iter'ns	Total Clock Time (s)	No. Starting Columns	No. Columns Gen'd
3.4	$1, 2, 3, 4$	$63, 339$	$65, 840$	0.2100	143	34.40	440	572
3.4	$2, 3, 4$	$59, 796$	$61, 522$	0.2026	126	28.73	852	378
3.4	$3, 4$	$57, 474$	$58, 635$	4.7724	48	237.79	$35, 144$	98
3.5	$1, 2, 3, 4$	$61, 176$	$62, 663$	0.2332	224	74.14	440	896
3.5	$2, 3, 4$	$55, 966$	$56, 840$	0.2220	198	47.39	852	594
3.5	$3, 4$	$55, 967$	$56, 533$	4.4077	131	586.34	$35, 144$	262

Table 4.4: Numerical results of column generation runs comparing the effect of sub-problem cost functions and lengths of pairings generated by the sub-problem

the restricted master problem initially. Now a combination of fully enumerated length of pairings, a limited set of pairings and generated pairings are used to find a solution. The goal of this is to reduce the run time by reducing the number of columns in the reduced master problem but improve the solution quality by adding a set of columns of which we know one must be in the optimal solution. The experiments performed and their results are found in table 4.5.

When the data from figures 4.3 and 4.4 are plotted, a clear tradeoff curve is developed. In the figures, the numbers by the data points refer to the row number to which the data point corresponds in their respective tables. Techniques which require more computational time yield better solutions. Even though at low computational times the solution quality increases rapidly with minor increases in computational time, the magnitude of increase in solution quality begins to decrease as computation time increases. This phenomenon is generally consistent with difficult optimization problems.

Figures 4.3 and 4.4 show one outlier below the tradeoff curve. This outlier is part of the hybrid column generation approach, using the full sub-problem cost function, and generating two, three and four day long pairings. This test instance has the best tradeoff between time and solution quality, with computational time falling below 50 seconds, finding an integer solution of $56, 540$ minutes. Figure 4.3 reveals that the hybrid column generation

SP Cost Function (eq'n #)	Pairing Sets			Z_{LP} (mins)	Z_{IP} (mins)	Avg. Iter. Time (s)	Iter'ns.	Total Clock Time (s)	No. Start Columns	No. Gen'd Columns
	All	Cover Only	Gen.							
3.4	1	2	$3, 4$	$57, 444$	$58, 045$	2.18	62	139.33	$21, 432$	124
3.4	1	2	$2, 3, 4$	$57, 346$	$58, 468$	1.59	68	114.15	$21, 432$	204
3.4	1	$2, 3$	$3, 4$	$55, 764$	$56, 001$	20.47	32	679.88	$57, 782$	64
3.4	1	$2, 3$	$2, 3, 4$	$55, 795$	$55, 994$	19.89	27	561.47	$57, 782$	81
3.4	$1, 2$	3	$3, 4$	$56, 694$	$57, 398$	11.95	32	409.83	$48, 246$	64
3.5	1	2	$3, 4$	$55, 989$	$56, 735$	2.21	156	352.99	$21, 432$	312
3.5	1	2	$2, 3, 4$	$55, 934$	$56, 458$	2.16	128	282.33	$21, 432$	384
3.5	1	$2, 3$	$3, 4$	$54, 464$	$54, 862$	20.86	87	1842.4	$57, 782$	174
3.5	1	$2, 3$	$2, 3, 4$	$54, 490$	$54, 855$	21.84	68	1416.6	$57, 782$	204
3.5	$1, 2$	3	$3, 4$	$55, 306$	$55, 966$	12.16	93	1352.7	$48, 246$	186

Table 4.5: Numerical results of column generation with reduced enumeration pairing sets

approach without adding reduced enumeration sets (table 4.4) consistently performs faster, with the exception of only generating three and four day pairings, at the expense of solution quality. The reduced enumerated set contributes to a better solution quality but they still require considerable computation times. The best result from table 4.5 is achieved by using the full set of one day pairings, adding two day pairings which cover legs not coved by one day pairings, and then generating two, three and four day pairings. This is done in 282.3 seconds find a solution of 56,458 minutes. This saves 75 minutes over adding the whole two day pairing set, saving 300 seconds of computing time. Similarly, adding partial sets of two and three day pairings while generating two, three and four day pairings would save a further 1,985 minutes in the objective value function, at a cost of 1370 seconds of computation time. This value is also an improvement over the full one and two day pairing set, generating three and four day pairings by 1678 minutes costing 830 seconds of computation time. Figure 4.4 compares the two different methods for computing the cost for the sub-problem. The full sub-problem arc cost function yields the best results, but the cost function consisting of just the negative sum of the duals yielded the fastest solution time. While slower the best trials appear to be from using the full cost function

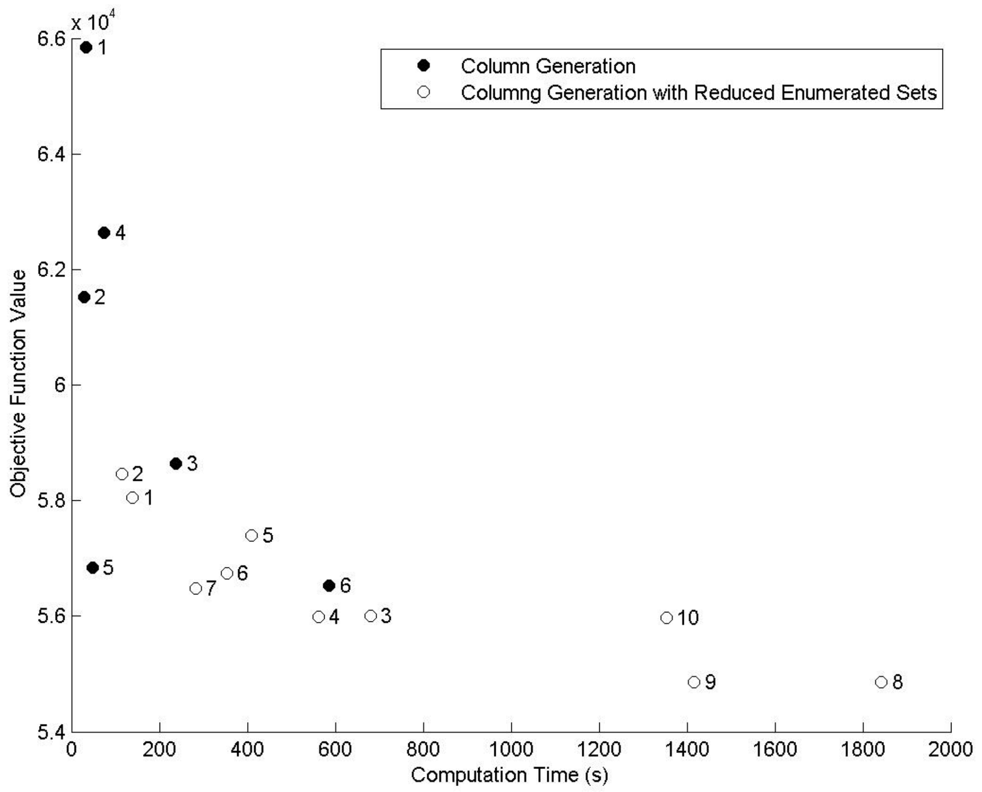

Figure 4.3: Comparison of computation time vs. solution quality tradeoff between the column generation approach and the hybrid generation-reduced enumeration sets technique.

from equation 3.5. The best solutions identified from tables 4.4 and 4.5 both use the full cost function.

The reduced sub-problem network programming reduced the computation time by an average of 0.21 seconds per sub-problem solved. The shortest path problem appears to be quite quickly solved using CPLEX and a reduction of problem size of about 25% yields very little time savings. This magnitude of time savings makes very little difference on a problem the size of this sample case. However, in larger cases and when applied to the sub-problem approach the repeated solving of networks these small savings could accumulate to a significant magnitude.

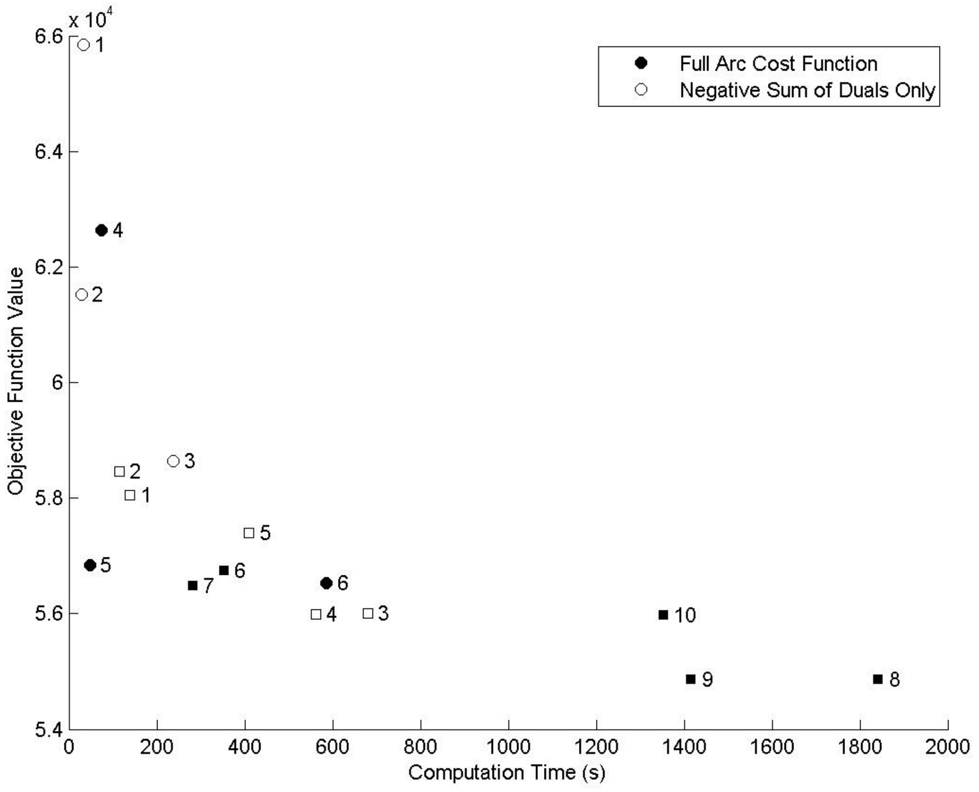

Figure 4.4: Comparison of computation time vs. solution quality tradeoff between subproblem arc cost functions. Circles indicate data from table 4.4 and squares data from table 4.5

Chapter 5

Conclusions and Future Research

The methods proposed in this thesis for the generation of duty periods were able to reduce the computation time of 8820 duties from 142.5 seconds to less than one second through the application of BFS search techniques, pre-allocation and pre-processing techniques. When generating duties, pre-allocation of data sets should exceed the number of expected duties. This allows for time savings due to prepossessing to be realized as well. In this case the number of duties generated are in the order of twenty times the number of legs in the schedule. Estimations of expected number of can be done aggressively because overestimation has far less impact on computational time than underestimation.

In the generation of pairings, the column generation approach is a good way to generate pairings of two days and longer. Computational results showed that generating one day pairings exhaustively and then generating longer pairings using the column generation approach was best. The best results were achieved when generating two, three and four day pairings, while fully enumerating one day pairings. Taking the generation of all pairings and the arc cost function suggested in literature, $\bar{c}_d = -\sum_{i \in d} y_i$, as an approximation of other column generation approaches, the above suggestions, combined with the application of $\bar{c}_d = c_d - \sum_{i \in d} y_i$ as the cost function for duties in the sub problem, saved $9,000$ minutes (13.7%) in the objective value function, while only taking 13 seconds longer. If the objective function of this method is to be improved further, adding the partially enumerated set of two day pairings to the restricted master problem, while still generating two, three and four day pairings will yield a 75 minute better objective value in just over 300 seconds less than by adding the whole set of two day pairings.

While adding partial sets of enumerated pairings to the restricted master problem did improve the solutions of the column generation approach significantly, the magnitude of the increase in computational time makes the tradeoff less palatable. Although, it is better than adding fully enumerated sets of pairings. The most reasonable improvement was 9,382 minutes (14.3%) down to 56,458, in 282 seconds.

The computational improvements in the duty generation experiments are improvements due to programming techniques. This shows that it may be beneficial to have programming and computing expertise when designing future implementations. Future studies should take this into account and perhaps utilize the expertise of computer scientists and programmers.

Column generation is shown to be a good solution for the generation of pairings, but more research is required. More complicated rules must be incorporated in order to make the model created in this thesis commercially usable. Sub-problem networks can designed to do so through multi-commodity flows and labeling techniques. An alternative way which may be a good method for starting the restricted master problem is to use a manually modified previous schedule from a similar period or to have a scheduling expert craft a starting solution. The current method does not necessarily have an integer feasible starting point. The starting point has a clear impact on the final solution of the column generation, this phenomenon could be studied further.

The model presented uses a shortest path to generate one pairing of given length per iteration, in future work this should be addressed. A sub-problem generating several pairings at once should be attempted. Given one set of optimal dual values from the restricted master problem, several negative reduced cost pairings can be made. Improvement of the performance of the column generation approach may be achieved using a method which takes greater advantage of the dual information passed into the sub-problem.

References

[1] R. Anbil, C. Barnhart, L. Hatay, E. Johnson, and V. Ramakrishnan. Crew-pairing optimization at american airlines decision technologies. In T.A. Ciriani and R.C. Leachman, editors, *Optimization in Industry*. John Wiley and Sons Ltd., 1993.

[2] R. Anbil and J. Forrest. Column generation and the airline crew pairing problem. *Documenta Mathematica*, 35(1):45 – 58, 1998.

[3] R. Anbil, E. Gelman, B. Patty, and R. Tanga. Recent advances in crew-pairing optimization at american airlines. *Interfaces*, 21(1):62 – 74, 1991.

[4] R. Anbil, R. Tanga, and E. Johnson. A global approach to crew-pairing optimization. *IBM Syst. J.*, 31(1):71–78, 1992.

[5] E. Andersson, E. Housos, N. Kohl, and D. Wedelin. Crew pairing optimization. In Gang Yu, editor, *Operations Research in the Airline Industry*, volume 9 of *International Series in Operations Research & Management Science*, pages 228–258. Springer US, 1998.

[6] J. Arabeyre, J. Fearnley, F. Steiger, and W. Teather. The airline crew scheduling problem: A survey. *Transportation Science*, 3(2):140 – 163, May 1969.

[7] E. Baker, L. Bodin, W. Finnegan, and R. Ponder. Efficient heuristic solutions to an airline crew scheduling problem. *A I I E Transactions*, 11(2):79–85, 1979.

[8] M. Ball and A. Roberts. A graph partitioning approach to airline crew scheduling. *Transportation Science*, 19(2):107, 1985.

[9] C. Barnhart, A. Cohn, E. Johnson, D. Klabjan, G. Nemhauser, and P. Vance. Airline crew scheduling. In Randolph W. Hall, editor, *Handbook of Transportation Science*, volume 56 of *"International Series in Operations Research & Management Science"*, pages 517–560. Springer US, 2003.

[10] C. Barnhart, E. Johnson, G. Nemhauser, M. Savelsbergh, and P. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46(3):pp. 316–329, 1998.

[11] Cynthia Barnhart, Ellis L. Johnson, Ranga Anbil, and Levent Hatay. A column-generation technique for the long-haul crew-assignment problem. In Tito A. Ciriani and Robert C. Leachman, editors, *Optimization in Industry II*, pages 7–24. John Wiley & Sons, Inc., 1994.

[12] J. Beasley and B. Cao. A tree search algorithm for the crew scheduling problem. *European Journal of Operational Research*, 94(3):517 – 526, 1996.

[13] E. Butchers, P. Day, A. Goldie, S. Miller, J. Meyer, D. Ryan, A. Scott, and C. Wallace. Optimized crew scheduling at air new zealand. *Interfaces*, 31(1):pp. 30–56, 2001.

[14] H. Chu, E. Gelman, and E. Johnson. Solving large scale crew scheduling problems. *European Journal of Operational Research*, 97(2):260 – 268, 1997.

[15] T. Crainic and J. Rousseau. The column generation principle and the airline crew scheduling problem. *INFOR*, 25(2):136–151, 1987.

[16] G. Desaulniers, J. Desrosiers, Y. Dumas, S. Marc, B. Rioux, M. Solomon, and F. Soumis. Crew pairing at air france. *European Journal of Operational Research*, 97(2):245 – 259, 1997.

[17] G. Desaulniers, J. Desrosiers, A. Lasry, and M. Solomon. Crew pairing for a regional carrier. In *Proc. 7th Internat. Workshop on Computer-Aided Scheduling of Public Transport*. Springer, 1997.

[18] J. Desrosiers, Y. Dumas, M. Desrochers, M. Soumis, B. Sanso, and P. Tredeau. A breakthrough in airline crew scheduling. Technical report, Cahiers du GERAD G-91-11, 1997.

[19] A. Ernst, H. Jiang, M. Krishnamoorthy, B. Owens, and D. Sier. An annotated bibliography of personnel scheduling and rostering. *Annals of Operations Research*, 127(1):21–144, 2004.

[20] A. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier. Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, 153(1):3 – 27, 2003.

[21] M. Etschmaier and D. Mathaisel. Airline scheduling: An overview. *Transportation Science*, 19(2):127, 1985.

[22] M. Gamache, F. Soumis, G. Marquis, and J. Desrosiers. A column generation approach for large-scale aircrew rostering problems. *Operations Research*, 47(2):pp. 247–263, 1999.

[23] I. Gershkoff. Optimizing flight crew schedules. *Interfaces*, 19(4):pp. 29–43, 1989.

[24] B. Gopalakrishnan and E. Johnson. Airline crew scheduling: State-of-the-art. *Annals of Operations Research*, 140(1):305–337, 2005.

[25] G. Graves, R. McBride, I. Gershkoff, D. Anderson, and D. Mahidhara. Flight crew scheduling. *Management Science*, 39(6):736 – 745, 1993.

[26] E. Housos and T. Elmroth. Automatic optimization of subproblems in scheduling airline crews. *Interfaces*, 27(5):pp. 68–77, 1997.

[27] D. Klabjan, E. Johnson, G. Nemhauser, E. Gelman, and S. Ramaswamy. Solving large airline crew scheduling problems: Random pairing generation and strong branching. *Computational Optimization and Applications*, 20(1):73–91, 2001.

[28] S. Lavoie, M. Minoux, and E. Odier. A new approach for crew pairing problems by column generation with an application to air transportation. *European Journal of Operational Research*, 35(1):45 – 58, 1988.

[29] M. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):pp. 1007–1023, 2005.

[30] M. Minoux. Column generation techniques in combinatorial optimization: A new application to crew pairing. In *Proceedings XXIVth AGIFORS Symposium*, pages 15 – 30, 1984.

[31] R. Rushmeier, K. Hoffman, and M. Padberg. Recent advances in exact optimization of airline scheduling problems. Technical report, Department of Operations Research and Operations Engineering, George Mason University, 1995.

[32] P. Vance, A. Atamturk, C. Barnhart, E. Gelman, E. Johnson, A. Krishna, D. Mahidhara, G. Nemhauser, and R. Rebello. A heuristic branch-and-price approach for the airline crew pairing problem. Technical report, LEC-97-06, Georgia Institute of Technology, Atlanta, GA, 1997.

[33] P. Vance, C. Barnhart, E. Johnson, and G. Nemhauser. Airline crew scheduling: A new formulation and decomposition algorithm. *Operations Research*, 1997.