Contents

List of Figures

List of Tables

Introduction

The area of research addressed in this thesis is *airline scheduling.* Airline scheduling traditionally consists of solving the following five planning problems one after the other. First, marketing decisions in the *schedule design problem* determine the schedule of flights the airline operates. Given the set of flights in a schedule, the solution of the *fleet assignment model* determines which flight is operated by which aircraft type. Next, the *aircraft routing problem* seeks a minimal cost assignment of available aircraft to the flights. Similarly, the *crew pairing problem* (or *tour of duty problem*) allocates generic crew to flights in a minimal cost way. A set of generic crew pairings is constructed subject to many rules to ensure that each flight is covered by the correct number of crew members. The last of the planning problems is the *crew rostering problem.* Based on the constructed crew pairings, a *line of work* is assigned to each individual crew member.

Traditionally, all five scheduling problems are solved in sequence although the problems are interdependent. Clearly, all subsequently solved problems must assign aircraft types, aircraft, and crew to the flights that are determined in the schedule design problem. Once the fleet assignment problem is solved, one aircraft routing problem is solved for each fleet type. The crew pairing problem depends on the aircraft routing problem, since the connection time between two flights a crew is allowed to operate can differ depending on whether the crew stays on the same aircraft or not. Finally, individual crew members are assigned to generic crew pairings in the crew rostering problem. Since airlines operate in a highly competitive market, the main goal of most of these problems is the minimisation of a cost objective. The decline in airline passengers following the events of September 2001, rising fuel prices, and competitive pressure from low-cost airlines increase the need for traditional airlines to operate as efficiently as possible. Solving the five problems sequentially can lead to a suboptimal solution due to the interdependencies between the problems. Once one problem is solved, the solution of this problem may restrict the feasible solutions of all subsequently solved problems resulting in an overall suboptimal solution. Additionally, a large percentage of the variable costs of airline operations occurs in the crew pairing problem which is solved late in the sequence.

The minimisation of planned costs alone to create a highly efficient schedule neglects the characteristics of the environment in which such a schedule is operated. A highly efficient schedule usually features very short ground times between flights for aircraft and crew to keep aircraft utilisation high and crew costs low. During airline operations, however, disruptions are likely to occur because of delayed passengers, aircraft malfunction, or weather conditions, to name just a few. Once disruptions occur and ground times between flights of one aircraft are minimal, the flights operated subsequently by the same aircraft will also depart late. If, additionally, crew are changing aircraft on a connection with short ground time after a delayed flight, the flight operated subsequently by the crew will most likely also depart late. Such a propagation of delay can quickly cause serious disruptions of wide parts of the schedule. We refer to a schedule where the effects of an initial disruption on other flights in the schedule are minimal as *operationally robust*. A schedule that is not robust can cause large additional costs for an airline, for example requiring reserve crews and passenger re-accommodation, and resulting in damage to reputation.

A famous example of the effects of operating a non-robust schedule was provided - not intentionally - by the UK based low-cost airline Easyjet in August 2002. Many flights were delayed or cancelled and thousands of passengers had to be re-accommodated all over Europe. The airline was operating highly cost efficient crew pairings together with very short connection times:

> Many budget airlines have fast turnarounds, with airlines unloading their passengers and quickly re-boarding.
>
> This allows them to make the maximum use of their small fleets.
>
> While the strategy is cost effective, it also increases the likelihood

> of cancellations when technical or staffing problems arise.[1]

Delays on a Saturday morning propagated throughout the day and became so severe that crew reached their maximum working hours and it became illegal for them to continue operating:

> Crews caught up in the delays worked up to their maximum hours and then had to be allowed home to rest.[2]

The reasons for the accumulation of delays were described by a spokesperson of Easyjet:

> We thought the new rostering system would be more efficient and better. It proved to be anything but that. [...]
>
> The system was "splitting up" crews, meaning that a re-fuelled plane and a pilot could be waiting at Luton airport but the cabin crew would be stuck in Barcelona. It [Easyjet] plans to return to its old rostering system next month.[3]

It took the airline several days to recover from this operational disaster until all flights were departing as scheduled. The airline subsequently removed some of the flights from their schedule to allow for more buffer time during operations.

Ehrgott and Ryan [2002] and Yen and Birge [2006] have shown that the robustness of crew pairing solutions can be significantly improved if aircraft changes are only made when ground time between the incoming and outgoing flights is much greater than the minimum ground time. This can be achieved in the crew pairing problem by penalising aircraft changes when ground time is short. Robust crew pairing solutions then have "crew following the same aircraft" as much as possible and changing aircraft only when ground time between flights is much longer than the minimum. In this sense, the robust crew pairing solution depends on the given aircraft routing solution. Again, a sequential solution method may result in a suboptimal solution compared to a solution method that considers both problems simultaneously.

[1]http://news.bbc.co.uk/2/hi/business/2182650.stm (14/05/2008)
[2]http://news.bbc.co.uk/2/hi/uk_news/2172537.stm (14/05/2008)
[3]http://www.telegraph.co.uk/news/uknews/1404034/Easyjet-cancels-flights-as-rota-fails.html (14/05/2008)

Clearly, there exists a trade-off between minimal planned cost and operational robustness. Ideally, we would like to solve a bicriterion problem with the two objectives of cost and robustness considering all airline scheduling problem simultaneously in one integrated model replacing the traditional sequential approach. Such a formulation to integrate all airline scheduling problems is currently intractable. All individual problems are already hard to solve and integration increases the complexity of the formulation.

As a step towards integration of all airline scheduling problems, three of the problems are considered in this thesis: schedule design, aircraft routing, and crew pairing. We investigate whether it is possible to reduce the cost of the sequential approach solution and simultaneously increase its robustness by considering the three problems simultaneously rather than sequentially. We expect the largest gain in cost and robustness by considering these three problems and do not include the fleet assignment and crew rostering problems in our formulation. The fleet assignment model is important for large airlines with multiple aircraft types. In the context relevant for this thesis, the fleet can be regarded as homogeneous and fleet assignment can be omitted. The main objective of the crew rostering problem is maximising crew satisfaction rather than minimising cost. The crew rostering problem has therefore no influence on the cost of the overall solution and is also not considered.

In the first part of the thesis we only consider two of the problems: aircraft routing and crew pairing. We formulate the *robust and integrated aircraft routing and crew pairing problem* in one integrated model. This model yields one optimal solution for the two problems where the objective function is a weighted sum of cost and a robustness measure, penalising crew changing aircraft in the objective function. Because the problem is hard to solve, decomposition methods are proposed in the literature to solve the integrated problem (see for example Mercier et al. [2005]), but excessive computation times are necessary to solve the model to optimality. We propose two novel solution methods for the integrated model: an *iterative approach* and a *Dantzig-Wolfe decomposition approach.*

The iterative approach is an optimisation-based heuristic approach: instead of solving the integrated model, the two original problems are solved iteratively. Starting with a cost minimal crew pairing solution, in each iteration we solve the aircraft routing problem first, taking into account the current crew pairing solution, i.e. encouraging aircraft to follow the crew. Then, given the aircraft routing solution we re-solve the crew pairing problem and this time encourage the crew to follow the aircraft. We only use the objective functions in both problems to pass information from the problem solved previously to generate more and more robust solutions. Hence, the constraints of the models are unaltered and the complexity of the two problems is not increased. This procedure generates a series of feasible solutions for the integrated model with varying costs and robustness measures. The airline is not required to associate a monetary value with robustness a priori but can observe the trade-off between cost and robustness and then choose a solution they prefer to operate.

Various crew groups such as captains, first officers, and flight attendants are required to operate an aircraft. We therefore extend the iterative approach and consider multiple crew groups at the same time.

While the iterative approach generates feasible solutions very quickly, it cannot guarantee to find a solution of a certain quality specified beforehand. Nevertheless, a (possibly infeasible) lower bound on the crew pairing cost is provided by the algorithm so that the worst case solution quality can be observed. To obtain feasible lower bounds on the solution quality, we propose a Dantzig-Wolfe decomposition approach capable of solving the integrated model to optimality for a weighted sum objective function of crew pairing cost and robustness measure. Again, both problems are solved individually and the original structures of the problems are preserved. Aircraft routing problem and crew pairing problem each form one subproblem of the decomposition approach. The approach iterates between a master problem and both subproblems until an optimal solution to the integrated model is found.

In an extension, the schedule design problem is partially integrated into the formulation. We do not consider constructing a schedule from scratch because this problem is passenger demand driven and very complex. Also, a high degree of consistency is required between successive schedules. The departure times of some flights in the schedule are allowed to vary in some interval, which is why the problem is called the *time window problem*. We investigate whether such flexibility can further increase robustness and decrease crew pairing costs. The problem is difficult since we consider weekly scheduling periods and all flights with the same flight number and same origin and destination must depart at the same time on all weekdays. Therefore, constraints must be included in the model to *synchronise* departure times for such flights over multiple days. We propose a model for the *robust and integrated aircraft routing and crew pairing problem with time windows* and propose two solution methods for this problem. The first method uses an aircraft routing solution and a crew pairing solution as input. The sequences of flights included in routings and pairings of the solutions remain fixed. We find re-timings of the departure times for the fixed sequences such that robustness is maximised. In a second method we allow time windows within the iterative approach. A branch-and-bound algorithm that enforces branches on the time windows is used to synchronise the departure times.

In order to verify the performance of our solution approaches, we apply all solution methods to various domestic airline schedules of Air New Zealand. The iterative approach yields low cost solutions which are highly robust compared to the traditional sequential approach. We compare the quality of the solutions of the iterative approach with optimal solutions obtained by the Dantzig-Wolfe decomposition approach. We also compare the performance of the Dantzig-Wolfe decomposition approach with that of Benders decomposition which is currently known as the most successful approach in the literature. However, the approach has the disadvantage of adding constraints to the original formulations which can cause computational difficulties. By applying both time window solution methods, we demonstrate significant further savings in crew pairing cost and robustness. Often, the departure times of only very few flights are changed to achieve the improvements.

The main contributions of this thesis can be divided into two parts. Firstly, from a theoretical point of view, we want to answer the question whether it is possible to solve the integrated aircraft routing and crew pairing problem (with and without time windows) by a decomposition method that does not add constraints to the individual models of aircraft routing and crew pairing problems. This enables us to use existing efficient solution methods to solve each individual problem. All solution methods we propose preserve the original structures of aircraft routing and crew pairing problems. Secondly, the main focus of this thesis is to solve a practical problem. This poses additional challenges compared to solving a simplified mathematical model that only partially reflects reality. All rules and requirements of the Air New Zealand problem are considered in our solution approaches. We do not make any simplifications and the solutions we generate are ready to be operated in practice.

The thesis is organised as follows. In Chapter 1 we discuss mathematical models and solution methods that are commonly used in airline scheduling and for other operations research problems. The concepts are used throughout the thesis and being familiar with them is helpful to understand the motivation behind our methodology. We describe operations research problems in the airline industry and review the most relevant approaches in the literature in Chapter 2. The chapter provides detailed information on the problems we are interested in as well as on the planning process of airline scheduling in general. In Chapters 3 and 4 we describe in detail our solution approach for the aircraft routing and crew pairing problems, respectively. We list details on the particular problem instances and the specialised solution techniques that are tailored to address the specific problems. While the production crew pairing solver was provided by Air New Zealand to be used for the computational experiments, an aircraft routing algorithm and algorithms to solve all integrated problems were implemented from scratch. In Chapter 5 we describe the integrated and robust aircraft routing and crew pairing problem. We present a model and various solution approaches. These approaches are compared in an extensive computational experiment section. In Chapter 6 we enhance the model by also considering time windows. We again present computational experiments before we summarise our experiences and results in the Conclusion.

Chapter 1

Mathematical Background

In this thesis a number of different airline optimisation problems are addressed. Many of these problems can be formulated as mathematical optimisation models that have similar structures and many properties in common. Hence, the solution techniques to solve the various problems also have many similarities. In this chapter, we review the most popular models and solution techniques used for airline optimisation problems.

A linear optimisation model where all of the solution variables are required to be integer valued is called *integer program (IP)*:

$$\begin{array}{lrcl} \text{Minimise} & \boldsymbol{c}^T\boldsymbol{x} & & \\ \text{subject to} & A\boldsymbol{x} & = & \boldsymbol{b} \\ & \boldsymbol{x} & \in & \mathbb{Z}^n_+. \end{array} \tag{1.1}$$

The integer matrix A is of size $m \times n$, $\boldsymbol{c}$ and $\boldsymbol{b}$ are integer vectors of size n and m, respectively, and $\boldsymbol{x}$ is required to be integer. If only some (or none) of the variables $\boldsymbol{x}$ are required to be integer (1.1) is called *mixed integer program* (or *linear program (LP)*). In this thesis, variables $\boldsymbol{x}$ are also assumed to be non-negative unless stated otherwise. To solve (1.1), usually the *linear relaxation (or LP-relaxation)* of problem (1.1) is solved first, where the integrality conditions on the $\boldsymbol{x}$ variables are relaxed:

$$\begin{array}{lrcl} \text{Minimise} & \boldsymbol{c}^T\boldsymbol{x} & & \\ \text{subject to} & A\boldsymbol{x} & = & \boldsymbol{b} \\ & \boldsymbol{x} & \geq & 0. \end{array} \tag{1.2}$$

Column generation techniques (see Section 1.3) are often used to solve the LP-relaxation. Once problem (1.2) is solved, a branch-and-bound method (see Section 1.4) can be used to solve the original problem (1.1). Many airline optimisation problems can be formulated as large scale *set partitioning* or *multi-commodity flow problems* which are special cases of (1.1). In the subsequent sections we describe these two problems.

1.1 Set Partitioning Problem

The *set partitioning problem (SPP)* (see Wolsey [1998]) can be formulated as follows:

$$\begin{array}{lrcl} \text{Minimise} & \boldsymbol{c}^T\boldsymbol{x} & & \\ \text{subject to} & A\boldsymbol{x} & = & \mathbb{1} \\ & \boldsymbol{x} & \in & \{0,1\}^n. \end{array} \tag{1.3}$$

The set S we want to partition contains m elements and matrix A is a $m \times n$ binary matrix. Each column $\boldsymbol{a_j}, 1 \leq j \leq n$, of A represents a subset of S and contains a 1 in row i if element $i \in S$ is an element of this subset and a 0 otherwise. Value $c_j \in \mathbb{R}$ represents the cost of column $\boldsymbol{a_j}$. The solution of (1.3) is a cost minimal partition of the set S.

An example in airline scheduling where the set partitioning model can be used, is assigning crew members to operate flights of a schedule. All flights in the schedule form the set S and the columns of A represent subsets in the form of sequences of flights a single crew member can operate. All flights of the schedule must be partitioned so that each flight is operated by some crew member.

Sometimes additional constraints are used to model the consumption of limited resources. When these constraints with non-unit right-hand sides are added to problem (1.3) the resulting problem is called a *generalised set partitioning problem.* These constraints may for example limit the number of crew available at a particular crew base.

A special form of the set partitioning problem is the so-called *rostering problem.* For the rostering problem the matrix A can be written as $A = \binom{A'}{A''}$. A subset of rows (denoted by A') of matrix A forms the following block-diagonal structure:

$$A' := \begin{pmatrix} 1 & 1 & \cdots & 1 & 0 & 0 & \cdots & 0 & & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 1 & 1 & \cdots & 1 & & \vdots & & & \vdots \\ \vdots & & & \vdots & \vdots & & & \vdots & \cdots & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & & 1 & 1 & \cdots & 1 \end{pmatrix}.$$

For example, if a task requires to assign sequences of jobs to a number of employees then the constraints formed by A' ensure (together with the right hand side equal to 1) that exactly one sequence of jobs is assigned to each employee while the remaining constraints formed by A'' ensure that each job is assigned exactly once.

The constraints represented by A' are called *convexity* or *generalised upper bound (GUB)* constraints.

If we replace the $=$ sign in (1.3) with $\leq$ (or $\geq$), the corresponding formulation is called *set packing problem* (or *set covering problem*).

Zero-One Matrices with Integer Property

Three classes of zero-one matrices are known to ensure that all extreme points of the LP relaxation of (1.3) are integer and are referred to as matrices with the *integer property.* This means we can replace the binary conditions $\boldsymbol{x} \in \{0,1\}^n$ with lower and upper bounds $0 \leq \boldsymbol{x} \leq \mathbb{1}$ and are still guaranteed to find an integer optimal solution. This greatly simplifies the solution procedure of (1.3), see Sections 1.3 and 1.4 below. The three classes known are *totally unimodular* [Hoffmann and Kruskal, 1956], *balanced* [Berge, 1961], and *perfect* [Padberg, 1974]. A matrix is called totally unimodular, if the determinant of every square submatrix is -1, 0, or 1. A matrix is balanced if it does not contain any square odd submatrix with row and column sums equal to 2, i.e. the submatrix does not contain an odd order 2-cycle. We explain the structure of perfect matrices in Section 3.3.4 where we see that the class of perfect matrices is useful for rostering problems. The class of perfect matrices is the largest class and contains the class of balanced matrices which in turn contains the class of totally unimodular matrices.

1.2 Multi-Commodity Flow Problem

Another common formulation frequently used to solve airline optimisation problems is the *multi-commodity flow problem (MCF)* (see Ahuja et al. [1993]). A network $G(V, A)$ with nodes V and directed arcs A linking the nodes is given. Each arc $a(u, v) \in A$ has a capacity $c(a(u, v))$. Note that multiple arcs may connect nodes u and v. A total of k commodities $K_1, K_2, \ldots, K_k$ are defined by $K_i(s_i, t_i, d_i)$ where s_i and t_i are source and sink nodes of commodity i and d_i is the demand. The non-negative value of variable $f_i(a(u, v))$ represents the flow of commodity i along arc $a(u, v)$. The minimum cost multi-commodity flow problem can be stated as follows:

$$\begin{aligned}
\text{Minimise} \quad & \sum_{a(u,v)\in A} \left(\sum_{i=1}^{k} p_i(a(u,v)) f_i(a(u,v)) \right) && (1.4) \\
\text{subject to} \quad & \sum_{i=1}^{k} f_i(a(u,v)) \leq c(a(u,v)), \text{ for all } a(u,v) \in A, \\
\sum_{w\in V} f_i(a(w,v)) - & \sum_{w\in V} f_i(a(v,w)) = b_v^i, \text{ for all } v \in V, 1 \leq i \leq k,
\end{aligned}$$

where $p_i(a(u, v))$ is equal to the cost of sending one unit of flow of commodity i along arc $a(u, v)$ and $b_v^i = \begin{cases} -d_i, & \text{if } v = s_i \\ d_i, & \text{if } v = t_i \\ 0, & \text{otherwise.} \end{cases}$

The first set of constraints models capacities on the edges. The second set ensures flow conservation at each node: for each node that is not a source or a sink of commodity i, the amount of commodity i that enters the node must also leave the node. Also, the flow of commodity i that leaves the source node and enters the sink node must be equal to d_i, respectively. Constraints can be added to ensure that a certain amount of a commodity flows through a node. Model (1.4) assumes a single source and sink node for each commodity. If multiple source or sink nodes are required, artificial super source or sink nodes can be added. Super source and sink nodes are only linked to the original source and sink nodes, respectively. The capacities on the arcs linking super nodes and original nodes are set to the original supply and demand of the commodity, respectively.

The aircraft routing problem (see Chapter 3) can be formulated as a multi-commodity flow problem. A network representation of the flight schedule is used where flights are represented by nodes and two nodes are joined by an arc if an aircraft can operate these two flights in sequence. Commodities (i.e. air-craft) are shipped through this network such that the number of commodities arriving at an airport, is also departing from this airport (flow conservation). A capacity constraint on the source node can ensure that only the number of available aircraft is used. Additional constraints ensure that each flight is operated by exactly one aircraft.

1.3 Column Generation

The simplex algorithm is commonly used to find a cost minimal solution to a linear program (1.2). A minimal cost solution $\boldsymbol{x}^*$ of (1.2) is attained at one of the extreme points of the set $X = \{\boldsymbol{x} : A\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} \geq 0\}$. The simplex algorithm iterates from one extreme point to an adjacent extreme point until a cost minimal solution is found. Each extreme point is represented by a set of m linearly independent columns of A, the so called *basis*. An adjacent extreme point is reached by swapping exactly one basic column with one *non-basic* column of A. As the basis entering column (*pricing step* of the simplex algorithm), the column with minimal *reduced cost* over all non-basic columns is chosen if this cost is negative. Otherwise, optimality of the current solution (extreme point) is guaranteed and the algorithm stops. The reduced cost of a non-basic column $\boldsymbol{a}_s$ of matrix A is calculated as follows:

$$r_s = \left(c_s - \boldsymbol{c}_B^T A_B^{-1} \boldsymbol{a}_s\right).$$

Costs c_s and $\boldsymbol{c}_B$ are associated with columns $\boldsymbol{a}_s$ and the basis matrix A_B, respectively. The vector $\boldsymbol{\pi} = \boldsymbol{c}_B^T A_B^{-1}$ is the *dual* vector of solution $\boldsymbol{x}$. For a detailed description of the simplex algorithm see for example Chvátal [1983] or Schrijver [1986].

For practical problems in the airline industry the matrix A in problem (1.2) can contain a very large number of columns. In fact this number can be so large that it may take a very long time to even construct the matrix and the simplex algorithm may not be able to find the optimal solution within reasonable time.

Ford and Fulkerson [1958] and Dantzig and Wolfe [1960] introduced the idea of only implicitly considering all variables. The method is called *delayed column generation* (see Lübbecke and Desrosiers [2004] for more details) and works as follows. Instead of problem (1.2) the so called *restricted master problem* is solved with the simplex method:

$$\begin{array}{lrcl} \text{Minimise} & \boldsymbol{c}'^T\boldsymbol{x} & & \\ \text{subject to} & A'\boldsymbol{x} & = & \boldsymbol{b} \\ & \boldsymbol{x} & \geq & 0. \end{array} \tag{1.5}$$

Matrix A' initially only consist of a small (possibly empty) subset of all columns contained in A and $\boldsymbol{c}'$ contains the costs accordingly. To guarantee feasibility of (1.5), an artificial identity matrix is appended to A'. In the pricing step of the simplex algorithm a column with negative reduced cost which enters the basis must be found. Not only all non-basic columns of A' are checked for negative reduced cost but also all columns of the original matrix A not yet contained in A'. For the latter part the so-called *column generation subproblem* is solved. If $\boldsymbol{\pi}$ is the dual vector of the current basic solution the subproblem must identify a column $\boldsymbol{a}_s$ of A with negative reduced costs $r_s = (c_s - \boldsymbol{\pi}^T\boldsymbol{a}_s) < 0$ or guarantee that no such column exists.

The method is particularly beneficial whenever the column generation subproblem can be solved efficiently, e.g. as a combinatorial optimisation problem. In this case it may not be necessary to know all columns of matrix A explicitly to solve the original problem (1.2). The negative reduced cost columns and corresponding variables and costs are added to the restricted master problem and the simplex algorithm continues. Once no negative reduced cost column can be found, the current solution of (1.5) is an optimal solution for the original problem (1.2).

1.4 Branch-and-Price

Once the LP-relaxation of (1.1) is solved it is likely that some variables of $\boldsymbol{x}$ will have non-integer values. Most applications in airline scheduling require integer solutions. For example, it does not make sense to operate a single flight by two halves of an aircraft. A technique called *branch-and-bound* (see Barnhart et al. [1998b] for a survey) is then used to obtain integer solutions. The current fractional solution is stored as the *root node* of the *branch-and-bound tree.* In the *branching* step two nodes are added to the tree. At each node only a subset of all variables is considered. Depending on some properties of the fractional variables, we divide the variables into two (not necessarily disjoint) sets. At one node only the first subset of variables is considered while at the other node the second subset is considered. The variables are divided in such a way that the previous fractional solution is infeasible at either node. At both nodes the LP relaxation of (1.1) must be solved again where, additionally, all branching decisions for the node must be satisfied. We choose a node to be solved first. If the solution again contains fractional values the branching step is repeated on the current node, otherwise an integer solution of (1.1) is found. Note that the addition of two nodes at each branching step is the most common branching procedure and called *binary search.* Other branching strategies, adding more than two branches at each branching decision, can be used in a similar fashion.

At each node where an integer solution is found, the solution value is compared to the LP-relaxation solution value and if the gap between both values is small enough we terminate and return the integer solution. Otherwise, we store the solution and continue to explore the branch-and-bound tree. Once we find an integer solution we can stop branching on nodes that have a fractional solution with larger objective value (*bounding*) than the best integer solution found. The part of the tree below such a node cannot yield a better integer solution. When solving the LP-relaxation at each node, we can also use the column generation technique (*pricing*). The overall process of obtaining integer solutions is then called *branch-and-price.* If we do not generate new columns in the branch-and-bound tree we cannot guarantee to find an optimal or even feasible integer solution.

The branching decisions that are made are incorporated into the simplex algorithm by removing variables that violate the current branching decisions. Variables can be removed by setting their upper bound to 0. The decisions must also be obeyed in the column generation problem. Specially tailored branching rules exist for many airline scheduling problems. These are described in more detail in the model and solution sections.

Another technique to obtain integer solutions is to add constraints that cut off the current fractional solutions and is called *cutting plane* method. This method can be combined with the methods described above to *branch-and-cut* and *branch-and-price-and-cut*, respectively.

1.5 Linear Program Decomposition Principles

In this section common decomposition principles for large scale linear programs are described and compared, namely Dantzig-Wolfe decomposition, Benders decomposition, and Lagrangian relaxation. The common idea of decomposition principles is to decompose the original problem into smaller problems that can be solved more efficiently. These problems are then solved iteratively and information is passed from one to another until an optimal solution for the original problem is found. Similarly, in a relaxation method the difficult part of the problem is relaxed and its violation is penalised in the objective function.

1.5.1 Dantzig-Wolfe Decomposition

In this section we describe the *Dantzig-Wolfe decomposition* principle. An original LP is decomposed into an LP master problem and an LP subproblem and both are solved by LP techniques. This is a special case of the column generation principle (see Section 1.3) where the subproblem can have a more general form and can also be solved by combinatorial optimisation or enumeration algorithms for example. The goal is to solve the following linear problem:

$$\begin{array}{rrcl} \text{Minimise} & \boldsymbol{c}^T\boldsymbol{x} & & \\ \text{subject to} & A\boldsymbol{x} & \geq & \boldsymbol{b} \\ & \boldsymbol{x} & \geq & 0, \end{array} \tag{1.6}$$

with $A \in \mathbb{R}^{m \times n}, \boldsymbol{b} \in \mathbb{R}^m, \boldsymbol{c} \in \mathbb{R}^n$ and $\boldsymbol{x} \in \mathbb{R}^n$.

We can rewrite this problem by splitting A and $\boldsymbol{b}$ into $A = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix}$ and $\boldsymbol{b} = \begin{pmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \end{pmatrix}$:

$$\begin{array}{rrcl} \text{Minimise} & \boldsymbol{c}^T\boldsymbol{x} & & \\ \text{subject to} & A_1\boldsymbol{x} & \geq & \boldsymbol{b}_1 \\ & A_2\boldsymbol{x} & \geq & \boldsymbol{b}_2 \\ & \boldsymbol{x} & \geq & 0, \end{array} \tag{1.7}$$

where all $A_1, A_2, \boldsymbol{c}, \boldsymbol{b}_1, \boldsymbol{b}_2$ are real valued with appropriate dimensions.

If at least one of the sets of constraints is very large or hard to solve we can decompose problem (1.7) into two smaller and thus easier to solve problems. We reformulate problem (1.7) as the equivalent so-called *Dantzig-Wolfe master problem* (see Dantzig and Wolfe [1960]):

$$\begin{array}{rrcrcccc} \text{Minimise} & \boldsymbol{c}^T(V\boldsymbol{\lambda} & + & W\boldsymbol{\mu}) & & & & (dual) \\ \text{subject to} & A_2(V\boldsymbol{\lambda} & + & W\boldsymbol{\mu}) & \geq & \boldsymbol{b}_2 & \rightarrow & \boldsymbol{\pi} \\ & \mathbb{1}^T\boldsymbol{\lambda} & & & = & 1 & \rightarrow & \pi_c \\ & \boldsymbol{\lambda}, & & \boldsymbol{\mu} & \geq & 0. & & \end{array} \tag{1.8}$$

We define a polyhedron $P = \{\boldsymbol{x} \in \mathbb{R}^n_+ | A_1\boldsymbol{x} \geq \boldsymbol{b}_1\}$, $P =$ conv $(\{\boldsymbol{v}_1, \dots, \boldsymbol{v}_k\}) \cup$ cone $(\{\boldsymbol{w}_1, \dots, \boldsymbol{w}_l\})$ and sets $V = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_k\}$ and $W = \{\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_l\}$. Set V contains the extreme points of polyhedron P and set W contains the extreme rays of P. The set of constraints $A_1\boldsymbol{x} \geq \boldsymbol{b}_1$ is implicitly satisfied by the construction of V and W and $\boldsymbol{x} \in P$ is represented as a sum of a convex combination of extreme points and a conical combination of extreme rays of P.

As in Section 1.3, V and W are not needed to be known a priori but can be constructed during the solution process of the master problem (1.8). To solve the master problem, we start with initially empty matrices V and W and solve this restricted master problem. To guarantee feasibility, an artificial identity matrix is added to formulation (1.8). A phase I/II approach is used or large costs are associated with the artificial variables to ensure that no artificial variable has positive value in an optimal solution. We obtain a dual vector $\boldsymbol{\pi}$ associated with constraints $A_2(V\boldsymbol{\lambda} + W\boldsymbol{\mu}) \geq \boldsymbol{b}_2$ and a dual value π_c associated with the convexity constraint $\mathbb{1}^T\boldsymbol{\lambda}$. We solve a pricing *Dantzig-Wolfe subproblem* to check if a column with negative reduced costs exists:

$$\begin{array}{lrcl} \text{Minimise} & (\boldsymbol{c}^T - \boldsymbol{\pi}^T A_2)\boldsymbol{x} - \pi_c & & \\ \text{subject to} & A_1\boldsymbol{x} & \geq & \boldsymbol{b}_1 \\ & \boldsymbol{x} & \geq & 0. \end{array} \tag{1.9}$$

There are three possible outcomes for problem (1.9):

1. An optimal extreme point solution $\boldsymbol{v}$ exists with $(\boldsymbol{c}^T - \boldsymbol{\pi}^T A_2)\boldsymbol{v} - \pi_c < 0$. In this case we add the negative reduced cost vector $\boldsymbol{v}$ to matrix V of the master problem.

2. Problem (1.9) is unbounded. Here, we obtain an extreme ray $\boldsymbol{w}$ with $(\boldsymbol{c}^T - \boldsymbol{\pi}^T A_2)\boldsymbol{w} < 0$ and add $\boldsymbol{w}$ to matrix W of the master problem.

3. The optimal solution $\tilde{\boldsymbol{x}}$ has non-negative reduced cost $(\boldsymbol{c}^T - \boldsymbol{\pi}^T A_2)\tilde{\boldsymbol{x}} - \pi_c \geq 0$.

In cases (1) and (2) we add a negative reduced cost column to the master problem and re-solve the master problem and continue iterating between master and subproblem. In the last case optimality of the master problem is guaranteed, or the problem is infeasible if artificial variables with positive value are part of the solution. The optimal solution of the master problem is also an optimal solution of the original problem (1.7).

In each iteration, a bound on the solution quality can be calculated. Suppose $\zeta = \min(\boldsymbol{c}^T - \boldsymbol{\pi}^T A_2)\boldsymbol{x}$ is the optimal solution value of the current subproblem (1.9) without constant π_c and $\boldsymbol{\pi}$ the associated dual of the optimal solution of the current restricted Dantzig-Wolfe master problem. We can show that vector $\binom{\boldsymbol{\pi}}{\zeta}$ is a feasible solution to the dual problem of (the unrestricted) master problem (1.8) (see Wolsey [1998]). The value $(\boldsymbol{b}_2^T\boldsymbol{\pi} + \zeta)$ is a lower bound for the optimal solution value of (1.8) and hence (1.7). Combined with the upper bound available from the optimal solution of the restricted Dantzig-Wolfe master problem, we obtain an *optimality gap* for the solution and can stop the algorithm once this gap is sufficiently small.

1.5.2 Benders Decomposition

Another frequently used decomposition technique is Benders decomposition (Benders [1962], Minoux [1986]). Benders decomposition also iterates between a master problem and a subproblem but here constraints are generated by the subproblem and added to the master problem instead of variables as in the Dantzig-Wolfe decomposition.

We now consider the following LP:

$$\begin{array}{lccccl} \text{Maximise} & \boldsymbol{c}_1^T\boldsymbol{x}_1 & + & \boldsymbol{c}_2^T\boldsymbol{x}_2 & & \\ \text{subject to} & A_1\boldsymbol{x}_1 & + & A_2\boldsymbol{x}_2 & \leq & \boldsymbol{b} \\ & \boldsymbol{x}_1, & & \boldsymbol{x}_2 & \geq & 0, \end{array} \tag{1.10}$$

where $A_1, A_2, \boldsymbol{b}, \boldsymbol{c}_1, \boldsymbol{c}_2$ all take real values and $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are non-negative real variables.

We want to eliminate variables $\boldsymbol{x}_2$. This can be achieved via projection. To use projection we need to reformulate (1.10):

$$\begin{array}{lcccccccl} \text{Maximise} & z & & & & & & & \\ \text{subject to} & z & - & \boldsymbol{c}_1^T\boldsymbol{x}_1 & - & \boldsymbol{c}_2^T\boldsymbol{x}_2 & \leq & 0 \\ & & & A_1\boldsymbol{x}_1 & + & A_2\boldsymbol{x}_2 & \leq & \boldsymbol{b} \\ & & & \boldsymbol{x}_1, & & \boldsymbol{x}_2 & \geq & 0. \end{array} \tag{1.11}$$

Using Fourier-Motzkin elimination (see e.g. Schrijver [1986]) this is equivalent to:

$$\begin{array}{llcccccl} \text{Maximise} & z & & & & & & \\ \text{subject to} & uz & - & u\boldsymbol{c}_1^T\boldsymbol{x}_1 & + & \boldsymbol{v}^T A_1 \boldsymbol{x}_1 & \leq & \boldsymbol{v}^T\boldsymbol{b} \\ & & & & & \begin{pmatrix} u \\ \boldsymbol{v} \end{pmatrix} & \in & C, \end{array} \tag{1.12}$$

with $C = \{\binom{u}{\boldsymbol{v}} \in \mathbb{R}^{m+1} : \boldsymbol{v}^T A_2 - u\boldsymbol{c}_2^T = 0, u \geq 0, v \geq 0\}$. We can represent the polyhedral cone C as a conical combination: $C = \text{cone}\{\binom{u_1}{\boldsymbol{v}_1}, \ldots, \binom{u_s}{\boldsymbol{v}_s}\}$ and we can rescale these extreme rays such that u_i equals either 0 or 1. We can write $C = \text{cone}(\{\binom{0}{\boldsymbol{v}_k} : k \in K\}) + \text{cone}(\{\binom{1}{\boldsymbol{v}_j} : j \in J\})$ with $K \cup J = \{1, \ldots, s\}, K \cap J = \emptyset$. With this representation of C we can rewrite 1.12 as the so-called *Benders master problem*:

$$\begin{array}{llccccl} \text{Maximise} & z & & & & & \\ \text{subject to} & z & \leq & \boldsymbol{c}_1^T\boldsymbol{x}_1 & - & \boldsymbol{v}_j^T(A_1\boldsymbol{x}_1 - \boldsymbol{b}) & j \in J \\ & 0 & \leq & & - & \boldsymbol{v}_k^T(A_1\boldsymbol{x}_1 - \boldsymbol{b}) & k \in K. \end{array} \tag{1.13}$$

Similarly to the Dantzig-Wolfe decomposition approach, not all constraints are considered from the start. We start solving the restricted master problem with a small set (possibly empty in which case the optimal value of (1.13) z^* equals ∞) C and populate C during the algorithm by constructing constraints with a subproblem. Each time the restricted master problem is solved, we check if any constraint of the original problem is violated. Suppose the optimal solution of the current restricted master problem is $z^*, \boldsymbol{x}_1^*$. The check for violated constraints can be achieved by solving the following *Benders subproblem*:

$$\begin{array}{lcccl} \text{Minimise} & \boldsymbol{v}^T(\boldsymbol{b} - A_1\boldsymbol{x}_1^*) & + & u(-z^* + \boldsymbol{c}_1^T\boldsymbol{x}_1^*) & \\ \text{subject to} & & & \binom{u}{\boldsymbol{v}} & \in \; C. \end{array} \tag{1.14}$$

The subproblem is feasible since $\binom{0}{0} \in C$ is a solution. If $\binom{0}{0}$ is the optimal solution then the master and the original problem is solved to optimality. Otherwise we identified an extreme ray $\binom{u^*}{\boldsymbol{v}^*}$ with $\boldsymbol{v}^{*T}(\boldsymbol{b} - A_1\boldsymbol{x}_1^*) + u(-z^* + \boldsymbol{c}_1^T\boldsymbol{x}_1^*) < 0$. After rescaling, the ray will yield a constraint that is violated by the current master problem solution. We add this constraint to the master problem and re-solve.

1.5.3 Lagrangian Relaxation

Lagrangian decomposition (Geoffrion [1974], Fisher [1981], Fisher [1985], Martin [1999]) is also successfully applied to a number of airline scheduling problems.

Again, we consider problem (1.7) and reduce the problem to include only one part of the constraints. In contrast to Dantzig-Wolfe decomposition we now add the second set of constraints to the objective function together with a penalty for violation of the constraints.

This results in the *Lagrangian relaxation* of (1.7) for any given $\boldsymbol{\lambda} \geq 0$:

$$\begin{array}{rlcccl} L(\boldsymbol{\lambda}) = & \text{Minimise} & \boldsymbol{c}^T\boldsymbol{x} & - & \boldsymbol{\lambda}^T(A_2\boldsymbol{x} - \boldsymbol{b}_2) & \\ & \text{subject to} & A_1\boldsymbol{x} & & \geq & \boldsymbol{b}_1 \\ & & \boldsymbol{x} & & \geq & 0. \end{array} \tag{1.15}$$

The solution value to (1.15) is a lower bound for the solution value of (1.7) because for any feasible solution $\tilde{\boldsymbol{x}}$ of (1.7) the following equation holds:

$$\boldsymbol{c}^T\tilde{\boldsymbol{x}} \geq \boldsymbol{c}^T\tilde{\boldsymbol{x}} - \boldsymbol{\lambda}^T(A_2\tilde{\boldsymbol{x}} - \boldsymbol{b}_2) \geq \min_{x \geq 0, A_1x \geq b_1} \boldsymbol{c}^T\boldsymbol{x} - \boldsymbol{\lambda}^T(A_2\boldsymbol{x} - \boldsymbol{b}_2) = L(\boldsymbol{\lambda}).$$

To solve (1.7) we maximise problem (1.15) over all $\boldsymbol{\lambda} \geq 0$. This is called the *Lagrangian Dual Problem*:

$$\begin{array}{lccc} \text{Maximise} & L(\boldsymbol{\lambda}) & & \\ \text{subject to} & \boldsymbol{\lambda} & \geq & 0. \end{array} \tag{1.16}$$

The solution to (1.16) can be found with a *subgradient method* (see Schrijver [1986]) which is easy to implement. We start solving (1.15) for a given $\boldsymbol{\lambda}_0$ and obtain a solution $\boldsymbol{x}_0$, then update $\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k - \gamma_k(A_2\boldsymbol{x}_k - \boldsymbol{b}_2)$, and re-solve (1.15). The value γ_k is a specified step length and vector $A_2\boldsymbol{x}_k - \boldsymbol{b}_2$ is called the *subgradient*. This process continues until a stopping criterion is fulfilled, e.g. the gap between a feasible solution of (1.7) and the lower bound obtained by (1.16) is sufficiently small. If the solution of (1.16) is infeasible for (1.7), a heuristic method can be used to obtain a solution for (1.7).

An alternative way to solve the Lagrangian dual problem is to reformulate the

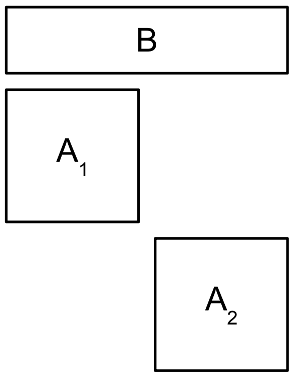


Figure 1.1. Block-diagonal matrix structure.

problem as a linear program with a large number of constraints (see Wolsey [1998]). The LP can then be solved with a branch-and-cut method, which is equivalent to applying Dantzig-Wolfe decomposition to the dual of the LP.

Other methods to solve the Lagrangian dual problem include bundle methods based on quadratic programming (Hiriart-Urruty and Lemaréchal [1993]), analytic centre cutting plane methods based on an interior point algorithm (Goffin and Vial [2002]), or ellipsoid methods (Khachian [1979]).

1.5.4 Comparison of Decomposition Methods

Although the three decomposition methods seem to be very different they are closely related to each other. Problem (1.10) is the dual problem of (1.7). Equivalently, the Dantzig-Wolfe master problem and the Benders master problem are duals of each other. Benders decomposition is therefore equivalent to applying Dantzig-Wolfe decomposition to the dual problem.

Additionally, the following equation holds:

$$\min_{x \geq 0, A_1 x \geq b_1} (\boldsymbol{c}^T - \boldsymbol{\pi}^T A_2)\boldsymbol{x} = \min_{x \geq 0, A_1 x \geq b_1} \boldsymbol{c}^T \boldsymbol{x} - \boldsymbol{\pi}^T (A_2 \boldsymbol{x} - \boldsymbol{b}_2) - \boldsymbol{\pi}^T \boldsymbol{b}_2 = L(\boldsymbol{\pi}) - \boldsymbol{\pi}^T \boldsymbol{b}_2.$$

Therefore, the Dantzig-Wolfe subproblem and the Lagrangian relaxation problem yield the same lower bound on the solution value of the original problem.

In a particular case when the matrix structure is block-diagonal (Figure 1.1), decomposition methods are used very successfully in many applications. When solving practical applications, the problems can be very large and a standard simplex method cannot be applied because of performance and memory issues. Decomposition methods perform well if it is possible to decompose the matrix into several subproblems that are much smaller than the original problem. In the example in Figure 1.1, only the coupling constraints in matrix B are kept in the master problem while the problems with matrices A_1 and A_2 can be solved separately in two different subproblems, often even in parallel. This can enable very fast solution times.

Other methods to solve large-scale linear or integer programs include bundle methods (Hiriart-Urruty and Lemaréchal [1993]), Wedelin's algorithm (Wedelin [1995]), and cross decomposition methods (van Roy [1986]).

Integer Program Decomposition

All decomposition methods described can also be applied to general mixed integer problems. Some care must be taken when conical or convex combinations are computed to ensure integrality of the resulting vector. Which decomposition method is chosen to solve a problem depends on the particular structure of the problem. Often the integer requirements or particular parts of the variables or constraints cause difficulties in which case these should be relaxed. We consider a more general case than (1.7):

$$\begin{array}{lrcl} \text{Minimise} & \boldsymbol{c}^T\boldsymbol{x} & & \\ \text{subject to} & A\boldsymbol{x} & \geq & \boldsymbol{b} \\ & \boldsymbol{x} & \in & X \end{array} \tag{1.17}$$

$X = P \cap \mathbb{Z}_+$ and $P \in \mathbb{R}^n$ a polyhedron. Relaxing constraints $Ax \geq b$ with Dantzig-Wolfe decomposition or Lagrangian relaxation again yields the same bounds. By replacing X with $conv(X)$ in formulation (1.17) both methods can be solved by linear programming, the Lagrangian relaxation with a large number of constraints, the Dantzig-Wolfe decomposition with a large number of variables. These constraints and columns are formed by sets V and W of extreme points and extreme rays of the set $conv(X)$. The Lagrangian relaxation approach yields multipliers for the problem while Dantzig-Wolfe decomposition yields a feasible solution x where the integrality of x remains to be checked. It is not sufficient to use integer convex/conical combinations of elements of sets V and W for x to be integer. However, in the important special case of a binary linear program when $X \in \{0,1\}^n$, all integral points of a bounded set X are already vertices of $conv(X)$ and each x is the trivial convex combination of a single element of V. Note that in the integer case, the primal dual relationship between the master problems of Dantzig-Wolfe and Benders decomposition does not hold any longer but depends on the structure of $conv(X)$.

Which solution approach is used depends on the model and the specific problem instance. Quite often, different solution techniques must be compared on a particular problem in order to verify the best approach. In some cases some models are better suited than others. In case of primal degeneracy for example, dual methods are usually preferred. However, if primal feasible solutions are required throughout the solution phase, e.g. for early termination, a primal method may be the only choice. The choice of a solution method also depends on how fast a solution must be found and how accurate the solution needs to be. A Lagrangian relaxation approach may quickly result in good lower bounds but only a heuristic primal solution. Dantzig-Wolfe decomposition on the other hand may result in a primal feasible solution but the simplex method may require a long time to converge.

Finally, characteristics of the model are also important. It is desirable to obtain subproblems that can be solved quickly. If the subproblems are naturally integer, however, the LP relaxation of the Dantzig-Wolfe master problem is not tighter than the LP relaxation of the original formulation and therefore does not yield an improved lower bound. Hence, the integrality property of the subproblems may be undesirable if the integrality gap of the original problem is large (see Desrosiers et al. [1995]). For further and more in-depth discussion on the topic we refer to Nemhauser and Wolsey [1988], Lübbecke and Desrosiers [2004], and Ralphs and Galati [2006].

1.6 Multiobjective Optimisation

An increasingly important concept in airline scheduling problems as well as in many other areas of operations research is *multiobjective optimisation (or* *multicriteria optimisation).* In all formulations presented in this chapter so far, only a single objective is considered. In practical applications often multiple objectives must be optimised simultaneously. In most cases these objectives contradict each other. In this brief overview we consider an integer program with two objectives (1.18). We refer to Ehrgott [2005] for more details. A *biobjective integer program* is defined as:

$$\begin{aligned} \text{Minimise} \quad z(\boldsymbol{x}) &= \begin{cases} z_1(\boldsymbol{x}) \\ z_2(\boldsymbol{x}) \end{cases} \\ \text{subject to} \quad A\boldsymbol{x} &= \boldsymbol{b} \\ \boldsymbol{x} &\in \mathbb{Z}^n. \end{aligned} \tag{1.18}$$

The *feasible set* is called X. Its image under the objective function is called $Z := z(X)$. We define the following order on the objective space $\mathbb{R}^2$:

$$\boldsymbol{y}^1 \preceq \boldsymbol{y}^2 \quad \Leftrightarrow \quad y_k^1 \leq y_k^2,\ k = 1, 2;\ \boldsymbol{y}^1 \neq \boldsymbol{y}^2; \boldsymbol{y}^1, \boldsymbol{y}^2 \in \mathbb{R}^2.$$

Our goal when solving the biobjective problem is to find feasible solutions such that no other feasible solutions exist that are better with respect to one component of the objective vector $z(\boldsymbol{x})$ and not worse with respect to the second component.

Definition 1 A feasible solution $\hat{\boldsymbol{x}} \in X$ is called *efficient* or *Pareto optimal* if there does not exist any $\boldsymbol{x}' \in X$ with $(z_1(\boldsymbol{x}'), z_2(\boldsymbol{x}')) \preceq (z_1(\hat{\boldsymbol{x}}), z_2(\hat{\boldsymbol{x}}))$. The image $z(\hat{\boldsymbol{x}}) = (z_1(\hat{\boldsymbol{x}}), z_2(\hat{\boldsymbol{x}}))$ of $\hat{\boldsymbol{x}}$ is called *non-dominated.* We distinguish different types of efficient solutions:

- *Supported* efficient solutions are those efficient solutions that can be obtained as optimal solutions to a (single objective) weighted sum problem:

$$\begin{aligned} \text{Minimise} \quad & \lambda^1 z_1(\boldsymbol{x}) + \lambda^2 z_2(\boldsymbol{x}) \\ \text{subject to} \quad & \boldsymbol{x} \in X, \end{aligned} \tag{1.19}$$

  for some $\lambda^1 > 0, \lambda^2 > 0$. The supported non-dominated points lie on the boundary of the convex hull *conv*(Z) of the feasible set in objective space.

- Supported efficient solutions which define an extreme point of $conv(Z)$ are called *extreme* supported efficient solutions.
- The remaining efficient solutions are called *non-supported* efficient solutions. They cannot be obtained as solutions of a weighted sum problem as their image lies in the interior of $conv(Z)$.

An example of all non-dominated points of a problem is given in Figure 1.2.

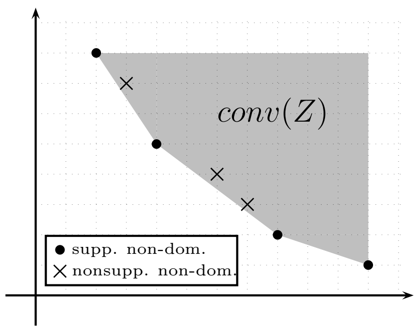


Figure 1.2. Supported and non-supported non-dominated points.

Chapter 2

Airline Scheduling Background and Literature

The operation of an airline requires a large number of decision making and optimisation problems to be solved. An airline needs to solve problems as diverse as forecasting passenger demand, assigning aircraft and crew to all flights they operate, purchasing and maintaining aircraft, handling luggage and cargo, organising catering, handling passengers at check-in and the gate, and taking care of re-accommodation of passengers and crew in case of disruptions.

The complexity of the problems but also the need for finding cost optimal solutions in order to be competitive have motivated a large amount of research in airline optimisation problems over the last 50 years. Optimisation models, heuristics, and simulations are among the *Operations Research (OR)* methods that have been specifically developed or adapted to efficiently solve large scale problems in the airline industry.

The problems can generally be classified into *strategic and tactical planning problems* and *operational (or day-of-operations) problems.*

Strategic problems include decisions about the size and composition of the fleet of the airline, e.g. how many new aircraft of which size should be acquired. Another strategic problem is to decide where to locate crew bases in the flight network and how many crew members are needed at each crew base. The decision to enter a new origin-destination market is also a strategic decision problem.

Among the *tactical planning problems* are airline scheduling, pricing, and revenue management problems, see van Ryzin and Talluri [2002] for a survey on the latter two. The scheduling process usually starts about 12 months before the schedule is operated and lasts until the day-of-operations. The *airline scheduling problem* is usually decomposed into five planning problems and traditionally these problems are solved sequentially. First, marketing decisions in the *schedule design problem* determine which flights the airline operates. Given the set of flights in a schedule the solution of the *fleet assignment problem* determines which flight is operated by which aircraft type. Next, a minimal cost assignment of available aircraft to flights is found in the *aircraft routing problem*. The last of the tactical planning problems is *crew scheduling*, usually decomposed into two consecutive stages, namely *crew pairing* and *crew rostering problems*. Crew members must be assigned to operate all flights in the schedule. Firstly, the *crew pairing problem* (or *tour of duty problem*) allocates generic crews to flights in a minimal cost way. Secondly, in the *crew rostering problem*, monthly or fortnightly *work rosters (or lines of work)* are constructed based on the cost minimal crew pairings and assigned to each individual crew member.

On the day-of-operations a large number of additional *operational problems* must be solved. On one hand the planned schedule must be executed. On the other hand disruptions occur frequently, which makes it necessary to change the planned schedule during operations. Among sources of disruptions are unforeseen maintenance tasks, late passengers, late crew, or bad weather. The execution of the schedule as well as the disruption management usually takes place in the airline operations control centre. If disruptions occur, flights must be delayed or cancelled, aircraft and crew must be re-scheduled and passengers must be re-accommodated. The resulting models are similar to their planning counterparts but usually span a smaller time horizon and must be solved much faster, often in a matter of minutes. Hence, special techniques, such as heuristics, to obtain good solutions quickly are often utilised to solve the operational problems. For many airlines, disruption recovery is a mostly manual process relying on the experience of their schedulers rather than the utilisation of mathematical models. We refer to the following recent contributions for a description of airline operations and recovery procedures: Stojković et al. [1998], Lettovský et al. [2000], Filar et al. [2001], Stojković et al. [2002], Rosenberger et al. [2003], Yu et al. [2003], and Ball et al. [2007].

We describe the airline scheduling process and contributions in the literature in more detail in the following sections. We do not address revenue management, cargo, or passenger aspects of airline operations in this review.

2.1 Airline Scheduling Problems

Airline scheduling consists of five different types of planning problems: schedule design, fleet assignment, aircraft routing, crew pairing, and crew rostering. In this section we summarise each of these problems and describe the literature on solution approaches for each of the problems. We further present approaches in which some of the individual problems are integrated into more comprehensive models. We conclude with formulations that include *robustness* measures. A planned solution is understood to be operationally robust if disruptions of some flights in the schedule have a minimal effect on other flights in the schedule. Recent surveys on airline scheduling problems are provided by Gopalan and Talluri [1998b], Barnhart et al. [2003a], Barnhart and Cohn [2004], and Klabjan [2005]. In the sections that describe the individual problems, only contributions are cited that address a single problem. Contributions addressing multiple problems are listed in Section 2.2.

2.1.1 Problem Characteristics

In this section we present important problem characteristics that are common among many airline scheduling problems.

The literature distinguishes between *daily*, *weekly*, and *dated* scheduling problems.

In the *daily problem* it is assumed that the schedule repeats every day, i.e. the same flights are operated on each day. This is the most common approach described in the literature. Many airlines in North-America operate the same schedule on each weekday and a subset of flights on the weekend. For other airlines the schedule may vary on a day to day basis.

After solving the daily problem the generated solutions for crew and aircraft are repeated every day to obtain a solution for a week. The *weekly exception problem* is then solved to eliminate infeasibilities on the weekends. Alternatively, a *weekly problem* can be solved where it is assumed that the schedule repeats every week but may vary on different days of the week. The solution for a single week is then repeated to obtain a solution for a longer period. Restricting the aircraft or crew solutions to repeat daily can cause suboptimal solutions even if the schedule repeats daily as shown in Andersson et al. [1998].

A more general approach is to solve a fully *dated problem* where no restrictions are imposed between solutions on different days. This problem must be solved when there is a transition from one schedule to another for solutions that span both (different) schedules. Here, specific start and end dates are given for which the problem must be solved. This version of the problem is also commonly solved by airlines where the schedule varies frequently from day to day or week to week. Because of the longer time horizon this problem is much harder to solve than daily or weekly problems and may be intractable for large schedules containing many flights.

With respect to the flight network structure two different models are common. The *hub-and-spoke network* is widely used among airlines in North-America. In this network only large airports (hubs) are linked by direct flights and all smaller airports (spokes) are only connected to a single hub. Many aircraft meet at a hub at the same time ensuring the existence of many feasible connections. This property leads to a very large number of feasible solutions. A second type of network is the *point-to-point (or inter-connected) network.* In contrast to the hub-and-spoke network, in a point-to-point network many airports are linked with multiple other airports by direct flights.

From a modelling point of view the following two different network types are distinguished: *connection networks* and *time-line networks.*

In a *connection network* the nodes represent arrivals or departures of flights. *Flight arcs* represent the flights and *connection arcs* link the arrival of an incoming flight with the departure of an outgoing flight if it is possible to operate these two flights in sequence with the same aircraft. This is the case if the destination of the incoming flight is the origin of the outgoing flight and sufficient time between arrival and departure allows to disembark and embark passengers and to clean, refuel, and reload the aircraft. This minimal required

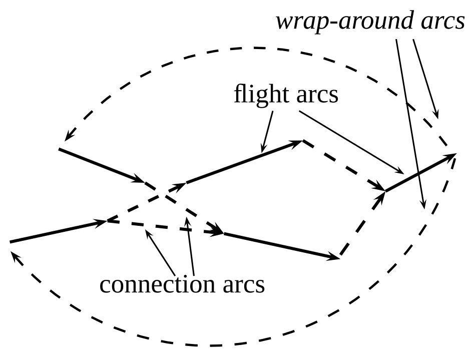


Figure 2.1. Connection network with 5 flight arcs.

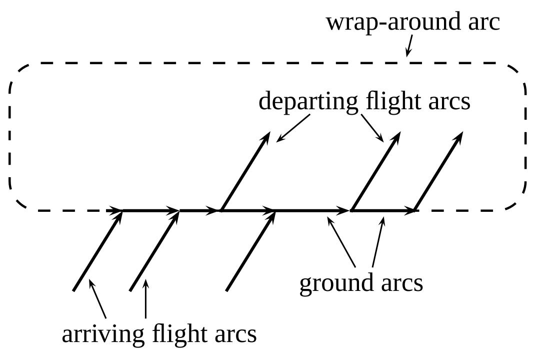


Figure 2.2. Time-line network for a single airport.

time between arrival of the first and departure of the second flight is called *minimal turn-time* for aircraft and *minimal sit-time* for crew.

The *time-line network* consists of nodes for time and location of each departure and each arrival of all flights. The arrival time is hereby adjusted by adding the minimal turn-time (or sit-time). The two types of arcs are flight arcs (as in the connection network) and *ground arcs.* A ground arc links two consecutive activity nodes (departures or arrivals) at the same airport. The flow on a ground arc represents all aircraft or crew on the ground at a particular airport and time.

The time-line network consists of many fewer arcs than the connection network but the model does not distinguish between individual aircraft or crew on ground arcs. If a daily or weekly problem is solved, both networks are extended by *wrap-around arcs* that link the last flights in the schedule with the first flights and link the airport at the end of the horizon with the start, respectively. Wrap-around arcs are needed so that an aircraft routing or a crew pairing can span multiple days in a daily problem. Figures 2.1 and 2.2 show small examples of each network type. A final differentiation between network models can be made depending on the activity represented by an arc. In the *flight based* model an arc represents exactly one flight while in the *duty-period based* model (see Section 2.1.5) an arc can comprise multiple flights that result in a feasible work day. In the second model more feasibility constraints can be included implicitly in the network but many more arcs may exist. We also use the term *duty-period based* for aircraft networks if an arc models a sequence of flights an aircraft can operate.

2.1.2 Schedule Design

The most important decisions for an airline involve the *schedule design.* These decisions determine the profit an airline will achieve and determine the input data for all other airline scheduling problems. The airline must decide in what markets to operate. This includes determining city (or airport, port, station) pairs to connect with direct flights (or sectors, legs), how frequently the flights are offered, at which times of the day, and on which days of the week. These decisions are influenced by demand forecasts for itineraries, the resources the airline has available, and competitor behaviour. The schedule also needs to be operated by crew and aircraft and time slots must be available at airports. Considering all these aspects and solving the schedule design problem to optimality therefore requires us to consider all schedule planning problems and solve them in a single integrated model. This is currently intractable due to the complexity of each individual problem and the large size of the problem.

Another difficulty when constructing a schedule from scratch is that the necessary data is usually not fully available to an airline. Data required includes *unconstrained demand* for all possible origin-destination itineraries for any point in time, which is the largest possible demand without taking actual fares and capacities of origin-destination pairs into account and cannot be observed. The *actual demand* for flights with given capacity depends on the airline's schedule as well as schedules of other airlines while the airline's schedule depends on the demand. Fares must also be assigned to each itinerary and are difficult to estimate. Again, fares are depending on the schedule and also on fares a competitor may introduce in the same market.

From a practical point of view many changes to the airline's airport infrastructures may be necessary if the network structure changes and for operational reasons the airline prefers a high level of consistency from one schedule to the next. This is also important for the loyalty of frequently travelling business customers.

For these reasons, a schedule is usually not constructed from scratch but by adapting a schedule from a previous period. This is usually a manual process driven by marketing decisions. In the literature the schedule design problem is not discussed as a separate optimisation problem. Since other resources, e.g. available aircraft or crew, must be taken into account schedule design is discussed in combination with other airline scheduling problems such as fleet assignment. We describe various approaches that integrate schedule design with other airline scheduling problems in Section 2.2. An exemption is on-demand airline scheduling where a new schedule is constructed each day for a fleet of small jet planes for the following day based on demand. Usually, small regional airports are connected by such a service, allowing the flexibility needed for short-term realisation. Recent contributions can be found in Espinoza et al. [2008a] and Espinoza et al. [2008b].

Instead of constructing a schedule from scratch, a common approach described in the literature is to solve a *schedule augmentation problem*. Here, the original schedule is given and only small deviations from that schedule are permitted. These deviations may be the addition or deletion of sets of flights to or from the schedule (Lohatepanont and Barnhart [2004]) or small deviations in departure times of some flights in the schedule. The latter case is called *time window (or re-timing) problem*. In this problem the flights of the schedule are fixed but departure times vary in some interval around the originally scheduled departure time (Klabjan et al. [2002]).

Recent contributions describing schedule design problems include Büdenbender et al. [2000], Erdmann et al. [2001], Barnhart et al. [2002b], and Armacost et al. [2002].

2.1.3 Fleet Assignment

The *fleet* of an airline consists of all aircraft the airline has available to operate the schedule. These aircraft are usually varying in type, e.g. Boeing 737 or Airbus 320. The type of aircraft determines its capacity and the cost for operating a particular flight. The *fleet assignment problem* decides which aircraft type operates which flight. The objective is to maximise profit while allocating exactly one aircraft type to each flight in the schedule and respect the number of available aircraft of each type.

Profit is usually modelled as the difference between *unconstrained revenue* and *assignment costs*. *Unconstrained revenue* of a schedule is the maximum possible revenue regardless of the capacity of the aircraft type assigned to each flight. *Assignment costs* include flight operating costs, passenger carrying costs and *spill costs*. *Spill costs* arise if demand for a flight exceeds the capacity of the aircraft assigned to that flight and not all passengers can be carried. This results in loss of revenue and passengers get *spilled* onto the flight network. These passengers are either *re-captured* by the same airline or lost to some other airline. Since empty seats would be “wasted” if the capacity exceeds the demand for a flight, the airline must carefully assign aircraft types to flights in order to maximise profit.

Abara [1989] and Hane et al. [1995] propose a *basic fleet assignment model (FAM)* to solve the problem. FAM is a multi-commodity flow problem with additional constraints. Abara [1989] use a connection network to model the flight network while Hane et al. [1995] base the model on a time-line network.

As described in Section 2.1.1, the second network consists of fewer arcs than the first one but it is impossible to distinguish between specific aircraft on ground arcs. For this reason, maintenance requirements for an individual aircraft cannot be guaranteed in time-line networks (see Section 2.1.4 for a description of maintenance requirements). It is possible to add constraints such that the aggregated maintenance requirements over all aircraft are satisfied by the solution. In the connection network each individual aircraft can be modelled but the formulation contains many more variables and may be intractable.

The flow conservation constraints in FAM ensure that each aircraft arriving at an airport is departing from that airport at some later time. Additional constraints ensure that each flight is assigned to exactly one aircraft type and that not more aircraft than available are used of each type. The objective function is a sum of flight and aircraft type specific operating costs (independent of the number of passengers carried), carrying costs depending on the number of passengers on board, spill costs (the sum of all itineraries that could not be carried due to capacity) and recaptured revenue (spilled passengers recaptured on other itineraries).

Hane et al. [1995] consider the daily problem. The model is also called *flight-based fleet assignment model* because spill and re-capture costs are calculated for each flight independently. Since passengers may travel on multi-flight itineraries this method cannot estimate spill accurately because passengers that are spilled from one flight must also be spilled from the other flights of the itinerary. Hane et al. [1995] solve the formulation with an LP based branch-and-bound method.

In all daily models the schedule and demand is assumed to be independent of the weekday which is often not true in practice. Also, the fleet assignment solution is required to be equal on every day which can lead to suboptimal solutions. Solving problems with a large time horizon is currently intractable for fleets of large size.

Barnhart et al. [2002a] describe an enhanced fleet assignment model. When passengers are spilled from one flight in an itinerary the model takes into account effects on demand of other flights in the itinerary. A so called *passenger mix model* is added to the basic FAM formulation. For a schedule with given fleet assignment the passenger mix model determines the minimal cost (carrying cost plus spill cost) flow of passengers through the network such that the capacity of each flight is not exceeded and the unconstrained demand is not violated on any itinerary. This enables more accurate estimation of spill and re-capture costs which leads to improved solutions compared to basic FAM. The enhanced problem is called *itinerary-based fleet assignment model* and is solved with LP based column generation and branch-and-bound techniques. The resulting problem formulation contains many variables and is hard to solve. Barnhart et al. [2006] improve the formulation and its computational tractability.

Kliewer [1996], Belobaba and Farkas [1999], and Yan and Tseng [2002] also describe enhanced demand and revenue models in combination with fleet assignment. The concept of *demand driven dispatch* is introduced by Berge and Hopperstad [1993]. Here, the original assignment of aircraft types can be changed closer to the date of departure once demand forecasts have become more accurate.

Barnhart et al. [1998a], Jarrah and Strehler [2000], and Ahuja et al. [2001] consider the maximisation of *through benefits.* A through connection contains two flights operated by the same aircraft. Passengers prefer direct flights from the origin to the destination of their journey. If no direct flight exists, passengers prefer to stay on the same aircraft during their itinerary. This saves transferring in a terminal and possibly missing a connection as well as possible baggage loss. The additional amount passengers are willing to pay for this convenience is called through benefit. If the same aircraft type is assigned to both flights of a connection this benefit is added to the objective value of the fleet assignment problem.

Other contributions towards the fleet assignment problem include Gu et al. [1994], Subramanian et al. [1994], Talluri [1996], and Rushmeier and Kontogiorgis [1997]. For more detailed information on the fleet assignment problem we refer to the recent overview of concepts, models, and algorithms by Sherali et al. [2006].

2.1.4 Aircraft Routing

In the *aircraft (or maintenance) routing problem* one needs to find sequences of flights, called *routings (or rotations)*, operated consecutively by a single aircraft. A rotation is an aircraft routing that starts and ends at the same airport. Each aircraft regularly needs to undergo different maintenance checks. These need to be performed at a maintenance station before some maximal time between maintenance checks elapses. The goal of the aircraft routing problem is to assign each flight in the schedule to exactly one maintenance feasible aircraft routing. Additionally, one cannot use more aircraft than available.

The required maintenance checks vary in duration and frequency in which they must occur. Only a certain amount of time, flying time and number of takeoffs are allowed to elapse between two consecutive checks. Basic checks such as visual inspections must occur frequently, for example every 36 hours, and last from one to several hours. Other, less frequent but much more thorough checks may disassemble and reassemble the aircraft. For this kind of check the aircraft is taken out of service for several weeks. In aircraft routing formulations usually only the short and medium length checks are considered that occur on a basis of one to several days.

If a fleet assignment problem has been solved prior to the aircraft routing problem, the latter can be solved for each aircraft type separately, as only the flights in the schedule operated by this particular type must be included in the problem formulation. This can reduce the problem size significantly and enable fast solution times. Most of the subsequently described models can be applied to a single or multiple fleet problem. When applied to multiple fleet types, the aircraft routing problem also solves the fleet assignment problem since together with the aircraft also the aircraft type is assigned to each flight.

Some airlines impose the additional condition that each aircraft must fly all flights in the schedule within some certain amount of time, the so called big-cycle constraint. This condition is equivalent to finding an Euler-tour in the underlying network (e.g. Clarke et al. [1997]).

The aircraft routing problem can be extended from only finding a feasible solution to finding a solution that maximises *through revenue*. As in the fleet assignment problem, through revenue is generated if two flights are operated in sequence by the same aircraft (a through connection). Other costs that can be considered are operating costs (e.g. fuel consumption), if these are variable between aircraft of the same type, or costs to increase the robustness of the solutions (see Section 2.3).

The aircraft routing problem is described in detail in Clarke et al. [1997] and Gopalan and Talluri [1998a].

Daskin and Panayotopoulos [1989] consider the problem of assigning routes to aircraft in a hub-and-spoke network. They do not consider maintenance restrictions. The problem is formulated as a set packing formulation and solved with Lagrangian relaxation. Two sets of constraints ensure that each route is assigned to at most one aircraft and that each aircraft is assigned to at most one route for each time period. The second set of constraints is relaxed in the Lagrangian approach which is embedded into a heuristic in order to obtain a feasible solution.

Feo and Bard [1989] combine the aircraft routing problem with the *maintenance base location problem*. The minimal number of maintenance bases that are needed to satisfy four day maintenance requirements for a given schedule is determined. The problem is modelled as a minimal cost multi-commodity network flow problem. The aircraft routings for each day are given as input and must be connected to form maintenance feasible multiple day routings. Because of the size of the problem a two phase heuristic method is used to solve the problem. In the first phase, good routing solutions for independent aircraft are obtained. In the second phase the best routing solutions from phase one are used to determine minimal cost maintenance locations subject to maintenance feasibility. The second phase is modelled as a set covering problem and solved with a greedy heuristic.

Clarke et al. [1997] consider through revenue and two different maintenance checks, a routine check that lasts 4 hours and must be performed every three days and an avionics check that includes the routine check and some additional checks. The avionics check must be performed every four days and lasts five hours. Also, the big-cycle constraint is imposed, which is why the problem is modelled as an Euler-tour problem with side constraints. The side constraints ensure that no maintenance requirement is violated. The formulation is equivalent to an asymmetric travelling salesman problem with side constraints. Clarke et al. [1997] solve the problem with Lagrangian relaxation by relaxing sub-tour elimination constraints and maintenance feasibility constraints and adding them dynamically once they are violated. To prove optimality the procedure is embedded in a computationally expensive branch-and-bound method.

Gopalan and Talluri [1998a] and Talluri [1998] consider daily maintenance routing with a maintenance check required every three or four days and a periodically required balance-check, resulting in an Euler-tour problem. All maintenance occurs at night when all aircraft are grounded. As a first step of the solution process the feasible connections of the network during the day between non-overnight stations are limited by applying first-in-first-out or last-in-first-out heuristics. The resulting network contains one arc for each sequence of flights between overnight stations and nodes for overnight stations. In a second step the maintenance routing problem is solved on this reduced network where the three day maintenance requirement is taken into account. Fixing connections can cause the existence of sub-tours called *locked rotations*. A heuristic is used to swap flights to unlock those rotations and improve the maintenance routing.

Sriram and Haghani [2003] consider a weekly maintenance routing problem with two different maintenance checks. It is modelled as a multi-commodity network flow problem where the routings during each day are required as input. The model results in a complex linear formulation and is solved by heuristic local search.

Grönkvist [2006] combines constraint programming and column generation techniques to solve the *tail assignment problem*. In tail assignment individual aircraft are considered rather than generic maintenance feasible aircraft routings. Usually this problem is only solved a few days prior to the day of operation. The flight network is modelled as a connection network where arcs represent connections between flights. The tail assignment problem is modelled as a set partitioning formulation (see Chapter 3) where the constraints ensure that each flight is operated by exactly one aircraft. Constraint programming is used in a preprocessing step in order to reduce the number of arcs. Connection arcs violating the number of available aircraft and arcs that cannot be part of any feasible solution are removed. Grönkvist [2006] achieves a significant reduction in the number of arcs. After the preprocessing step, column generation and a heuristic fixing process are used to obtain integer solutions for the simplified problem.

Sarac et al. [2006] consider the aircraft routing problem on an operational level rather than a planning level. The model is a set partitioning formulation with additional constraints to ensure sufficient maintenance capacity at the maintenance bases and is solved via branch-and-price. The set partitioning constraints ensure that each aircraft is assigned to exactly one routing and each flight is operated by exactly one aircraft. The additional constraints ensure the availability of maintenance slots and man power to carry out the required maintenance checks. Due to these additional constraints the branching strategy described in Section 3.3.4 must be altered. Sarac et al. [2006] use a combination of follow-on (see Section 4.4.3) and aircraft-flight pair (see Section 3.3.4) branching to obtain integer solutions.

2.1.5 Crew Pairing

Similar to aircraft routings, *crew pairings (or tours-of-duty)* are sets of flights which can be operated in sequence by the same crew. Additionally, the pairings must start and end at the same crew base and satisfy all sorts of work regulations. The goal of the crew pairing problem is to find a minimal cost set of crew pairings such that each flight is contained in exactly one pairing. Usually crew pairings are divided into *duty periods*. A duty period spans one or multiple flights on a single workday. A crew pairing consists of one or multiple duty periods which are separated by (over-night) rest periods. The construction of legal pairings is subject to a large number of rules imposed by civil aviation regulation authorities, employment contracts, and agreements.

Rules include maximal allowed flying time per duty period, maximal allowed flying time in a rolling time window, minimal rest requirements, or meal break requirements. "The maximum allowed duty time is 8 hours for any rolling 24 hour time period" is an explicit example of such a rule.

After fuel costs, crew salary is the second largest operational cost an airline has to account for. Therefore finding a minimal cost solution to the crew pairing problem is very important for an airline. Cost is usually a nonlinear function of flying time, total elapsed work time, and time away from the home base. Besides paid hours (productive and unproductive), costs can be included for ground transport, meals, accommodation, and the cost of *passengering* crew within the pairing. A transfer of crew is referred to as *passengering* or *deadheading* if crew are travelling as passengers. This is necessary if crew are required to operate a flight that does not depart at their current location or to return to their home base.

The problem can be solved separately for different crew types. Different rules apply to technical crew (i.e. captains and first officers) and cabin crew (i.e. flight attendants) and while technical crew usually stay together during a duty period it is possible to split up cabin crew after a flight and rejoin them with other crew members to operate subsequent flights. Also, most crew are only qualified to operate a particular aircraft type (especially technical crew) or a family of very similar aircraft types. In this case the crew pairing problem can be solved for each aircraft type or family separately. For cabin crew, multiple crew members are required on each flight depending on the size of aircraft and possibly the number of passengers transported. Although cabin crew can be split up after operating on a large aircraft to operate on different smaller aircraft subsequently, from an operational as well as a robustness (see Section 2.3) point of view it is desirable to keep crew together as much as possible which is referred to as *unit crewing.* It is also possible to replace a crew member with a higher ranked crew member, e.g. replace a first officer with a captain, which is called *rank over-covering.*

The literature focuses on the technical crew problem because potential cost savings are much higher than for cabin crew. The flight attendant problem is for example considered in Kwok and Wu [1996]. Wallace [2001] considers the international (long-haul) flight attendant crew pairing problem for schedules from Air New Zealand. Flight attendants are usually qualified to operate on several different aircraft types (of the same family) and the number of flight attendants required depends on the aircraft type. Additional complexity arises from the possibility to split up a crew and re-join the crew members with members from another crew. Wallace [2001] uses a combination of column and row generation to solve the problem.

An airline usually operates out of a number of different crew bases located in cities within the airline's flight network. At each crew base a certain number of crew members is available in any time interval. The required crew members of the crew pairing solution must meet the available resources for each crew base.

The crew pairing problem is computationally challenging for two reasons. Each pairing has a very complicated rule and cost structure. Additionally, a very large number of feasible pairings exist. For large schedules the total enumeration of all possible pairings is therefore often intractable. It is also important to find good quality solutions to the problem since a few percent improvement in cost can yield multi million dollar savings in crew salaries over the year. For these reasons the crew pairing problem has received a lot of attention in the literature. Here we review some of the most important formulations. Desaulniers et al. [1998] and Barnhart et al. [2003b] describe the crew pairing problem and related literature in detail and Gopalakrishnan and Johnson [2005] give a comprehensive overview of state-of-the-art solution methods.

In order to simplify the problem, often a daily optimisation problem is solved first (see Section 2.1.1). The daily solution is then repeated to cover the schedule of the whole week. Since the schedule is usually different on the weekend some pairings will be infeasible during the weekend and are called *broken pairings*. These infeasibilities are resolved in the weekly exception problem.

A further classification of standard approaches can be made by the network type that is used to model the flight network. We distinguish *flight networks*, (e.g. Graves et al. [1993]) and *duty period networks* (e.g. Lavoie et al. [1988] and Barnhart et al. [1994]). The *flight network* consists of nodes for each departure and arrival as well as flight and connection arcs linking the nodes. The *duty period network* contains arcs for duty periods and for overnight rests. The nodes in this network represent the start or the end of duty periods, respectively. The duty period network contains many arcs but all duty legality rules can be embedded in the network. In both networks pairings are represented as paths in the network. The choice of the network depends on the problem structure. As a general rule of thumb, duty period networks are preferable whenever the total number of feasible duty periods does not exceed the total number of flights in the schedule by more than a small factor. Since in international schedules duty periods rarely consist of more than one flight, these schedules are often modelled as duty period networks. For domestic schedules on the other hand, where duty periods can contain several flights, the flight network is generally used because the total number of feasible duties is very large.

Until the 1990s, local improvement heuristics were mainly used to solve the crew pairing problem due to the lack in computational power and because heuristics are relatively easy to implement. In general, heuristics are not able to provide a bound on the quality of the solution and are unable to guarantee to find a feasible solution even if such a solution exists. Because of these drawbacks, today the use of optimisation methods or optimisation based methods is clearly favoured when solving the crew pairing problem.

The crew pairing problem is most commonly modelled as a set partitioning problem where the constraints ensure that each flight is operated by exactly one crew. Resource limitations at the crew bases can be included by adding two-sided knapsack constraints (called *base constraints*) to the formulation. The set partitioning model is usually solved with LP based branch-and-bound methods and column generation techniques (see Chapter 1).

When deadheading is allowed, the formulation can be changed to a set covering formulation requiring each flight to be covered at least once. This formulation has the drawback that pairing rules that are different for passengering on a flight instead of operating the flight cannot be modelled. In a set partitioning formulation, deadheading can be considered in the column generation process but without including the passengering flight in the column of the matrix. Two pairings that are identical except for an additional deadhead flight that is contained in only one of the pairings result in identical columns of the matrix. The two columns can be distinguished by their costs.

When modelled as a set partitioning problem, two individual problems must be solved. The first is to construct feasible pairings and the second is to solve the IP formulation.

Enumerating all pairings a priori is intractable for medium or large sized schedules because of the large number of feasible pairings. One historic approach is to only generate all pairings over a subset of flights (Anbil et al. [1991], Gershkoff [1989]). This approach is called *local search row approach* because in each iteration only a subset of all rows (constraints) of the formulation is considered. Starting from a feasible solution, a small number of pairings is chosen by some heuristic rule. The set of flights contained in the chosen pairings form a subproblem of the original set partitioning problem. All feasible pairings are generated for the flights contained in the subproblem and the subproblem is then solved to optimality. The columns in the original problem that are covering the flights of the subproblem are replaced with the optimal pairings. Then, another set of pairings is chosen and a new subproblem is solved. Many iterations are needed to find good solutions and the procedure can get stuck in local optima.

To avoid local optima and the enumeration of all pairings, the set partitioning formulation is nowadays usually solved by column generation methods. Here, the IP forms the master problem and in a column generation subproblem pairings with negative reduced costs are generated. The solution process iteratively solves both problems until no pairings with negative reduced cost can be found.

Pairing generation can be achieved in flight or duty period networks. Crew pairings are represented as paths in both networks. Three common approaches exist to find pairings with low reduced costs:

Using resource constrained shortest path, a label must be maintained for each feasibility rule and each nonlinear cost component as in Desrochers and Soumis [1989], Barnhart et al. [1994], Vance et al. [1997a] and Desaulniers et al. [1997a]. See also Section 3.3.3 for details.

Galia and Hjörring [2003] describe a k-shortest path approach. They first find a shortest path. If the path is feasible and has negative reduced cost it is returned, if it is infeasible the second shortest path is considered. This process continues as long as the k-shortest path incurs negative reduced cost.

A third approach is to perform depth first search enumeration of the pairings, e.g. Anbil et al. [1998], Andersson et al. [1998], Klabjan et al. [2001a], and Makri and Klabjan [2004]. Andersson et al. [1998] describe how pairing generation is performed at Carmen Systems. This approach separates the algorithm from the rules that are applicable to the pairings. All rules can be defined by the user in a specialised rule language. The pairings can then be checked by the rule system for feasibility.

Klabjan et al. [2001a] describe the generation of random pairings which they add to a set of pairings with low reduced costs. Connections for the pairings are picked randomly in such a way that the probability of selecting a connection increases with shorter sit-time.

Solving the IP formulation is decomposed into two phases. First the LP relaxation is solved with the simplex method. Pairings can either be generated a priori or during the algorithm with column generation. In the second phase a branching scheme is used to obtain an integer solution to the problem. Pairings can be generated only during the LP phase and the IP is solved for this fixed set of pairings. This procedure is called branch-and-bound. If instead, pairings are also generated during the branching process in the IP phase, the method is referred to as branch-and-price.

In the IP phase a *constraint branching rule* should be used instead of a *variable branching rule*. The latter will either force the existence or non-existence of a particular variable in the solution. This is difficult and time consuming to enforce inside the column generation procedure. It also does not yield a balanced branch-and-bound tree since forcing a variable into the solution eliminates all other variables with entries in common rows, but forbidding a variable does not restrict the solution space significantly. A *constraint branching rule* that is particularly well suited for crew-pairing-like set partitioning problems is branching on *follow-on* sector pairs (Ryan and Foster [1981]). Two flights must be covered by the same crew in sequence or are not permitted to be covered in sequence. This rule is used for example in Anbil et al. [1992], Anbil et al. [1998], Barnhart et al. [1994], Vance et al. [1997a], and Desaulniers et al. [1997a]. The rule can easily be enforced in the column generation network by removing arcs. Another constraint branching rule is called *time-line branching* and is proposed by Klabjan et al. [2001a]. In one branch the connection time between a particular flight and the next flight must be below some threshold and in the other branch it must be above the threshold. If all flights in the schedule depart at different times, time-line branching is a valid branching rule. This can be achieved by slightly perturbing the departure times.

Alternative approaches to solve the crew pairing problem include Vance et al. [1997b], Desaulniers et al. [1997a], and Andersson et al. [1998]. Vance et al. [1997b] decompose the problem into two stages, first partitioning the flights by duty periods and then the duty periods by pairings. They use Dantzig-Wolfe decomposition to solve the problem, the flight set partitioning constraints are forming the subproblem. This formulation yields a tighter LP bound but is harder to solve than the standard set partitioning formulation. Desaulniers et al. [1997a] use a nonlinear multi-commodity network flow formulation and solve it with a Dantzig-Wolfe decomposition method. The master problem becomes a set partitioning problem for the flights and the subproblems are resource constrained shortest path problems that determine feasible crew pairings. Andersson et al. [1998] formulate the crew pairing problem as a set covering problem. Wedelins algorithm (Wedelin [1995]) is used to solve the resulting optimisation problem.

Further references addressing the crew pairing problem include Hoffman and Padberg [1993], Barnhart et al. [1995], Chu et al. [1997], Barnhart and Shenoi [1998], Butchers et al. [2001], and Klabjan et al. [2001b]. Hoffman and Padberg [1993] use a branch-and-cut approach to solve the crew pairing problem. The long-haul problem is addressed in Barnhart et al. [1995] and Barnhart and Shenoi [1998], Barnhart et al. [1995] focusing on the assignment of passengering flights. Butchers et al. [2001] give details on the crew pairing solution methods at Air New Zealand. A weekly crew pairing problem is solved in Klabjan et al. [2001b]. Next to cost, a second objective is introduced ensuring that the pairings are as similar as possible on each day of the week.

2.1.6 Crew Rostering

The last of the planning problems is *crew rostering*. Monthly (or fortnightly) work schedules (also called *line-of-work*) must be assigned to each individual crew member. These are constructed by concatenating the pairings from the previous problem. Aside from pairings, the work schedules contain activities such as reserve duties, days off, leave, and training periods. Again, many work regulations such as rest time requirements and time limits on the working periods have to be satisfied. The objective is again to minimise cost but more importantly to maximise crew satisfaction by constructing high quality rosters with respect to crew preferences.

In North-America rostering is usually a two phase process. In the first phase generic rosters, called *bidlines*, are constructed (see e.g. Christou et al. [1999]). Then, individual crew members bid on the published rosters and the assignment is based on crew priority, often seniority. An advantage is, that the crew member knows exactly what work to expect if the bid is successful. But conflicts can occur between the assigned rosters and pre-assigned tasks such as vacation or training periods. In that case some rosters can only be partially assigned and additional crew is required causing a more expensive solution. This approach is called *bidline* approach.

Outside of North-America it is common that schedules are constructed and assigned directly to each crew member individually, which is called *personalised rostering* (see e.g. Kohl and Karisch [2004]). Here, the crew members express preferences for certain attributes of the roster without knowing the exact line-of-work they will be assigned to. In this approach either certain quality criteria are maximised for each roster or individual preferences of each crew member are considered. The preferences can either be assigned with respect to seniority, the most senior crew members get as many of their preferences awarded as possible, or on an equal share basis.

The solution method for both types of rostering problems is very similar. As the crew pairing problem, the crew rostering problem is most commonly solved as a set partitioning problem. Constraints ensure that each activity is assigned to some crew member. Additional GUB-constraints (see Section 1.1) ensure that each individual crew member is assigned to exactly one roster. Ryan [1992] first models the rostering problem as a set partitioning problem. To decrease the problem size, not all possible columns are considered but only a precomputed subset. For a given duty a limited number of duties is chosen that can be operated next by any crew member. Also, a number of activities such as desired days off or training tasks are preassigned to crew members.

These techniques ensure that no line-of-works are constructed with unwanted characteristics such as too many days off between tasks or undesired sequences of duties. The techniques also result in a matrix which is more balanced (see Section 1.1) and, hence, the solution is expected to contain fewer fractions simplifying the IP solution process.

There are two versions of the crew rostering problem, the *short-haul* and the *long-haul* problem which are structurally very different. In the short-haul problem each work period consists of many short duties which implies that the columns in the set partitioning formulation contain many ones. The long-haul problem contains much fewer longer duties per work period and hence the columns are less dense. For this reason the short-haul problem is much harder to solve than the long haul problem.

Gamache and Soumis [1998] describe an optimality approach for the rostering problem. This approach is based on a set partitioning formulation and solved by column generation and branch-and-price. They do not pre-assign any activity. The subproblem is a resource constrained shortest path problem modelled on a connection network with work-patterns (pairings) represented as nodes and arcs connecting two nodes if it is possible to work both pairings in sequence. They use constraint branching and a disjoint column generation strategy to speed up the solution process. One subproblem is solved for each employee and, in order to prevent identical columns in different subproblems, all nodes contained in a negative reduced cost column, obtained from a previously solved subproblem, are removed.

Further contributions that address the crew rostering problem include Day and Ryan [1997], Gamache et al. [1998], Gamache et al. [1999], Cappanera and Gallo [2001], and Kohl and Karisch [2004].

Day and Ryan [1997] describe the rostering process for the short-haul problem at Air New Zealand. Rostering is decomposed into two phases. In the first phase off days are assigned and in the second phase pairings and other activities are assigned between the off days. Both problems are solved by column generation.

Gamache et al. [1998] describe the preferential bidding system at Air Canada. Because of a strict seniority principle one crew member can be considered at a time and the assigned pairings are eliminated for subsequent problems when less senior crew members are considered. Column generation and branch-and-bound is used to solve the IP formulations. Also, Gamache et al. [1999] report results from Air France.

Cappanera and Gallo [2001] formulate the problem as a multi-commodity network flow problem. They tighten the formulation with valid inequalities and use an exact IP solution approach.

Kohl and Karisch [2004] give a comprehensive overview on the aircrew rostering problem. They also give some details on the rostering procedure at Carmen Systems. Depth first search is used to generate rosters on a graph that contains nodes for activities and an arc between two nodes if it is possible to assign both activities in sequence. During the construction of a roster a rule evaluation algorithm is called that verifies if a partial roster can be extended to a feasible roster. This procedure is chosen rather than standard column generation methods to separate the rules from the optimisation algorithm.

Recently, Ernst et al. [2004a] provide an extensive annotated bibliography of rostering problems. Other recent contributions that address the rostering problem include Dawid et al. [2001], Sellmann et al. [2002], and Thiel [2005].

2.2 Integration of Airline Scheduling Problems

Traditionally, airline scheduling problems have been solved sequentially although all five scheduling problems are interdependent. Among others the following dependencies exist in the sequential solution approach. The schedule design problem determines the set of flights that must be considered by all subsequent problems. But cheaper fleet assignment or crew pairing solutions might exist if the schedule could be altered slightly. In the aircraft routing problem a subset of flights, determined in the fleet assignment problem, is considered. Rules in the crew pairing problem depend on the underlying aircraft routing solution. And finally, the crew pairings are combined to rosters in the crew rostering problem.

Ideally, all airline scheduling problems should be considered in one integrated model. It is also desirable to include revenue management decisions and passenger aspects into the formulation. Currently, however, such a total integration of all problems is computationally intractable. Each single problem is already hard to solve and requires specialised solution techniques as described in the previous sections. Combining two or more problems in one integrated formulation usually increases the complexity of the problem and often makes it impossible to solve the problem efficiently.

Considerable progress has been made in the past 10 years to integrate two or more problems into tractable models. As first steps towards total integration, two problems are considered either in an integrated model or by solving one problem while considering important aspects of another one. In the following we describe such integration approaches. We conclude with the description of recent formulations that integrate aspects of three of the original problems.

Integration of Fleet Assignment and Schedule Design

Early approaches integrating FAM and schedule design work iteratively (see Etschmaier and Mathaisel [1985] for a survey). Demands for a given schedule are evaluated first. Then, FAM is solved and in the resulting schedule flights for addition and deletion are identified. With this new set of flights the demand for the schedule is evaluated again.

Rexing et al. [2000] integrate the basic FAM problem and the time window problem with the goal of increasing revenue. Their model uses the time-line network. They discretise the time windows and add copies of flight arcs for each possible departure time. Additional constraints ensure that exactly one copy of each flight is operated. Preprocessing of the network is necessary before the problem can be solved for realistic sized problems. To avoid solving the large LP formulation, they introduce an iterative approach. First, all multiple copies of flight arcs are replaced by a single flight arc with reduced duration. This flight arc departs at the end of the departure time window and arrives at the beginning of the arrival time window. This network contains as many arcs as the basic FAM formulation. If the solution is feasible for the original problem the algorithm terminates with an optimal solution. Otherwise flight pairs are identified for which the minimal connection time is violated. For these flight pairs the real duration flight arcs (one copy for each departure time) are re-introduced in the model and the problem is re-solved. Many iterations may be necessary if time windows are large.

Another integrated model for schedule design and fleet assignment is presented by Lohatepanont and Barnhart [2004]. They use the origin-destination fleet assignment model. They distinguish between a mandatory set of flights that must be assigned to aircraft types and optional flights that can or can not be included in the solution in order to maximise profit. They use column and row generation and branch-and-bound to solve the model. As an additional difficulty that needs to be taken into account, removing or adding flights to the schedule can change the demand on other flights.

Other recent approaches that integrate FAM and schedule design include Yan and Tseng [2002], Ahuja et al. [2004], and Bélanger et al. [2006].

Integration of Fleet Assignment and Aircraft Routing

A weekly aircraft routing problem is modelled as a set partitioning problem using a *string* formulation by Barnhart et al. [1998a]. A *string* is a maintenance feasible sequence of flights that starts and ends at a maintenance base. The set of flights is partitioned by maintenance feasible strings and the aircraft can be of different fleet types. The big-cycle constraint can be modelled similar to sub-tour elimination constraints in the travelling salesman problem. The authors solve the model by branch-and-price. The subproblem is a resource constrained shortest path problem on a connection network with labels for each maintenance type and for each nonlinear cost component.

Integration of Fleet Assignment and Crew Pairing

Barnhart et al. [1998c] partially integrate fleet assignment and crew pairing in a multi-commodity flow formulation. They enhance the basic FAM model by adding an approximation of the crew pairing problem based on a duty period formulation. Not all constraints of the original crew pairing model are considered and costs are underestimated. After this enhanced FAM model is solved crew pairing problems are solved for each fleet type. They report considerable savings in cost caused by making (slightly more expensive) decisions in the fleet assignment problem that enable much cheaper crew pairing solutions.

Integration of Aircraft Routing and Crew Pairing

The pairings generated in the crew pairing problem depend on the aircraft routings as follows. A pair of flights form a *connection* if both can be operated in sequence by the same crew or aircraft. The *turn-time* (for aircraft) or *sit-time* (for crew) is the time between the arrival of the inbound flight and the departure of the outbound flight of a connection. All turn/sit-times must exceed a lower bound for the routing or pairing to be feasible. This lower bound is called minimal turn-time for aircraft and minimal sit-time for crew (see Section 2.1.1). The minimal sit-time can exceed the minimal turn-time but when crew stay on the same aircraft, the minimal turn-time applies to both, aircraft and crew. The feasible solution space of the crew pairing problem is therefore limited by the previously solved aircraft routing problem leading to a suboptimal solution from a more comprehensive point of view.

A model to integrate aircraft routing and crew pairing is proposed by Cordeau et al. [2001] and also by Mercier et al. [2005]. They use Benders decomposition (Benders [1962]) and branch-and-price to solve the model. Cordeau et al. [2001] model the aircraft routing problem as the master problem while Mercier et al. [2005] employ the crew pairing problem as the master problem. Since most of the cost is originating from crew, in the second approach the aircraft routing problem only transfers feasibility information back to the master problem, while in the first approach also optimality information must be transferred to the master problem. For this reason the latter approach can solve larger problems in less computation time. Both approaches add inequalities with highly fractional coefficients to the set partitioning polytopes of the problems which causes slow convergence towards an optimal solution.

Cohn and Barnhart [2003] also integrate aircraft routing and crew pairing problems. They extend the crew pairing problem by using the aircraft routing problem as a second column generator next to the crew pairing generator. For each solution of the aircraft routing problem, one variable is added to the crew pairing problem and a convexity constraint ensures the selection of exactly one of the aircraft routing solutions in the final solution of the problem. LP based branch-and-price is used in this computationally expensive solution method. Mercier et al. [2005] report that their Benders decomposition approach yields better solutions in less computation time than the extended crew pairing model of Cohn and Barnhart [2003].

Integration of Fleet Assignment, Aircraft Routing, and Crew Pairing

Clarke et al. [1996] include maintenance and crew considerations into FAM. Overall maintenance requirement constraints for all aircraft of the same fleet are added to the basic FAM. Base constraints are added to fulfil crew flying time requirements. They use the dual steepest edge simplex to solve the LP relaxation and a *fixing procedure* and branch-and-bound algorithm to obtain integer solutions. In the *fixing* step, variables with a fractional value close to 1 are set to 1, before continuing with the branch-and-bound process. This approach does not guarantee the feasibility or optimality of the subsequently solved aircraft routing or crew pairing problem.

Rushmeier and Kontogiorgis [1997] also include aggregated aircraft maintenance and crew considerations into the basic FAM model. First the LP relaxation of the multi-commodity flow formulation is solved. Integer solutions are obtained with a fixing heuristic and a branch-and-bound procedure.

Recently, Papadakos [2007] fully integrate the fleet assignment, aircraft routing, and crew pairing problems as an extension of the model of Mercier et al. [2005]. They use simplified crew pairing costs and rules. They use Benders decomposition to solve the problem by solving one crew pairing subproblem for each fleet. They use Dantzig deepest-cut pricing as well as a dominance relaxed constrained shortest path algorithm (see Section 4.4.2) to solve the subproblems. The deepest cut heuristic is also used in the fleet assignment master problem. To obtain integer solutions the authors first branch on the fleet variables. Once all flights are separated by fleet type, they solve a maintenance routing problem for each fleet and branch on follow-on flight pairs. A heuristic depth-first branching routine is used. The solution method is enhanced by generating Pareto optimal cuts. They show cost savings compared to FAM with maintenance routing (Barnhart et al. [1998a]) and the integrated model of Mercier et al. [2005].

Sandhu and Klabjan [2007] partially integrate fleet assignment, aircraft routing, and crew pairing with a similar approach as Klabjan et al. [2002] and solve the model with both Lagrangian relaxation and Benders decomposition.

While fleet assignment and crew pairing are considered in their original formulations, only *plane-count constraints* are added to model the maintenance routing requirements. *Plane-count constraints* ensure that only the number of available aircraft is used at any time. This works well for hub-and-spoke networks but the results are not as good for point-to-point networks that do not contain as many feasible connections as hub-and-spoke networks.

Integration of Schedule Design, Fleet Assignment, and Aircraft Routing

Desaulniers et al. [1997b] integrate the basic FAM problem, the aircraft routing problem, and the time window problem to increase revenue. They formulate a set partitioning and a multi-commodity network flow model and solve the models with column generation and branch-and-bound. The second model is also decomposed with Dantzig-Wolfe decomposition with flight covering and aircraft flow conservation constraints forming the master problem. This model is an extension of the model of Abara [1989] with added time window constraints and constraints that ensure the feasibility of used connections.

Erdmann et al. [2001] solve the schedule design problem for a charter airline and explicitly model aircraft routes for each aircraft in the fleet. They solve the mixed integer path based formulation with branch-and-cut-and-price where aircraft routing and passenger itinerary subproblems must be solved.

Ioachim et al. [1999] integrate time window, fleet assignment, and aircraft routing problems in a multi-commodity flow formulation. Aircraft of different types must be assigned to flights in a schedule of one week and the departure times of the flights vary in some window. Moreover, flights are labelled with an identifier and flights on different days with the same identifier must depart at the same time. Hence, departure time synchronisation constraints are needed. The model is solved with a Dantzig-Wolfe column generation approach embedded in a branch-and-bound framework to obtain integer solutions. Results on a weekly schedule are given.

Integration of Schedule Design, Aircraft Routing, and Crew Pairing

Klabjan et al. [2002] partially integrate aircraft routing, crew pairing, and schedule design. They reverse the order of the crew pairing and aircraft routing problems. Plane-count constraints are added to the crew pairing problem to ensure the existence of a feasible solution for the aircraft routing problem. Their results are based on a hub-and-spoke network. To include schedule design in the model the departure time of each flight varies in some time window. This is done by relaxing feasibility parameters in the crew pairing problem and hence generating a larger set of pairings. Each feasible pairing has a departure time attached to each flight contained in the pairing. Klabjan et al. [2002] solve the crew pairing problem via an LP based branch-and-bound algorithm.

Cordeau et al. [2001] also reverse the sequential approach and try to solve the crew pairing problem first, followed by the aircraft routing problem (Klabjan et al. [2002]). They apply this approach to a point-to-point network but were not successful in obtaining feasible solutions for the aircraft routing problem.

Mercier and Soumis [2007] extend their model (Mercier et al. [2005]) and integrate aircraft routing and crew pairing with time windows for the departure times. Flights may depart five minutes earlier or later than originally scheduled. Binary variables are used to indicate which departure time is assigned to a flight. Constraints, counting the binary departure time variables for the crew and aircraft solutions, ensure that the same departure times are used in the solutions of both problems. Again, the authors use Benders decomposition to solve the problem.

Integrated Vehicle Scheduling Models

Besides the literature specialised on airline scheduling problems, a large number of publications in the area of vehicle routing (Cordeau et al. [2007]) exist, dealing with very similar problems. Haase et al. [2001], Freling et al. [2003] and Huisman et al. [2005] propose models to integrate vehicle and crew scheduling. Borndörfer et al. [2002] and Borndörfer et al. [2004] describe a proximal bundle method for the integrated multi-depot vehicle and duty scheduling problem in public transit.

2.3 Robustness

The common goal of most problem formulations in the previous section is to find a cost minimising or profit maximising solution from a planning perspective. Deterministic flying and turn around times are assumed in all previous models. Solutions for airline scheduling problems are usually the least expensive if crew or aircraft spend only a minimal amount of time on the ground between arrival and departure of flights and hence the total working or operating hours are minimised. Once disruptions occur in operations, due to delayed passengers, bad weather, or mechanical failures for example, these solutions may appear brittle in that short delays can cause very severe disruptions. Because insufficient buffers are available between flights to compensate for delays, a single initial delay can quickly propagate throughout a large part of the schedule affecting many flights. This sort of disruption may incur large *recovery costs* caused by additional crew requirements, compensation for passengers with delayed or cancelled flights, and damaged reputation of the airline. In an attempt to find *robust solutions*, the objective that needs to be minimised is the sum of planned costs and recovery costs (referred to as *operational costs*) rather than just planned costs as in the previous sections. In a *robust solution* disruptions in some part of the schedule have a minimal effect on other parts of the schedule.

It is easy to measure the performance of the schedule - once it has been operated - in total minutes of delay that occurred during the operation. It is difficult, however, to predict the total minutes of delay or to attach costs to them. It is, for example, difficult to estimate how many minutes of delay will result in a customer not booking with the airline again and the associated loss of revenue. The total minutes of delay occurring during operations are a sum of initial delays caused by unforeseen events and consequential delays that are caused by flights being delayed or cancelled because of the initial delay. The decisions made when planning the schedule cannot affect the first type of delay but influence the second type of delay. It is difficult to estimate the consequential delays that will occur during operations because they largely depend on the strategy an airline uses to recover from disruptions, which is a mostly manual process for many airlines. The decisions made during this manual process include delaying or cancelling flights and re-routing aircraft, crew, and passengers. This recovery process needs to be taken into account when the minutes of delay are forecast for a planned schedule.

A common measure for robustness among airlines is *on time performance (OTP)*, i.e. the percentage of all flights in the schedule that depart on time. A flight is usually referred to as on time if it departs within 10 minutes of the scheduled departure time.

Because recovery costs are difficult to measure from a planning perspective, recent attempts in the literature add various *robustness measures* to different airline scheduling problem formulations to estimate recovery costs. Considering a robustness measure as an optimisation goal leads to potentially more robust solutions, i.e. solutions that are less vulnerable to disruptions, and hence result in low recovery costs and high OTP. In the following we summarise various robustness measures that have been introduced for the schedule design, fleet assignment, aircraft routing, and crew pairing problems.

Robustness and Flight Re-timing

Recently, the topic of flight re-timing in combination with robustness has become increasingly popular in the literature. Recent contributions include Lan et al. [2006], Wu [2006], Burke et al. [2007], Fuhr [2007], and AhmedBeygi et al. [2008].

The approach of Lan et al. [2006] is twofold. In a first approach they find aircraft routings that minimise the propagation of delay by using historic distributions of delay. In a second approach Lan et al. [2006] re-time the flights of the schedule in order to minimise the number of missed passenger connections.

Instead of a simulation Fuhr [2007] propose an analytic approach to evaluate performance. They solve an approximation of the analytical model.

AhmedBeygi et al. [2008] redistribute slack in the planned schedule in order to minimise the effects of disruptions while leaving the aircraft routing and crew pairing solutions fixed. This approach is very similar to our independently developed re-timing approach described in Section 6.3.2. They use a probability of delay as a measure of robustness. In a first step only delays that follow immediately the initial disruption are considered while subsequently all down-stream effects are considered. They use a different model to solve the problem which is larger than ours but possesses some nice integer properties.

Wu [2006] and Burke et al. [2007] consider flight re-timing to enable robust aircraft routings and are described below.

Robustness and Fleet Assignment

For FAM, uncertainty of departure times is taken into account by Rosenberger et al. [2004], Smith and Johnson [2006], Kang [2003], and Bian et al. [2003].

Rosenberger et al. [2004] use *hub connectivity* and the number of *short cycles* as the measures of robustness for the solution. *Hub connectivity* is the number of legs in the routings that start at one hub, end at another hub, and only visit spokes in between. If hub connectivity is low delays at one hub are less likely to affect operations at other hubs. They also observe that airlines usually do not cancel single flights but cancel a *cycle of flights* that starts and ends at the same airport. They find FAM solutions with low hub connectivity that also contain many short cycles.

A similar idea is presented in Smith and Johnson [2006]. The authors solve the fleet assignment problem and limit the number of different fleet types that can serve each airport.

Kang [2003] decompose the schedule into different sub-schedules of relatively independent flights called layers. The idea is that a delay in one layer does not affect flights in other layers.

Bian et al. [2003] find that the arrival and departure delay depends on the number of aircraft on the ground at KLM's major hub.

Listes and Dekker [2002] and Pilla [2006] consider robust fleet assignment solutions with respect to uncertainty of demand.

In a stochastic model Listes and Dekker [2002] take demand fluctuations into account. They maximise the expected profit of the fleet assignment given probabilities for the realisation of given demand scenarios. They solve the problem with a scenario aggregation approach.

Pilla [2006] propose a two-stage stochastic model. Only fleet types that can be operated by the same crew (*fleet family*) are assigned to each leg in the first stage (about 90 days prior to departure). In the second stage (2 weeks prior to departure) originally assigned fleet types of the same family can be swapped once most of the demand is realised. This procedure is called demand driven dispatch (see Berge and Hopperstad [1993]).

Robustness and Aircraft Routing

For aircraft routing problems the robustness measures focus on availability of aircraft in case previously operated flights are disrupted (Ageeva [2000], Wu [2006], and Burke et al. [2007]).

The measure of robustness in Ageeva [2000] is the number of times when two different aircraft routings *meet*. Aircraft routings meet when the aircraft of both routings are at the same airport at the same time. This permits the two aircraft to be swapped: if one aircraft is delayed the other aircraft can operate the more profitable route.

Wu [2006] consider a fixed aircraft routing solution and re-time flights within the routings to enlarge buffer times for flights that are likely to be delayed.

Burke et al. [2007] consider multiple robustness objectives of the aircraft routing problem. They maximise the number of possible aircraft swaps and minimise the probability of a flight to be delayed by varying departure times. They use a so called multi-meme memetic algorithm to solve this biobjective problem.

Robustness and Crew Pairing

For the crew pairing problem three common robustness approaches exist: minimise operational cost instead of planned cost (Schaefer et al. [2005]), minimise the number of crew changing aircraft if ground time is small (Ehrgott and Ryan [2002], Mercier et al. [2005], and Yen and Birge [2006]) and maximise the swapping opportunities for two crew similarly to aircraft swapping opportunities (Shebalov and Klabjan [2006]). It is particularly important to find robust crew solutions since a major part of variable operational cost is crew salary and airlines cannot afford many standby crews to cover flights in case of disruptions.

Schaefer et al. [2005] use expected operational cost for crew pairings instead of planned cost. Interactive effects between pairings are ignored. This assumes that a delay can only cause further delays within the same pairing. Also, a basic *push-back* strategy for recovery is used. In this strategy the flights are delayed until crew and aircraft are available. The authors use SimAir to estimate the costs and to evaluate the quality of their solutions. SimAir is a Monte Carlo simulation of airline operations that permits the evaluation of schedules and recovery policies in operations, see Rosenberger et al. [2002].

If only minimal ground time is available when crew are changing aircraft after a delayed flight, the subsequent flights operated by the crew and both aircraft will be delayed. After a few aircraft changes many flights may be delayed by the initially minor delay. Ehrgott and Ryan [2002] and Yen and Birge [2006] therefore penalise crew changing aircraft in the objective function whenever the ground time is small. Yen and Birge [2006] formulate the crew pairing problem as a stochastic programming problem in a computationally expensive approach. Ehrgott and Ryan [2002] propose a deterministic approach. Crew pairings are penalised where crew are changing aircraft and the sit-time of the crew is less than the minimal sit-time plus the expected delay of the incoming flight. Crew who stay on the same aircraft are not penalised. Thus, crew connections where disruptions are likely to propagate onto multiple flights are penalised. Robustness is treated as a second objective function in a bicriteria approach. A similar measure of robustness is used in the integrated aircraft routing and crew pairing approach by Mercier et al. [2005].

Similarly to the aircraft swapping measure in Ageeva [2000], Shebalov and Klabjan [2006] solve the crew pairing problem first and then maximise the number of *move-up* crews without increasing the planned cost too much. *Move-up* crews are crews that can potentially be swapped in case one crew is delayed. They compare their method with the method of solving the standard crew pairing problem by simulating disruptions. They find that their improved solutions incur significantly lower operational costs if the additional cost allowed for move-up crews is not too high.

For many of these robustness measures, the performance of a more robust schedule is evaluated by means of simulation and compared to traditional schedules. Often, significant reductions in recovery costs and an increase in OTP are indicated by the simulation results. Since simulations usually use simplified recovery methods the final test of the robustness of a solution, that is assumed to be more robust, will always occur once it is operating in practice.

2.4 Overview of Solution Approaches for Airline Scheduling Problems

Table 2.1 summarises main characteristics of the most relevant solution approaches for airline scheduling problems. Next to the reference, the problems considered in the approach are indicated by “S” for schedule design, “F” for fleet assignment, “A” for aircraft routing, “P” for crew pairing, “C” for crew rostering and “R” for robustness. The problem types are classified into short, medium, and long-haul and by daily, weekly, cyclic, and dated. The network types are classified into flight and duty period networks and connection and time-line networks. Indications of model and solution approach used are given. Unless specified otherwise, all approaches use the simplex method to solve the underlying linear program. The final column describes the largest problem size that was solved successfully by the approach. Similar tables can be found in Mercier [2006].

Reference	S	F	A	P	C	R	Problem Type	Network Type	Model	Solution Approach	Problem Size
Lavoie et al. [1988]				×			dated, medium,long-haul	duty period	Set covering problem	Column generation, heuristic IP	329 flights
Feo and Bard [1989]			×				dated week		MCF	Euler tour, heuristic	154 aircraft, 76 ports
Daskin and Panayotopoulos [1989]			×				single hub		Set packing	Lagrange, heuristic	
Anbil et al. [1992]				×			daily	flight	SPP	Heuristic column generation and branching	800 flights
Graves et al. [1993]				×			short-haul, daily	flight	SPP	Heuristics, cutting planes, elastic programming, heuristic column generation	1700 flights
Hoffman and Padberg [1993]				×				flight	SPP	Branch-and-cut	800 flights, 3 bases
Barnhart et al. [1994]				×			long-haul, dated month	time-line, connection	Set covering problem	Branch-and-price	833 flights, 41 ports, 2 crew bases
Hane et al. [1995]		×					short-haul, daily	time-line	FAM MCF	Simplex, heuristic branching	2500 flights, 150 ports, 11 aircraft types
Clarke et al. [1996]		×	×	×			short-haul, daily	time-line	FAM MCF	Heuristic branching	2572 flights, 11 aircraft types
Desaulniers et al. [1997b]	×	×	×				medium-haul, daily	connection	SPP, MCF	Column generation, Dantzig-Wolfe, heuristic branching	383 flights, 33 ports, 91 aircraft, 9 aircraft types
Vance et al. [1997b]				×			daily, short-haul	duty period	SPP, decomposition of flights and duty periods	Dantzig-Wolfe, column generation	174 flights
Rushmeier and Kontogiorgis [1997]		×	×	×			daily	connection	MCF	Heuristic branching	1620 flights, 8 aircraft types, 100 ports, 4 hubs

Reference	S	F	A	P	C	R	Problem Type	Network Type	Model	Solution Approach	Problem Size
Desaulniers et al. [1997a]				×			daily, medium-haul	duty period	MCF	Dantzig-Wolfe, column generation, heuristic branching	1157 flights, 63 ports, 2 crew bases
Clarke et al. [1997]			×				daily	time-line, aggregation	Asymmetric traveling salesman	Lagrange	3818 arcs, 1095 nodes
Chu et al. [1997]				×			daily, short-haul	flight	SPP	Column generation, heuristic branching	1200 flights
Barnhart and Shenoi [1998]				×			long-haul	time-line, duty period	Approximate MCF	Column generation, heuristic branching	875 flights, 2 crew bases
Barnhart et al. [1998c]		×		×			long-haul	time-line, point-to-point	MCF FAM, approximated CP	Column generation	964 flights, 2 aircraft types
Gopalan and Talluri [1998a]			×				daily, cyclic	connection	daily flight sequences	Euler tour, heuristic	12 aircraft, 33 flights
Abara [1989]		×	×				short-haul	connection	FAM MCF	Simplex	2300 flights, 150 ports, 500 aircraft, 10 aircraft types
Andersson et al. [1998]				×			-	connection	SPP	Wedelin, column generation, depth first pairing enumeration, rule language	10000 flights
Gamache and Soumis [1998]					×		two weeks	pairing based	SPP	Branch-and-price	111 pairings, 22 crew members
Barnhart et al. [1998a]		×	×				long-haul, weekly	time-line, duty period	string based SPP	Branch-and-price	1124 flights, 89 aircraft, 9 aircraft types, 40 ports
Ioachim et al. [1999]	×	×	×				weekly, long-haul	flight, connection	MCF, schedule synchronization constraints	Dantzig-Wolfe, column generation, BnB	106 flights, 1 aircraft type
Rexing et al. [2000]	×	×					daily	time-line	MCF	Branch-and-bound, iterative addition of arcs	2037 flights, 11 aircraft types

Reference	S	F	A	P	C	R	Problem Type	Network Type	Model	Solution Approach	Problem Size
Klabjan et al. [2001a]				×			daily, weekly	time-line, duty period	SPP	(Random) column generation, heuristic branching	654 flights
Cordeau et al. [2001]			×	×			dated, short-haul	connection	integrated SPP	Benders, column generation, heuristic branching	525 flights
Barnhart et al. [2002a]		×					daily	time-line, itinerary based	MCF	Column generation, row generation, heuristic branching	2044 flights, 9 aircraft types
Cohn and Barnhart [2003]			×	×				connection	SPP, aircraft routing solution variables	Column generation, BnP, cut generation	125 flights
Ehrgott and Ryan [2002]				×		×	dated week, short haul	flight, connection	biobjective SPP	Column generation, branch-and-price	750 flights
Klabjan et al. [2002]	×		×	×			daily, short-haul		crew SPP with plane count constraints	Column generation, relaxed feasibility, BnB	450 flights, 5 crew bases
Sriram and Haghani [2003]			×				weekly, short-haul	itinerary based	MCF	Local search	58 flights, 75 ports
Rosenberger et al. [2004]		×	×			×	daily	time-line	MCF		2558 flights, 9 aircraft types, 8 hubs
Lohatepanont and Barnhart [2004]	×	×					daily	time-line, itinerary based	MCF, optional flight legs	Column generation, row generation, heuristic branching	848 flights, 166 aircraft, 4 aircraft types
Schaefer et al. [2005]			×			×	daily		SPP, expected cost	(Random) column generation, heuristic branching	342 flights
Mercier et al. [2005]			×	×		×	daily	flight, connection	SPP	Benders, column generation, heuristic branching, pareto optimal cuts	707 flights, 143 aircraft
Grönkvist [2006]			×				dated	connection	SPP	Column generation, constraint programming, heuristic branching	2378 activities, 17 aircraft

Reference	S	F	A	P	C	R	Problem Type	Network Type	Model	Solution Approach	Problem Size
Shebalov and Klabjan [2006]				×		×	daily	flight, connection	SPP, move-up crew count	Lagrange, column generation	228 flights, 5 crew bases
Yen and Birge [2006]				×		×	daily, short haul	flight, connection	SPP, 2 stage stochastic program	Heuristic branching	79 flights, 11 aircraft
Sarac et al. [2006]			×				daily	connection	SPP	Branch-and-price	175 flights, 32 aircraft, 19 ports
Sandhu and Klabjan [2007]		×	×	×			daily	time-line, duty period, flight	FAM MCF, plane-count, crew constraints	Lagrange, column generation, Benders	942 flights, 4 aircraft types
Burke et al. [2007]			×			×	weekly	time-line, flight	biobjective MCF	Genetic algorithm, local search	500 activities
Mercier and Soumis [2007]	×		×	×			daily	flight, connection	SPP	Benders, column generation, cut generation, heuristic branching	523 flights
Papadakos [2007]		×	×	×			medium, long-haul, daily	flight, connection	SPP	Benders, column generation, heuristic branching, pareto optimal cuts	705 flights, 167 aircraft, 6 aircraft types

Table 2.1: Overview of solution approaches for airline scheduling problems.

Chapter 3

Aircraft Routing Problem

In this chapter we introduce a model for the aircraft routing problem and propose two solution methods. The methods are particularly well suited to solve the type of problem we are interested in by exploiting special characteristics of the problem instances. We describe the characteristics and present details of the solution procedures. We also demonstrate the performance of the algorithms on various data sets.

3.1 Model

The *aircraft routing problem* is the problem of assigning aircraft to a given set of flights in a schedule. We assign one *(aircraft) routing* (see Section 2.1.4) to each aircraft such that each flight of the schedule is contained in exactly one routing. Each routing is subject to maintenance requirements and other flying restrictions, and the number of available aircraft is fixed. The requirements and restrictions are described in Section 3.2. In the aircraft routing problem, each particular aircraft must be assigned to one specific routing. This problem is similar to the *crew rostering problem* where a line of work is assigned to each particular crew member.

Aircraft routings can be represented as columns of a binary $(m+a)\times n^R$ matrix A^R where m is the number of flights, a the number of available aircraft, and n^R the number of possible routings. The elements $(a_{ij})^R$ of the first m rows of matrix A^R are defined as follows:

$$(a_{ij})^R = \begin{cases} 1 & \text{if flight } i \text{ is contained in routing } j \\ 0 & \text{otherwise,} \end{cases}$$

with $1 \le i \le m, 1 \le j \le n^R$. Additionally, the element $(a_{m+i,j})^R$ is defined as:

$$(a_{m+i,j})^R = \begin{cases} 1 & \text{if routing } j \text{ is operated by aircraft } i \\ 0 & \text{otherwise,} \end{cases}$$

with $1 \le i \le a, 1 \le j \le n^R$. The last a constraints are referred to as *generalised upper bound (GUB) constraints (or aircraft convexity constraints)* and ensure that each aircraft is assigned to exactly one routing. With this matrix representation the aircraft routing problem can be formulated in the following manner:

$$\begin{array}{lrcl} \text{Minimise} & (\boldsymbol{c}^R)^T \boldsymbol{x}^R & & \\ \text{subject to} & A^R \boldsymbol{x}^R & = & \mathbb{1} \\ & \boldsymbol{x}^R & \in & \{0,1\}^{n^R}. \end{array} \tag{3.1}$$

The element c_j^R of $\boldsymbol{c}^R \in \mathbb{R}^{n^R}$ is the cost associated with routing j. We consider a robustness measure as the only cost of the aircraft routings, which is described in detail in Section 3.2. The decision variable $x_j^R \in \{0, 1\}$ takes value 1 if routing j is in the solution and 0 otherwise. Since a variable x_j^R is associated with a particular column of A^R and this column represents a particular aircraft routing we use the terms variable, column, and aircraft routing interchangeably.

Formulation (3.1) is called a *rostering model* which is a special case of the set partitioning model and $\mathcal{NP}$-hard.

We solve the aircraft routing problem with column generation and branch-and-price methods (introduced in Sections 1.3 and 1.4), as the number of possible aircraft routings is very large. Firstly, the LP relaxation of problem 3.1 is solved by alternately solving a restricted master problem and a column generation subproblem. The restricted master problem is solved with the simplex method and the column generation subproblem with a resource constrained shortest path algorithm. Once the LP relaxation is solved, an optimal integer solution is obtained with a branch-and-price algorithm.

We model the schedule as a directed *flight network* where *flight arcs* represent flights and nodes represent departure or arrival of a flight. Note that for the aircraft routing problem only flight nodes are necessary. However, we use the same network for the crew pairing problem where flight arcs become necessary to model passengering flights and in-flight meals. Besides flight arcs, *connection arcs* link the arrival of one flight with the departure of another flight if the two flights can be operated consecutively by the same aircraft. In a flight network, each aircraft routing corresponds to a path. The column generation problem is solved by a resource constrained shortest path problem. Costs and rules are incorporated into the network design or as resource constraints in the shortest path algorithm. Paths with negative reduced cost that represent feasible aircraft routings are returned as columns of A^R to the restricted master problem. We refer to Section 3.3.3 for more details. Note that we can construct a flight network and calculate shortest paths for each aircraft separately. This strategy allows us to include aircraft specific restrictions into the formulation and therefore the model effectively integrates fleet assignment problem and aircraft routing problem.

3.2 Rules

An airline specifies a number of rules the aircraft routing solution must satisfy. In the following we list all rules that are applicable to the problem instances of Air New Zealand and explain how they are implemented in the simplex algorithm and the resource constrained shortest path algorithm:

- AircraftCount
  The schedule must be operated by a fixed number of aircraft. For each aircraft a convexity constraint is included in the set partitioning formulation to ensure that each aircraft is assigned to exactly one aircraft routing. The problem can be modified to use at most the number of available aircraft by allowing aircraft to be idle (adding aircraft routings containing no flights) or equivalently by replacing the equality signs of the convexity constraints by less than or equal signs.

- Maintenance
  All airports are classified into maintenance and non-maintenance ports. All maintenance of the aircraft is performed at maintenance ports during the night when no flights are operated. Each aircraft is maintained each night if it is located at a maintenance port, otherwise it must be maintained at a maintenance port the following night. This requirement is included as a resource constraint in the shortest path calculation of the column generator. If an aircraft visits a non-maintenance port overnight during the construction of a path, we make sure that this aircraft will visit a maintenance port the following night. We must also make sure that the aircraft is maintained late in the morning on the day before the aircraft stays at a non-maintenance port. This is necessary so that the legal limit of 36 hours between two consecutive maintenance checks is not exceeded.

  If we solve the aircraft routing problem for a single day, the network can be altered for each aircraft to include the rule. If an aircraft starts at a non-maintenance port, only arrivals at maintenance ports are feasible end nodes for the aircraft routing on that day.

  There are no capacity constraints at the maintenance ports, i.e. all aircraft that overnight at a maintenance port can be serviced during the night. Since all aircraft at maintenance ports must be maintained that night, maintenance capacity limitations must be considered during schedule construction. The fixed schedule together with the positions of the aircraft at the start of the planning horizon determine the number of aircraft at each port for each night.

  If the number of aircraft is not pre-determined by the schedule, e.g. if fleet assignment is considered together with the aircraft routing problem, maintenance capacity constraints can be added to the model as follows. Constraints are appended to (3.1) for each combination of port and night where maintenance capacity is limited. The right hand side of such a constraint is set to the number of possible maintenance checks that can be carried out at a particular port during a particular night. The entries of the columns of matrix A for the appended constraints are all 0, except for the combination of port and night where the aircraft routing that is represented by the column terminates. In this case the entry is equal to

1. Together with less-or-equal signs, the additional constraints ensure that all maintenance capacity restrictions are obeyed.

- THROUGH
  A *through connection* consists of two flights that must be operated in sequence by the same aircraft. Through connections are included in the network if there is high passenger demand for direct connections (i.e. without the necessity of changing aircraft) on certain itineraries. A list of through connections is specified by Air New Zealand. The rule is enforced by removing arcs from the column generation network that connect the incoming flight with any other successor than the specified flight for every through connection. Additionally, all connections that connect other (than the specified) predecessor flights to the outgoing flight of the connection can be removed.

- OVERWATER
  Only a subset of all aircraft can perform international sectors, namely all aircraft that have life-raft equipment on board. We refer to these aircraft as *overwater capable*. All other aircraft cannot operate international sectors. This is enforced by eliminating international sector arcs from the column generation network for these aircraft.

- MINTURNTIME
  If an aircraft is to operate two flights consecutively, a *minimal turn-time* is required between the arrival of the incoming flight and the departure of the outgoing flight. Different minimal turn-times are defined for different airports and depend on the type of incoming and outgoing flight. Compliance with the MINTURNTIME rule is also incorporated implicitly via the network construction. Incoming flights are only connected to outgoing flights if the connection time is sufficient, i.e. exceeds the minimal turn-time.

- MINTURNSEQ
  If many connections with minimal turn-time are operated in sequence by the same aircraft, it is more likely that the flights at the end of this sequence become delayed during operations. Since no buffer time is available for the aircraft to compensate for delays that occurred early in this sequence, the last flights in such a sequence will very often depart late.

We avoid the occurrence of too many consecutive connections with minimal turn-times (denoted by MinTurnSeq) within an aircraft routing so that the solution is more likely to be operationally robust (see Section 4.3 for a detailed explanation). Such a sequence ends if a connection is operated whose ground time exceeds the minimal turn-time. Since all departure and arrival times occur at 5 minute intervals, the ground time must exceed the minimal turn-time by at least 5 minutes. In the following we refer to connections with minimal turn-time as *minimal turns*. Costs are accumulated for an aircraft routing during the shortest path calculation depending on the number of consecutive minimal turns it contains (i.e. the more consecutive minimal turns, the larger the cost). These costs are minimised in the objective function of the set partitioning formulation. We choose costs of 10, 100, and 1000 for minimal turn sequences with 2, 3, and 4 consecutive turns, respectively. Sequences with more than 4 minimal turns are prohibited. This strategy is motivated by the observation that a sequence of four consecutive minimal turns leads to much higher delays than the total delay caused by two sequences with two consecutive minimal turns each. Other functions of the number of consecutive minimal turns can easily be considered within the shortest path calculation. The total cost of minimal turns of an aircraft routing is the sum of the costs of all minimal turn sequences contained in the aircraft routing.

- ThroughValues
  A *through-value* is revenue attached to a pair of flights if they are operated in sequence by the same aircraft. Through-values can be added as (negative) costs to the aircraft routings such that as many through connections as possible are operated by the aircraft. This can be incorporated into the column generation method by adding costs to connection arcs. Since for our problem instances a set of through connections that must be operated by the same aircraft is given by Air New Zealand, we do not consider through-values.

- OperationalCosts
  Operational costs that vary between aircraft can be considered by attaching different costs to the flight arcs for different aircraft. This makes it possible for example to model different fuel efficiencies among the air-

craft and assign more flying to more efficient aircraft. In the scenarios we consider, the differences between aircraft are very small. Hence, we assume that all aircraft are identical and do not impose any operational costs.

- MaintenanceLimit
  The legal requirement for the maximal time between maintenance checks is 36 hours. Instead of maintaining the aircraft each night as in current practice, each aircraft can be maintained less frequently but at least every 36 hours. However, maintaining aircraft during the day, when only limited time is available between flights, is for operational and robustness reasons not desirable. It can be achieved in the column generation process using a resource constraint and only generating paths that contain as few maintenance checks as possible but two consecutive checks within 36 hours. We do not exploit the legal maintenance limit in our algorithm and schedule maintenance checks as described in rule Maintenance.

- FlyingTime
  Another requirement could be for all aircraft routings in the solution to contain approximately the same number of flights and the same amount of flying time. Since the total number of flights and the total flying time in the schedule is known, targets for these values can be added to the shortest path calculation. The target values can be set to be the average number of flights and average flying time per aircraft, respectively. Using resource constraints in the shortest path algorithm, aircraft routings are generated with flying time and number of flights as close to the target values as possible. We do not include this rule in our implementation of the aircraft routing algorithm.

- BigCycle
  The BigCycle condition, all aircraft must operate all flights in the schedule within a certain period, is not applicable to schedules that change frequently. The BigCycle condition is equivalent to finding a single aircraft routing that contains each flight exactly once and wraps around from the end to the beginning of the planning horizon as many times as aircraft are available. If all aircraft must operate all flights in a schedule of one week then the solution needs to span as many weeks as

aircraft are available. This is not practical if the schedule already changes in the subsequent week. The nature of the schedule of the problem instances we consider enables the equal utilisation of all aircraft. Many aircraft meet at only a few airports for overnight stays. Aircraft routings of subsequent days can easily be swapped between different aircraft if necessary. For these reasons we do not include the BigCycle condition in our approach.

Additional rules are needed if the aircraft routing problem is solved as part of an integrated aircraft routing and crew pairing model. We describe these rules in Chapters 5 and 6. Note that in this section, the only costs that are associated with an aircraft routing are MinTurnSeq costs. Finding an optimal solution to the aircraft routing problem is therefore equivalent to finding an aircraft routing solution with a minimal number of consecutive minimal turn sequences. This approach is generalised in subsequent chapters.

3.3 Solution Methods

As outlined in Section 3.1, the aircraft routing problem is modelled as a set partitioning model of the rostering type and is solved with column generation and branch-and-price.

Columns of the set partitioning model correspond to paths in a flight network. In this section, we first describe a preprocessing step to reduce the size of the flight network. Secondly, the column generation and branching procedures are described in more detail. We finally present two methods to efficiently solve the problem.

3.3.1 Preprocessing

When performing column generation, we must repeatedly calculate resource constrained shortest paths in the flight network. The performance of the algorithm depends critically on the number of nodes and arcs contained in the network. Therefore a preprocessing step is performed on the flight network in order to reduce the number of arcs. Potentially, each incoming flight can be connected to all flights that leave the arrival airport at any time after the incoming flight has arrived and the minimal turn-time has elapsed, causing a large number of possible connection arcs in the network. We now show that we can reduce the number of arcs in the network significantly and still guarantee to find an optimal solution. Similar techniques are for example discussed in Grönkvist [2006].

We solve the aircraft routing problem for a given schedule with dated time horizon. We assume all arrival and departure times occur at 5 minute intervals starting from midnight. The number of available aircraft and locations of all aircraft at the beginning of the time horizon are given. Since the number of aircraft at each airport at the beginning of the time horizon is known we can determine the number of aircraft at each airport for every minute of the planning horizon. For a given airport p this is simply calculated by the following summation:

$$n_p^t = n_p^0 + a_p^t - d_p^t,$$

where n_p^t is the number of aircraft at airport p at time t (in minutes), n_p^0 the number of aircraft at airport p at time 0 (the beginning of the planning horizon), a_p^t the number of arrivals at airport p before time t, and d_p^t the number of departures at airport p before t.

When we calculate n_p^t for all times t and all airports p we can identify all times $\bar{t}$ when no aircraft is at airport $\bar{p}$, i.e. when $n_{\bar{p}}^{\bar{t}} = 0$. With this information we can greatly reduce the number of connection arcs in the network by removing connection arcs that cannot be part of any feasible solution since a flight arriving at $\bar{p}$ before $\bar{t}$ and a flight departing from $\bar{p}$ after $\bar{t}$ cannot be operated consecutively by any aircraft. Hence, we remove this connection arc from the network. The dramatic effect of this preprocessing step can be observed in Table 3.1 where the number of feasible connections for aircraft before and after preprocessing are compared.

If we assume that the minimal turn-time requirement is the only rule applicable and that the minimal turn-time is equal for all connections, we can construct a feasible solution of the aircraft routing problem with a greedy heuristic. We order all arrivals and departures at an airport by increasing time and consider one arriving flight at a time. For each arrival, the flight that is operated subsequently by the same aircraft is chosen to be the first feasible departing flight with respect to the minimal turn-time. This assignment will lead to a feasible aircraft routing solution if such a solution exists. Since most rules are relaxed it might be an infeasible solution to the original problem. It can still be useful to determine a lower bound on the number of aircraft that are required to operate the schedule. We can also use these aircraft routings as initial columns of the A^R matrix of the restricted master problem rather than just using an artificial identity matrix (see Section 1.3).

3.3.2 LP-Relaxation

The linear program of the restricted master problem is solved with ILOG CPLEX 10.1 (ILOG [2006b]) using the simplex algorithm with default parameter settings. The set partitioning model is formulated with the ILOG Concert Technology 2.3 (ILOG [2006a]) interface and CPLEX is called to solve the model. As a result, CPLEX returns the solution status (optimal or infeasible) together with the solution vector $\boldsymbol{x}$ and the dual solution vector $\boldsymbol{\pi}$. Note that because of the set partitioning formulation all variables are bounded between 0 and 1 and hence the solution can never be unbounded.

The column generation subproblem is called with $\boldsymbol{\pi}$ as input. If columns with negative reduced cost are returned these are appended to the set partitioning model and the model is re-solved.

To obtain integer solutions, we use our own branch-and-price method and only utilise CPLEX to solve the linear programs. As an advantage over the CPLEX MIP (Mixed Integer Programming) solver, we can generate columns while traversing the branch-and-bound tree and obtain optimal integer solutions.

Branching decisions are enforced in the set partitioning model by setting the upper bound of a variable to 0 if the column that is associated with the variable is infeasible with respect to the branching decisions.

3.3.3 Column Generation

Once the simplex algorithm finds an optimal solution $\boldsymbol{x}$ to the restricted master problem, a column generation subproblem is called. The column generator finds a column s with negative reduced cost $r_s = (c_s - \boldsymbol{\pi}^T \boldsymbol{a}_s)$ or guarantees that no such column exists.

The column generation problem is solved for each aircraft separately and modelled on the flight network as follows. Only flight and connection arcs that do not violate the MinTurnTime rule, the Through rule, and the OverWater rule are included in the network. During the branch-and-bound process we also do not include arcs that violate the branching decisions (see Section 3.3.4). The negative of each dual variable π_i is stored as the cost of the arc associated with flight i. Each overnight connection arc is assigned an attribute if maintenance can be performed during the stopover. The departure of every flight the aircraft can operate as a first flight of a routing is marked as a source node. The sink nodes are all arrivals of flights that the aircraft can operate as last flights of a routing. A minimal reduced cost column corresponds to a minimal cost path from a source node to a sink node. Hence, the column generation problem can be solved as a resource constrained shortest path problem which is implemented as a label setting shortest path algorithm (Algorithm 1). Since we solve the problem for a dated time horizon, the flight network does not contain any cycles and we can assume all nodes are ordered by increasing time. For the same reason it is sufficient to employ a label setting algorithm rather than a more sophisticated label correcting algorithm.

To find a shortest path satisfying resource constraints we need to loop once over all nodes, for an overview see Algorithm 1. A set of labels is attached to each node, each label representing a path from a source node to the current node. Each label contains the cost and resource usage of its path. Two resources are attached to each label, one contains the elapsed time since the last maintenance check, and the other one contains the number of minimal turns in the current MinTurnSeq sequence.

For each label l and outgoing arc a of a node i we *extend* the label to a successor label l' at the successor node i' (see Step 8 of Algorithm 1): arc a is appended to the path represented by label l to form a new path represented by l' and the cost of a is added to the cost of l to form the cost of l'. We also update the information of the MinTurnSeq rule. If the arc represents a minimal turn we increase the resource that counts the length of the current MinTurnSeq sequence by 1. If the connection arc exceeds the minimal turn-time the cost for the previous MinTurnSeq sequence is added to the cost of the label and

Algorithm 1 Label Setting - Resource Constrained Shortest Path

```
1: INPUT: A flight network for each aircraft that respects rules and branching
   decisions for this aircraft.
2: for each aircraft do
3:   Initialise label at source nodes with default values for cost and all re-
     sources
4:   for each node $i$ {ordered by departure time} do
5:     for each label $l$ at node $i$ do
6:       for each outgoing arc $a$ connecting $i$ to $i'$ do
7:         Extend label $l$ along arc $a$ to $l'$.
8:         if $l'$ dominates label(s) $\bar{l}$ of $i'$ then
9:           Discard label(s) $\bar{l}$.
10:          Save $l'$ as label of $i'$.
11:        else if Some label $\bar{l}$ of $i'$ dominates $l'$ then
12:          Discard label $l'$.
13:        end if
14:      end for
15:    end for
16:  end for
17:  Store all labels at end nodes as aircraft routings
18: end for
19: OUTPUT: A set of negative reduced cost aircraft routings for each aircraft
    (possibly empty).
```

the resource is set to 0. When we save a label as a path at a sink node, the cost for the current MinTurnSeq sequence is added to the cost of the path. The reduced cost of the path is equal to the sum over all costs of all arcs contained in the path. If maintenance is performed during the ground time represented by the connection arc, the Maintenance resource is updated accordingly.

The new label l' is deleted if any rule is violated, otherwise it is checked for *dominance*. A label l *dominates* another label $\bar{l}$ at the same node if all resources (including cost) used at label l are better than or equal to the corresponding resources used at label $\bar{l}$. In the case of the aircraft routing problem l dominates $\bar{l}$, if l incurs less (or equal) cost than $\bar{l}$, the last maintenance check in l was at the same time or later than the last one in $\bar{l}$, and the current number of consecutive minimal turns of l is at most that of $\bar{l}$. Only non-dominated labels are kept at each node since such labels always extend to a path as least as good as a dominated label. When extending a label it may be dominated by a label already present at the next node, in which case the new label is deleted.

Otherwise it is added to the list of labels at the new node. If it dominates one or more other labels at that node these labels are removed. Once the labelling algorithm is finished, all labels at sink nodes represent feasible aircraft routings with minimal reduced cost.

Additional resources can be added to the labels to model operational costs, to enforce a similar amount of flying time between aircraft routings, or to maximise the time between consecutive maintenance checks.

The column generation subproblem returns the aircraft routing with the most negative reduced cost for each aircraft to the restricted master problem or guarantees that no such aircraft routing exists. Hence, for each aircraft at most one aircraft routing is added to the set partitioning model and the restricted master problem is solved again. If no negative reduced cost aircraft routing exists for any aircraft, optimality (or infeasibility) of the LP relaxation of the original problem is guaranteed and the algorithm stops.

3.3.4 Branch-and-Price

After the LP relaxation is solved we start the branch-and-price process if any value x_f of the solution $\boldsymbol{x}$ is fractional. We branch on aircraft-flight pairs which is a special form of constraint branching proposed by Ryan and Foster [1981]. In one branch a particular aircraft is forced to operate a particular flight and in the other branch the aircraft is not allowed to operate that flight. Such a branching rule is much better suited for this kind of set partitioning problem than variable branching. In variable branching the aircraft routing associated with a fractional variable is forced to be in the solution or not. This leads to a very imbalanced branch-and-bound tree since forcing an aircraft routing to be in the solution restricts the feasible solution space significantly; banning an aircraft routing from the feasible solution space does not restrict the feasible solution space significantly since a large number of very similar aircraft routings may exist. An even more severe problem of variable branching is the feasibility check of aircraft routings within the column generation subproblem. If an aircraft routing is banned from the solution space it is difficult and computationally expensive to prevent this aircraft routing being generated again. Only after the whole shortest path is generated, can it be checked if this path is banned. Using constraint branching on the other hand, the feasibility of paths can easily be incorporated into the network design (see below).

To decide which aircraft-flight pair to branch on, for each flight and each aircraft we sum up all fractional values of variables associated with aircraft routings that cover this flight and are operated by the aircraft. We usually branch on the aircraft-flight pair with the highest fraction less than 1. We branch on the smallest fraction only if the highest fraction is significantly smaller than 0.5. Since forcing a particular aircraft to operate a flight is much more restrictive than forbidding it from operating a flight, we choose the first option as much as possible, i.e. we execute depth-first-one-branching. In other words, after one branch-and-bound node is solved, the next node to be considered is the most restrictive child node of the current node. Only if this decision leads to infeasibility, other nodes are considered.

At each node of the branch-and-bound tree the LP relaxation of problem (3.1) is solved again. The branching decisions we make at a node are incorporated into the simplex algorithm by setting the upper bound of variables to 0 if the associated aircraft routings are banned by the branching decisions.

Since we solve the column generation problem for each aircraft separately, the branching decisions are included in the flight network (for a particular aircraft) as follows. If the aircraft is forced to operate a particular flight, all connection and flight arcs that overlap in time with the forced flight arc are removed from the network for that aircraft. In fact, other arcs may be removed whenever the usage of the arc contradicts the usage of the forced arc, i.e. an arc can be removed if there exists no path between the arc and the forced arc. For all other aircraft the flight arc is removed from the network. If the aircraft is forced not to include a particular flight, this flight is removed from the network for this aircraft. In this case the networks for other aircraft remain unchanged.

To justify the strategy of branching on aircraft-flight pairs we refer to the theory of *perfect matrices* which was first proposed by Padberg [1974]. For a perfect matrix A and the problem $\min\{\boldsymbol{c}^T\boldsymbol{x} : A\boldsymbol{x} = \mathbb{1}, \boldsymbol{x} \geq 0, \boldsymbol{c} \in \mathbb{Z}^n\}$ there always exists an optimal integral solution vector $\boldsymbol{x}$. We first introduce some notation.

A graph G is called *complete* if every node is adjacent to every other node.

The *chromatic number* of G is the minimal number of different colours needed to colour all nodes of G such that no adjacent nodes have the same colour. A *subgraph* G' of G is a subset of the nodes of G together with all arcs of G linking nodes in the subset. A *clique* is a complete subgraph. A graph G is called *perfect* if for every subgraph G' of G the chromatic number of G' is equal to the maximal cardinality of a clique in G'.

Let G_I denote the *intersection graph* associated with a matrix A. The nodes of G_I correspond to columns of A and two nodes are linked by an edge if the two corresponding columns have a common 1 in any row. The rows of A must contain all cliques that are contained graph G_I. The matrix A is called perfect if the associated intersection graph is perfect.

We investigate the submatrices of A^R that consist of the columns of A^R associated with a single aircraft. The intersection graphs of these submatrices are complete since all columns have a common 1 in the aircraft convexity constraint. Every subgraph of a complete graph is also complete. Also, in any complete graph the chromatic number equals the cardinality of a maximal clique which is equal to the number of nodes in the graph. This results in the following theorem.

Theorem 3.3.1 *Each submatrix of A^R, that consists of the columns associated with a single aircraft, is perfect.*

Hence, not many fractions occur in the solutions of the LP relaxation of the aircraft routing problem. In particular, if there is only one aircraft then the solution of the LP relaxation is guaranteed to be integer. Intuitively, in this simple case the convexity constraint dominates all other constraints and hence all other constraints can be removed from the formulation. This will lead to only a single positive variable in the basis which is integer because of the right hand side of the convexity constraint being equal to one. The observation of Theorem 3.3.1 has a large impact on a wide range of problems that can be formulated as set partitioning problems with a convexity constraint, enabling this class of problems to be solved easily. The theorem has an even bigger impact on the class of rostering problems, e.g. staff rostering. This type of problem not only occurs in airline scheduling but in a wide range of industries, e.g. health care or public transport. This type of problem can be very large, depending on the number of staff and the duration of the time period that needs to be solved. Methods resulting from the theorem enable to solve such large scale problems to integer optimality. The fractions in the LP solution are caused by different aircraft/staff members competing for the same flight/piece of work. Such a fraction can be easily removed by applying a constraint branching strategy (see above), effectively assigning the flight/piece of work to one of the two aircraft/staff members. We refer to Ernst et al. [2004b] for more details on rostering problems and Gamache and Soumis [1998], Butchers et al. [2001], and Kohl and Karisch [2004] for successful solution methods.

3.3.5 Alternative Set Partitioning Formulation

In the case when all aircraft are identical, solving Formulation (3.1) can be difficult due to symmetry in the model. For all aircraft that start at the same port, identical columns are generated by the column generation subproblem. This can result in many equivalent columns being present in the matrix that only differ in the aircraft convexity constraint coefficient. Many equivalent columns cause degeneracy of the model and can make it very difficult to solve. To prevent the construction of equivalent columns, the aircraft convexity constraints can be substituted by one equality constraint for each starting port. The right hand side of this constraint is set to the number of aircraft starting at the port. The rest of the model remains unchanged. The column generation subproblem is only called for each starting port instead of for each aircraft. This alternative formulation removes the symmetry from the original model. The approach has the slight drawback that we cannot longer use aircraft-flight pair constraint branching as described in the previous section but only follow-on constraint branching to obtain integer solutions. We do not use this model for our computational experiments since the number of identical aircraft starting from the same port is usually very small, i.e. less than four.

3.3.6 Decomposition Methods

All scenarios that are addressed in this thesis are dated schedules. We solve a dated problem because the flights contained in the schedule vary significantly from day to day. A daily solution (Section 2.1.1) can therefore not be duplicated to cover a multiple day period. The size of the set partitioning formulation is large if a multiple day period such as a week is considered. Also, because many aircraft are staying overnight in only a few different airports and all these aircraft can operate many of the flights departing on the next morning, the number of possible overnight arcs is large. For these reasons, long computation times are needed to solve the LP as well as the resource constrained shortest path subproblems. In this section we describe two methods that decompose the fully dated formulation in order to guarantee fast solution times.

Sequential Method

The large number of overnight arcs causes a large number of feasible aircraft routings. These routings consist of relatively few different routings for each day that are combined with different overnight arcs. Many solutions with identical objective values exist, only differing by how the daily routings are joined together. Since it is sufficient to find one of these solutions we investigate how to decompose the problem by solving smaller time periods at a time.

We first consider the aircraft routing problem without any cost and the only rules that apply are the MinTurnTime rule and the Through rule which can both be included in the network design. We assume all flights are operated during the day and all aircraft are grounded overnight at some airport which is the case for the problem instances we consider. When solving this version of the aircraft routing problem for a dated period we observe that there is no interaction of aircraft routings between different days and hence each day of the period can be solved separately. An aircraft routing solution for the whole period can be constructed by concatenating the aircraft routings that span a day. This *sequential solution method* speeds up the the solution process considerably.

When we include the OverWater rule, the strategy of solving one day at a time can lead to infeasibility. This is the case for example, whenever aircraft that cannot fly to international destinations, are staying overnight at an airport from which only international flights depart on the next day.

It is easy to see that there is no interdependence between different days if the MinTurnTime, Through, Maintenance, and MinTurnSeq rules are applied and if all aircraft are identical (no OverWater rule). The aircraft maintenance requirements for an aircraft during a day depend only on the type of airport where the aircraft stayed during the night before, i.e. if the airport where the aircraft stayed overnight is a maintenance base or not. Also, it can safely be assumed that overnight connections are always longer than the minimal turn-time and hence no sequence of minimal turn-time connections can span multiple days.

When all rules are considered, solving the sequential method with one day at a time can lead to infeasible or suboptimal solutions with respect to the MinTurnSeq rule. This is caused for example by overwater capable aircraft that must do many consecutive minimal turns in the morning in order to reach the origin of international sectors.

We therefore modify the sequential strategy by solving a subperiod of x days with $1 < x < d$ and d the number of days in the whole period. We then shift the subperiod by y days ($y \leq x$ and $x + y \leq d$) and solve the next subperiod from day y to day $y + x$. A similar technique is used successfully to solve the crew rostering problem in Day and Ryan [1997]. Parameters x and y can be chosen depending on the rules applicable from solving subperiods of one day ($x = 1$) to solving the whole period at once ($x = d$). By solving overlapping subperiods ($y < x$), we can guarantee that a feasible extension of the solution of the previous subperiod exists. We choose the values $x = 2$ and $y = 1$ for the best compromise between running time and solution quality. When using an overlap of $y = 0$ days the OverWater rule causes infeasible solutions. Solving subperiods of more then $x = 2$ days can cause very long running times.

Flow Formulation Method

In this section we present a flow formulation for the dated period problem. This model guarantees global optimality of the solution but only needs to generate aircraft routings for single days. This approach features very fast solution times of the resource constrained shortest path calculations and leads to an optimal result in contrast to the approach of the previous section.

Again, we observe that the only interdependence between aircraft routings on different days is caused by the OverWater rule. This effectively separates the aircraft into two different fleets: one fleet can operate international sectors while the other fleet cannot operate international sectors. We can partition the schedule of the whole period by aircraft routings that only span a single day if we make sure that the correct number of aircraft of the overwater capable fleet is available on each day to operate the aircraft routings containing international sectors. After the problem is solved, an aircraft routing that spans the whole period is defined by the aircraft routings that span a day each and are operated by the same aircraft. This results in the following model.

We decompose the set partitioning (SPP) model (3.1) for the whole period into one SPP model for each day coupled by flow conservation constraints for the aircraft that are overwater capable. Since there are only two types of aircraft this condition preserves the flow of the other fleet as well. The flow conservation constraints only link two set partitioning problems on consecutive days. The constraints ensure that the number of aircraft that are permitted to fly over water and that end their routing at a specific airport at a specific day is equal to the number of overwater capable aircraft that leave from that airport the next morning.

Formulation (3.2) describes the altered set partitioning model. Matrix A^R is split into matrices A^R_d for each day d. Matrix A^R_d contains aircraft convexity constraints for each aircraft and flight covering constraints for the flights that are departing on day d. Similarly, costs $\boldsymbol{c}^R$ and decision variables $\boldsymbol{x}^R$ are split for each day. Flow conservation constraints are added to the model as follows. We define binary $m_p \times n^R_d$ matrices P^A_d and P^D_d where m_p is the number of airports and n^R_d the number of possible routings on day d. Elements $(p_{ij})^A_d$ of matrix P^A_d are defined as follows:

$$(p_{ij})^A_d = \begin{cases} 1 & \text{if routing } j \text{ is assigned to an overwater aircraft and} \\ & \text{routing } j \text{ ends at port } i \text{ on day } d \\ 0 & \text{otherwise,} \end{cases}$$

with $1 \leq i \leq m_p, 1 \leq j \leq n^R_d$. Similarly, elements $(p_{ij})^D_d$ of matrix P^D_d are defined as:

$$(p_{ij})_d^D = \begin{cases} 1 & \text{if routing } j \text{ is assigned to an overwater aircraft and} \\ & \text{routing } j \text{ starts at port } i \text{ on day } d \\ 0 & \text{otherwise,} \end{cases}$$

with $1 \leq i \leq m_p, 1 \leq j \leq n_d^R$. With this matrix representation the *flow formulation* of the aircraft routing problem can be represented as follows:

$$\begin{array}{llcccccccl} \text{Minimise} & (\boldsymbol{c}_1^R)^T \boldsymbol{x}_1^R & + & (\boldsymbol{c}_2^R)^T \boldsymbol{x}_2^R & + & (\boldsymbol{c}_3^R)^T \boldsymbol{x}_3^R & \dots & & \\ \text{subject to} & A_1^R \boldsymbol{x}_1^R & & & & & & = & \mathbb{1} \\ & & & A_2^R \boldsymbol{x}_2^R & & & & = & \mathbb{1} \\ & & & & & A_3^R \boldsymbol{x}_3^R & & = & \mathbb{1} \\ & & & & & & \ddots & & \vdots \\ & P_1^A \boldsymbol{x}_1^R & - & P_2^D \boldsymbol{x}_2^R & & & & = & 0 \\ & & & P_2^A \boldsymbol{x}_2^R & - & P_3^D \boldsymbol{x}_3^R & & = & 0 \\ & & & & & & \ddots & & \vdots \end{array} \tag{3.2}$$

All decision variables $\boldsymbol{x}_d^R \in \{0,1\}^{n_d}$ are binary variables. The decision variable $x_{d\,j}^R \in \{0,1\}$ takes value 1 if routing j is in the solution on day d and 0 otherwise. The second set of constraints ensures that the number of overwater capable aircraft that arrive on a day and stay overnight is equal to the number of overwater capable aircraft departing in the next morning.

This enhanced SPP formulation contains flight covering constraints for the whole period and is again solved with CPLEX, but aircraft routings that only span a day are generated for each day independently. The negative of the dual values of the associated flow conservation constraints must be added to the reduced costs of the aircraft routings during the shortest path calculation.

Not many (number of airports times (number of days minus 1)) flow conservation constraints have to be added to the original set partitioning formulation (3.1). We also expect most of these constraints to be easily satisfied during the solution process. The great advantage of the formulation is the speed-up in the column generation process since resource constrained shortest paths are generated on single day networks only. We also expect that significantly fewer columns need to be generated during the solution process than in the original problem.

During the LP solution phase, columns can be generated for each aircraft separately or once for overwater capable aircraft and once for all other aircraft. In the first case the start nodes represent only flights departing from the airport where the aircraft is located. In the latter case the start nodes are all possible first flights for all aircraft of the same type. In both cases we can solve the resource constrained shortest path algorithm for each start port separately. Maintenance conditions of the airport where the aircraft routings end can then be integrated into the network because the conditions only depend on the starting port. Since the second option incurs fewer calculations of resource constrained shortest paths, we solve the column generation once for each aircraft type during the LP solution phase. During the branching process we enforce the branching decisions by the network design and hence solve the resource constrained shortest path problem for each individual aircraft.

The model can easily be modified by decomposing the aircraft into more than two types depending on properties of the aircraft. The model can then be used to solve the integrated fleet assignment and aircraft routing problem. If all aircraft have different feasibility parameters, one needs to add flow conservation constraints for each aircraft. It remains to be checked how this affects the solution times which depend on how many of these constraints are difficult to enforce during the solution process. If the scheduling period is very long the number of rows can become very large and the simplex algorithm cannot solve the model efficiently any longer. In this case the matrix structure implies a Dantzig-Wolfe decomposition approach as the most natural solution method. The master problem only contains the flow conservation constraints and one aircraft routing subproblem must be solved for each day of the period.

3.4 Computational Experiments

We perform computational experiments on point-to-point flight networks corresponding to domestic airline schedules of Air New Zealand. The schedules mostly contain short-haul flights and a small number of medium-haul international flights. The schedules vary on a daily basis and we consider dated time periods of one day, three days, and one week. We consider four different schedules: summer 2005 (s05), winter 2005 (w05), summer 2006 (s06), and winter 2007 (w07). For the last schedule (w07) we additionally solve periods of 10 and 14 days. The schedule in the second week differs from the schedule in the first week for this scenario and the large schedules yield an indication of the scalability of the approaches to solve larger instances. For each schedule we compute solutions to the aircraft routing problem with three different approaches presented in the previous section: we investigate the performance of the sequential approach with a subperiod length of one day (seq1), the sequential approach with a subperiod length of two days and an overlap of subperiods of one day (seq2), and the performance of the flow method (flow).

The fleet consists of 14 aircraft for all scenarios, four of which can operate international sectors. The flight networks contain up to 750 flights and 3000 connection arcs per week. More details on the characteristics are given in Table 3.1. The table shows the scenario names, the number of available aircraft and the number of overwater capable ones. We also list the number of feasible aircraft connections before and after preprocessing.

scenario	aircraft (overwater)	flights	aircraft connections before preprocessing	aircraft connections after preprocessing
s05, 1 day	14 (4)	113	8678	170
s05, 3 days	14 (4)	330	17605	734
s05, 7 days	14 (4)	743	33191	2170
w05, 1 day	14 (4)	114	9312	188
w05, 3 days	14 (4)	336	18745	822
w05, 7 days	14 (4)	753	34391	2212
s06, 1 day	14 (4)	108	8930	210
s06, 3 days	14 (4)	324	17980	877
s06, 7 days	14 (4)	745	33789	2298
w07, 1 day	14 (4)	110	8993	177
w07, 3 days	14 (4)	330	18227	1057
w07, 7 days	14 (4)	751	34303	3091
w07, 10 days	14 (4)	1092	48570	4063
w07, 14 days	14 (4)	1510	64569	6467

Table 3.1. Characteristics of scenarios.

Tables 3.2 - 3.5 list results for the different schedules and solution methods. In the first two columns, scenarios (i.e. schedules and the number of days of the scheduling periods) and solution methods are listed. In some instances of seq1, the OverWater rule needs to be relaxed in order to find feasible solutions. For these scenarios all aircraft may operate international sectors and we denote the scenarios by "seq1r". The numbers of rows and columns refer to the number of rows in the constraint matrix and the total number of generated columns. For sequential approaches these numbers are maximal values over all subperiods. To obtain an estimate of the total number of columns that are generated and the number of branch-and-bound nodes evaluated, the values need to be multiplied by the number of subperiods that are solved in the scenario.

In the column "minimal turns" the number of consecutive minimal turns is listed for the whole solution period. The values (t_2, t_3, t_4) represent the occurrences of minimal turn sequences of length 2, 3, and 4, respectively. Since the penalty for 5 consecutive minimal turns is very large, such a sequence does not occur in any of the solutions. The number of branch-and-bound nodes to obtain an optimal integer solution is shown in column "BnB-nodes". For sequential approaches this is again a maximum over all subperiods. Finally, computation times to solve the whole period of the scenarios are listed. The total running times ("tot"), the LP solution times ("lp"), the branch-and-bound solution times ("ip"), and the total times used to generate columns ("colgen") are listed separately. The total running time includes setup and preprocessing times besides LP and IP solution times. The LP and IP solution times include the time used for column generation. All times are given in seconds. Note that we generate columns during the branch-and-bound process. The LP/IP gap is 0 for all scenarios shown, i.e. for all scenarios the objective values of the optimal LP and integer solutions coincide. Note that the only costs considered in the objective function are penalties for minimal turn sequences.

The results are similar for all schedules. For all scenarios of 7 day duration, the number of minimal turn sequences of length two is between 67-80 and between 1-12 for sequences of length three. Hardly ever four consecutive minimal turns need to be included in the solution. Also, run times, number of rows, columns, and branch-and-bound nodes are similar for all scenarios considered. We observe that the seq1 method is very fast but not always able to generate feasible solutions in which case we relax the OverWater rule. As expected (see Section 3.3.4), many of the LP solutions are integer or almost integer causing the number of branch-and-bound nodes to be small and IP solution times very short.

The seq2 method takes significantly longer than seq1 and many more aircraft routings are generated over two days than for a single day. The seq2 method always finds a feasible solution, and in all but 2 instances (10 and 14 day periods of schedule w07, Table 3.5) the solution is optimal with respect to the number of minimal turn sequences. The number of branch-and-bound nodes required is also significantly larger than for method seq1. From the increase in run time between methods seq1 and seq2 we conclude that solving subperiods of more days than two is impractical.

The flow method is faster than the seq2 method and generates fewer columns than the seq2 method for a single subperiod. Even when we solve a two week period with the flow method the number of aircraft routings generated is smaller than the number of aircraft routings generated for a two day period with seq2. The number of branch-and-bound nodes is larger than in the other approaches since the row dimension of the matrix is much larger. Compared with the total number of nodes required in the sequential approaches (number of subperiods $\times$ number of BnB-nodes) the number of nodes is still small. Branching on multiple days simultaneously can be investigated to speed up the solution process without deteriorating solution quality. Since the flow method guarantees to find the optimal solution the superiority of this approach is obvious. We utilise this method for all computational experiments in subsequent chapters.

				minimal	BnB-	times			
scenario	method	rows*	cols*	turns	nodes*	tot	lp	ip	colgen
s05, 1	seq 1	127	637	(11,3,0)	6	0.09	0.03	0.04	0.02
s05, 3	seq 1	127	821	(33,5,0)	6	0.19	0.14	0.05	0.06
s05, 7	seq 1r	132	2970	(67,10,3)	20	1.12	0.51	0.58	0.36
s05, 3	seq 2	233	10046	(33,5,0)	19	4.70	2.06	2.60	1.58
s05, 7	seq 2	247	23095	(67,10,3)	39	30.36	8.34	21.84	7.23
s05, 1	flow	127	1014	(11,3,0)	7	0.27	0.13	0.14	0.11
s05, 3	flow	398	3980	(33,5,0)	5	1.56	1.07	0.47	0.39
s05, 7	flow	919	12834	(67,10,3)	59	24.77	4.79	19.96	8.21

* for sequential approaches the maximal number over all iterations is listed.

Table 3.2. Computational results for summer 2005.

				minimal	BnB-	times			
scenario	method	rows*	cols*	turns	nodes*	tot	lp	ip	colgen
w05, 1	seq 1	128	1098	(12,2,0)	10	0.15	0.08	0.07	0.07
w05, 3	seq 1	129	2272	(35,5,0)	10	0.60	0.27	0.32	0.27
w05, 7	seq 1	133	3163	(71,12,1)	20	1.61	0.70	0.90	0.62
w05, 3	seq 2	236	25117	(35,5,0)	26	17.35	5.88	11.32	4.24
w05, 7	seq 2	249	25117	(71,12,1)	43	42.98	13.46	29.16	10.46
w05, 1	flow	128	1356	(12,2,0)	1	0.21	0.20	0.01	0.04
w05, 3	flow	404	5241	(35,5,0)	21	4.09	1.62	2.47	1.54
w05, 7	flow	929	14171	(71,12,1)	36	22.99	7.24	15.71	7.05

* for sequential approaches the maximal number over all iterations is listed.

Table 3.3. Computational results for winter 2005.

				minimal	BnB-	times			
scenario	method	rows*	cols*	turns	nodes*	tot	lp	ip	colgen
s06, 1	seq 1	122	1800	(13,0,0)	1	0.11	0.10	0.00	0.02
s06, 3	seq 1r	126	2430	(41,0,0)	13	0.59	0.38	0.21	0.24
s06, 7	seq 1r	134	2651	(80,1,0)	14	1.65	0.86	0.76	0.49
s06, 3	seq 2	230	33177	(41,0,0)	30	26.13	6.91	19.01	4.99
s06, 7	seq 2	247	34168	(80,1,0)	33	61.14	17.07	43.51	14.41
s06, 1	flow	122	2198	(13,0,0)	1	0.42	0.40	0.01	0.19
s06, 3	flow	392	8282	(41,0,0)	26	7.26	2.78	4.46	2.40
s06, 7	flow	921	18455	(80,1,0)	75	43.65	8.88	34.71	12.70

* for sequential approaches the maximal number over all iterations is listed.

Table 3.4. Computational results for summer 2006.

				minimal	BnB-	times			
scenario	method	ro.*	cols*	turns	no.*	tot	lp	ip	colgen
w07, 1	seq 1	124	1296	(12,1,0)	6	0.13	0.08	0.02	0.06
w07, 3	seq 1	129	4304	(25,4,0)	11	0.93	0.54	0.37	0.30
w07, 7	seq 1r	135	6647	(60,6,0)	18	2.82	1.62	0.50	0.74
w07, 10	seq 1r	135	6647	(93,8,0)	18	4.77	2.56	2.16	1.53
w07, 14	seq 1r	135	6647	(127,9,0)	21	6.55	3.53	2.94	1.76
w07, 3	seq 2	234	53107	(25,4,0)	26	28.29	11.69	16.25	5.13
w07, 7	seq 2	252	69792	(60,6,0)	33	98.77	46.46	51.17	21.36
w07, 10	seq 2	252	69792	(92,9,0)	33	144.20	65.21	77.39	31.96
w07, 14	seq 2	252	75689	(126,10,0)	47	227.88	103.64	121.74	48.35
w07, 1	flow	124	1482	(12,1,0)	6	0.40	0.23	0.17	0.14
w07, 3	flow	398	8494	(25,4,0)	50	10.14	2.61	7.51	3.33
w07, 7	flow	927	22387	(60,6,0)	73	64.41	18.09	46.26	20.64
w07, 10	flow	1349	31688	(93,8,0)	82	127.51	32.22	95.18	31.75
w07, 14	flow	1875	48781	(127,9,0)	92	287.55	69.38	216.39	59.08

* for sequential approaches the maximal number over all iterations is listed.

Table 3.5. Computational results for winter 2007.

Chapter 4

Crew Pairing Problem

In this chapter we describe the crew pairing problem and our solution approach in detail. We use a commercial solver for the crew pairing problem and hence obey all rules and restrictions applicable to the problem instances we are interested in. The generated solutions are therefore ready to be implemented in practice. All scenarios assume an aircraft routing solution as input together with the schedule data. Note that the model and all solution methods described are featured in the commercial solver. We add the AircraftChangeCost mechanism as described in Section 4.3. The cost constraint approach (Section 4.4.4) is introduced in Ehrgott and Ryan [2002].

4.1 Model

Given a flight schedule, the *crew pairing problem* is defined as the problem of assigning generic crews to flights in the schedule such that each flight is operated by exactly one crew. A sequence of flights which can be flown by a crew on one work day is called a *duty period*. After each duty period a *rest period* must be assigned to each crew member. An alternating sequence of duty periods and rest periods is called a *(crew) pairing* or *tour of duty*. Any crew pairing must start and end at the same crew base and is restricted by a number of rules such as rest time regulations or flying time restrictions (see Section 4.2). There are costs associated with each crew pairing. In the crew pairing problem we seek a minimal cost set of crew pairings that partition the flights in the schedule, i.e. each flight is contained in exactly one pairing.

The crew pairings can be represented as columns of a binary $m \times n^P$ matrix A^P where m is the number of flights in the schedule and n^P is the number of possible crew pairings. Entries $(a_{ij})^P$ of matrix A^P are defined as follows:

$$(a_{ij})^P = \begin{cases} 1 & \text{if flight } i \text{ is contained in pairing } j \\ 0 & \text{otherwise,} \end{cases}$$

with $1 \leq i \leq m, 1 \leq j \leq n^P$. With this matrix representation we formulate the crew pairing problem as a standard *set partitioning model*:

$$\begin{array}{lrcl} \text{Minimise} & (\boldsymbol{c}^P)^T \boldsymbol{x}^P & & \\ \text{subject to} & A^P \boldsymbol{x}^P & = & \mathbb{1} \\ & \boldsymbol{x}^P & \in & \{0,1\}^{n^P}. \end{array} \tag{4.1}$$

The element c_j^P of $\boldsymbol{c}^P \in \mathbb{R}^{n^P}$ is the cost associated with pairing j. The decision variable $x_j^P \in \{0,1\}$ has value 1 if pairing j is contained in the solution and 0 otherwise. The cost of a pairing is composed of a combination of flight time and duty time salaries, and meal, rest, and travel allowances (see Section 4.2). *Base-constraints* are added to the standard model to consider *base strengths* at the crew bases. The base strength restricts the number of crew pairings that can start at a crew base in a particular week or on a particular day. To include these restrictions, constraints are appended to formulation (4.1) for each combination of day (or week) and crew base where base restrictions apply. The columns of matrix A^P are appended by the following entries:

$$(a_{i+m,j})^P = \begin{cases} k_j & \text{if pairing } j \text{ starts at the crew base and in the time interval} \\ & \text{specified by base-constraint } i \\ 0 & \text{otherwise,} \end{cases}$$

with $1 \leq i \leq m^{BC}, 1 \leq j \leq n^P$. Integer value $k_j \in \mathbb{N}$ specifies the number of working days that are necessary to operate crew pairing j and m^{BC} is the total number of base-constraints. The base-constraints usually have inequality signs and non-unit integer right hand sides. The integer right-hand side ensures that at least ($\geq$) or at most ($\leq$) a given number of crew pairings start on a particular day or during a particular week from a given crew base. With base-constraints included the model is referred to as *generalised set partitioning model*:

$$\begin{array}{lrcl} \text{Minimise} & (\boldsymbol{c}^P)^T \boldsymbol{x}^P & & \\ \text{subject to} & A^P \boldsymbol{x}^P & = & \boldsymbol{b}^P \\ & \boldsymbol{x}^P & \in & \{0,1\}^{n^P}, \end{array} \tag{4.2}$$

where the entries of $\boldsymbol{b}^P$ have value 1 for all flight partitioning constraints and are non-negative integers for base-constraints. Besides crew pairing columns and variables, matrix A^P and variables $\boldsymbol{x}^P$ also contain slack and surplus columns and variables to satisfy the inequality base-constraints.

In the crew pairing formulation (4.2) generic crew members are assigned to flights. This is the main difference from the aircraft routing model (3.1) that assigns individual aircraft to flights. The generic crew pairings are assigned to particular crew members in a subsequent step in the crew rostering problem (see Section 2.1.6). Convexity constraints which ensure that each pairing is operated by the appropriate number of crew members are therefore not included in formulation (4.2). The total number of crew to operate all flights in the schedule is not known a priori but determined by the solution. However, the number of crew available is limited by the base-constraints.

Similar to the aircraft routing problem, the number of feasible crew pairings n^P is very large and we use column generation to solve the $\mathcal{NP}$-hard problem.

We model the network in the same way as for the aircraft routing problem, i.e. arcs represent flights as well as connections between flights if a crew can operate the two flights consecutively. A copy of a flight arc is added to the network if the crew may travel as passengers on this flight (also called *passengering*). These arcs are only included in the network. No entries are added to the column in the set partitioning formulation for passengering flights. Two columns may exist that cover exactly the same flight constraints, but one column contains a passengering sector while the other column does not. The two columns are distinguished by the objective coefficient and the entries covering the base-constraints. This is in contrast to a set covering formulation where passengering a sector is treated identically to operating the sector and the equality constraints are replaced by greater-than-or-equal constraints (see for example Wedelin [1995]). The set covering model results in fewer variables but costs and rules associated with passengering cannot be modelled accurately. Similarly to flight arcs, copies of connection arcs are added to the network if it is possible for crew to have a meal break during this connection: one connection arc represents the crew having a meal break during their stay at the airport while the other arc copy represents the crew not having a meal break. Each crew pairing is represented by a path in this network. The arcs contained in the path determine when meal breaks occur or if the crew travel as passengers. For each column the corresponding path must be stored so that this information can be retrieved for a solution. A resource constrained shortest path algorithm is used to find feasible crew pairings.

4.2 Rules

In this section we describe the rules that are applicable to feasible crew pairings. We limit the description to the most important rules, in particular with respect to robustness and the integration of aircraft routing and crew pairing problems as described in the following chapter. Some additional rules are imposed by Air New Zealand but these do not affect the characteristics of the results and are therefore omitted. Since we use the airline's crew pairing solver all of them are satisfied by the generated crew pairing solutions. All rules described apply to scenarios of dated scheduling periods. The crew pairing solver can also be used to solve cyclic problems (see Section 2.1.1).

- CrewCost
  The cost of a pairing is a sum of costs for working time (flying or on the ground), daily expenses allowances (paid for each day away from base), staying overnight, transportation, hotel rooms, meal breaks, idle time, working overtime, passengering crew, and other factors. Most of these costs can be assigned to arcs, for example flying time costs, overnight costs or passengering costs. Other (so called *non-arc*) costs must be calculated during the resource constrained shortest path calculation. An example is the cost of working overtime since this cost cannot be assigned to any arc because it depends on the starting time of the duty period. The cost resource of a label accumulates all arc and non-arc costs of the

path the label represents.

Additional penalty costs can be added to the cost of the path to influence certain characteristics. For example, crew can be discouraged from changing aircraft by adding penalties to the costs of connection arcs that represent aircraft changes, see AircraftChangeCost below.

- MinSitTime
  The *minimal sit-time* is the minimal time required between the arrival and departure of two flights a crew can operate in sequence. The minimal sit-time depends on the port and on the flight types (domestic or international) of the two flights. The minimal sit-time is usually shorter when crew stay on the same aircraft compared to when they change aircraft to give crew enough time to transfer to the departure gate of their next flight. The minimal sit-time is also different for crew that operate the second flight compared to crew travelling as passengers on the second flight. Only arcs that satisfy the minimal sit-time rule are included in the network.

- MaxSitTime
  Similarly to MinSitTime, the maximal sit-time specifies the maximal time on the ground for a crew member between consecutive flights he or she can operate. Connection arcs violating this rule are not included in the network.

- AircraftChangeCost
  Costs can be imposed for crew changing aircraft when the sit-time is below some threshold. The rule is explained in detail in the following section.

- DutyPeriodAircraftChangeLimit (DPACLim)
  The limit specifies how often a crew member can change aircraft during one duty period. The rule is explained in detail in the following section.

- BaseConstraints
  A fixed number of crew bases is located throughout the network. All crew pairings start at one of the crew bases and must end at the same one. For each day and week of the scheduling periods we consider, a minimal and maximal number of crew pairings is specified that can start

at a crew base on that day or week, respectively. This requirement is included in the set partitioning formulation (4.2) as knapsack constraints with non-unit right-hand sides for each day or week and crew base.

- MaxSectors
  Each duty period can contain a maximal number of flights which is enforced by a resource during the shortest path calculation.

- MaxDutyPeriods
  Any crew pairing can contain a maximal number of duty periods. This is enforced by a resource during the construction of the shortest path.

- LeadIn
  As general practice the airline solves the crew pairing problem for a fixed period (e.g. a week) of the schedule at a time. This is necessary since the number of crew available at each crew base changes over time. The schedule itself is also different for each week. Whenever a period is solved, some crew pairings that span multiple days may continue into the subsequent period. When such a subsequent period is solved, *lead-in* crew pairings that started in the previous period and continue into the current period must be taken into account. The user can specify a list of lead-in crew pairings, usually as a result of the solution of the previous period. All flights of the current period that are already operated by crew from the last period are removed from the set partitioning formulation since they do not need to be covered again. This ensures that the solution of the current period can be appended to the solution of the previous period without operating the same flight multiple times.

- ForceBan
  The user can specify a set of connections that are forced, i.e. must be operated by some crew, or banned, i.e. cannot be operated by any crew. This is enforced in the network design by only including feasible connection arcs with respect to the force and ban input data.

- Passengering (Deadheading)
  Crew can travel as passengers to operate a flight that is not departing from their current location or to return back to their home base. Passengering (also referred to as *deadheading*) may be permitted anytime

during a duty period, only at the start or the end of a duty period, or not at all. Passengering is considered in the network by adding a copy of a flight arc if the crew can use this flight to travel as passengers. These flight copies are only part of the network and not present in the set partitioning formulation. Additional costs for passengering can be added to the passengering arcs.

- TimeLimits
  Limits on the total length (in minutes) of a duty period or a crew pairing are enforced during the shortest path calculation by a resource.
- MinRestTime
  A minimal time of rest is required between two consecutive duty periods. The required rest needs to be enforced during the shortest path calculation by a resource since it is a function of the working time during the duty periods.
- MaxFlightTime
  Other restrictions on the flying time include maximal working time (in minutes) in any 24 hour rolling time window. This rule is enforced by a resource in the shortest path algorithm. The working time depends not only on the current duty period but also on previous duty periods.
- MealBreaks
  Meal breaks must take place in certain time windows within the duty period, and there must be sufficient ground time available. Furthermore, in-flight meals incur additional costs. Arcs are duplicated in the network if a meal break can take place during a connection or a flight. One copy of the arc includes the meal break while the other arc does not. Additional costs incurred by the meal break can be assigned to the appropriate arc. Other meal break rules are enforced during the shortest path calculation, e.g. maximal allowed time between two consecutive meal breaks.

4.3 Operational Robustness

An airline schedule is unlikely to be operated as planned because of disruptions. Delays occur frequently in airline operations and can for example be caused by late passengers, unscheduled maintenance requirements, or bad weather. Such disruptions cannot be controlled by the airline. Since the aircraft and crew that operate a delayed flight usually operate further flights that depart later during the day, these flights may be delayed due to the unavailability of aircraft or crew. In this section we describe how we minimise such subsequent delays.

A solution, where effects of potential delays are minimal, is called *operationally robust*. The concept of robust solutions is important since an airline is interested in achieving high *on-time performance (OTP)*, i.e. a high percentage of all flights in the schedule departs on-time. However, the *planned cost* of a more robust solution is usually high since slack is built in the schedule to compensate for delays. Bad OTP can incur large additional costs (referred to as *recovery costs*), caused by additionally required crews, compensation for passengers affected by delayed or cancelled flights, and damaged reputation of the airline. These additional costs may be much larger than the savings of using a solution with less planned cost that is also less robust. We try to identify solutions with low planned costs which are operationally robust, i.e. where disruptions will result in minimal recovery costs. Costs listed in this thesis are generally planned costs. We refer to the sum of planned costs and recovery costs as *operational costs*.

Before we describe how to obtain operationally robust solutions, we explain the concepts of short and restricted connections as introduced in Mercier et al. [2005]. First, we repeat some definitions from Chapter 2. If two flights can be operated in sequence by the same crew or aircraft (i.e. there exists a connection-arc linking both flights), the time between arrival of the incoming and departure of the outgoing flight is called *turn-time* for aircraft and *sit-time* for crew.

The minimal time required for an aircraft or a crew to operate a connection is called *minimal turn-time* or *minimal sit-time*, respectively. The required minimal sit-time can exceed the minimal turn-time. For example, crew need enough time to travel from the arrival gate, through the terminal(s), to the departure gate of the next flight. If crew stay on the same aircraft, the minimal turn-time for this connection also applies to crew, instead of the minimal sit-time. A connection between flights i and j is called *short* if

$$(\text{minimal turn-time})_{ij} \leq (\text{sit-time})_{ij} < (\text{minimal sit-time})_{ij}.$$

Thus, in a feasible solution, short connections are only allowed if crew stay on the same aircraft. Since we solve the crew pairing problem for a given aircraft routing solution this requirement is easily incorporated into the network construction by not including arcs for short connections that are operated by different aircraft.

In addition to allowing short connections only when crew stay on the same aircraft, we also prefer solutions where crew are not changing aircraft when the turn time is less than some *restricted time*. A connection between two flights i and j is called *restricted* if

$$(\text{minimal sit-time})_{ij} \leq (\text{sit-time})_{ij} < (\text{restricted time})_{ij}.$$

In contrast to short connections, crews are allowed to change aircraft if the connection is restricted, but we try to find solutions in which this occurs as rarely as possible. If crew change aircraft on restricted connections we refer to these connections as *restricted aircraft changes*.

We minimise the number of restricted aircraft changes that are operated to obtain operationally robust solutions. Minimal turn-times are usually operated in aircraft routings to keep costs low and connection times attractive for passengers. Hence, if a flight is delayed, the flight operated next by the aircraft is probably also delayed. In the aircraft routing chapter we minimise the number of minimal turn sequences in the solution to keep such subsequent delays small. If the crew are also changing aircraft on a restricted connection after the delayed flight, other flights might be affected by the initial delay. Due to the small buffer to compensate for the delay, the crew are likely to be late for the next flight they operate. This behaviour can propagate to a large number of delayed flights within a short amount of time. The solution is expected to be operationally more robust if crew change aircraft only when the sit-time provides sufficient buffer to compensate for a delay.

In Figure 4.1 examples of a non-robust (top) and a robust solution (bottom) are depicted. Flights are represented as rectangles with origin and destination airports indicated by 3-letter-codes. Aircraft routings are represented as rows of flights while crew pairings are represented as flights connected by lines. Dashed rectangles represent flights that are delayed. We can see in the first scenario that two flights are affected by the initial delay because the crew

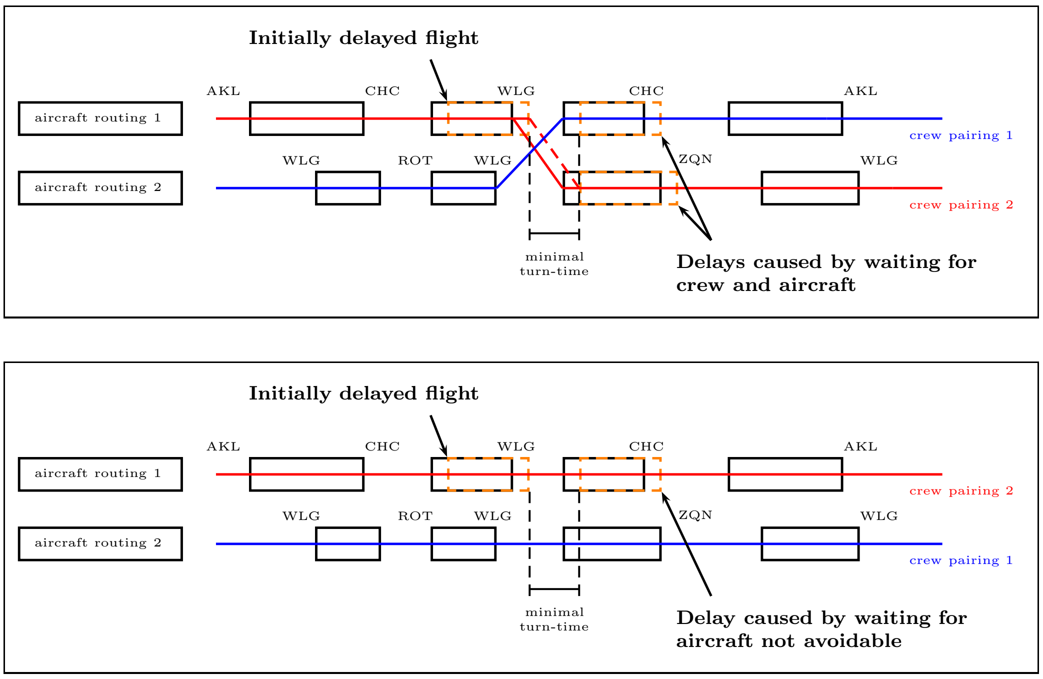


Figure 4.1. Comparison of a non-robust and a robust solution

operates a restricted aircraft change. In the second scenario only one other flight is affected by the initial delay.

There are two mechanisms in the crew pairing solver to limit the number of restricted aircraft changes: AircraftChangeCost and DPACLim rule.

AircraftChangeCost
Costs can be imposed for crew changing aircraft when the sit-time is below some threshold (restricted time). By minimising these costs as part of the objective function we encourage the crew to stay on the same aircraft whenever the sit-time is small.

The airline does not impose any costs for aircraft changes. In the computational experiments we analyse the impact of penalising all aircraft changes with a sit-time exceeding the minimal sit-time by 30 minutes or less. We impose costs that increase linearly with decreasing sit-time. The cost for changing aircraft on a restricted connection ij is denoted by c_{ij}^{AC}:

$$c_{ij}^{AC} = (k_1 - ((\text{sit-time})_{ij} - (\text{minimal sit-time})_{ij})) * k_2. \qquad (4.3)$$

Weights k_1 and k_2 are chosen such that c_{ij}^{AC} equals 7 for a restricted aircraft change with sit-time equal to the minimal sit-time, 6 for a restricted aircraft change with sit-time exceeding the minimal sit-time by 5 minutes, and so on until a weight of 1 is assigned to a restricted aircraft change with sit-time exceeding the minimal sit-time by 30 minutes. Note that all departure and arrival times in the schedules we consider are at 5 minute intervals starting from midnight. A different set of connections (e.g. those with a larger sit-time) or a different function of the sit-time (e.g. where weights increase exponentially with decreasing sit-time) could be chosen in a straightforward way. Comparing the scale of the aircraft change costs, the cost of a single day crew pairing varies between 500 and 1000.

Let RC denote the set of restricted connections. For a crew pairing solution $\boldsymbol{x}^P$, $RC(\boldsymbol{x}^P)$ is the set of restricted connections used in this solution. If $\boldsymbol{x}^R$ is a solution to the aircraft routing problem, then $\overline{RC}(\boldsymbol{x}^R)$ is the set of restricted connections induced by this solution. With this notation, the aircraft change cost c^{AC} of a crew pairing solution is the sum over all restricted aircraft changes that are contained in the solution:

$$c^{AC} = \sum_{ij \in \overline{RC}(x^R)} c_{ij}^{AC} \sum_{k, ij \in k} \boldsymbol{x}_k^P. \tag{4.4}$$

Here, k is used to index crew pairings and $ij \in k$ is used to indicate that connection ij is used in crew pairing k. To include this rule in the algorithm, the costs for changing aircraft are assigned to connection arcs of the column generation network.

DutyPeriodAircraftChangeLimit (DPACLim)
The limit specifies how often a crew member can change aircraft during one duty period. The airline introduced this rule in an attempt to increase the robustness of the solutions. In contrast to penalising aircraft changes, the DPACLim rule limits the total number of aircraft changes in a solution, independently of the sit-time of these aircraft changes. A resource is added to the labels to enforce this rule during the shortest path calculation. It is possible to count the number of aircraft changes during a duty period because an aircraft routing solution is given as input. This rule needs special attention in Chapter 5 when aircraft routing and crew pairing problems are integrated and

the aircraft routing solution is no longer fixed. We also analyse the impact of relaxing this rule in the computational experiments.

4.4 Solution Methods

In this section we describe an efficient solution method for the crew pairing problem. We use column generation and branch-and-price methods to solve the problem. We utilise the commercial crew pairing solver that is used by Air New Zealand. As a consequence, the crew pairing solutions satisfy all operational rules and requirements that are imposed by the airline. We describe the LP solution, column generation, and branch-and-price methods of the crew pairing solver in the following.

4.4.1 LP-Relaxation

The LP relaxation of problem (4.2) is solved by the simplex method with the ZIP (Zero-One Integer Programming) package (Ryan [1980]) which is written in FORTRAN. ZIP is a specialised zero-one integer programming solver that is equipped with many call-back functions to allow the user to control each step of the simplex algorithm and the branch-and-price process. The user can for example specify particular rules to determine entering or leaving variables or how branching is performed to obtain integer solutions. The column generation routine is called in each pricing step of the simplex algorithm if no entering column can be found among the non-basic columns of the matrix. We employ steepest edge pricing in the simplex algorithm.

Quite often crew pairings exist already that cover flights of the scenario period. These crew pairings may have been generated by solving a scenario for the same or another period with a very similar schedule. The pairings can be used to speed up the computation process. The user can specify a list of previously generated crew pairings that are available to the optimiser. These crew pairings are checked for feasibility and negative reduced cost and are added to the matrix A^P of the restricted master problem before the column generator is called. Also, the optimal basis of a previous solution can be used as an initial starting basis of the simplex algorithm. If some of the columns of the previous basis are now infeasible, these columns are treated as artificial columns in a phase I/II approach.

4.4.2 Column Generation

Column generation is performed on a flight and connection based network. Crew pairings correspond to paths in this network. Only connection arcs that satisfy all rules are included in the network. All rules that require taking multiple arcs into account are modelled as resources in the shortest path algorithm. We use a label setting algorithm as described for the aircraft routing problem in Section 3.3.3 to find paths with negative reduced costs. Compared with the aircraft routing generator, many more resources must be considered when solving the crew pairing problem because of the more complicated rule structure. The network also consists of many more arcs than the aircraft routing network because we cannot eliminate arcs based on the number of crew at an airport. Since crew pairings contain many fewer flights than aircraft routings, at most six per duty period, the total number of crew pairings we generate to obtain an optimal solution, is smaller than the total number of generated aircraft routings.

We use a *dominance relaxation* method to achieve fast solution times for the resource constrained shortest path calculation. In this method each time the column generation routine is called, a shortest path algorithm is executed multiple times in so called *stages*. Each stage can return crew pairings with negative reduced costs. A particular stage is only called if all previous stages did not compute any path with negative reduced cost. In all but the very last stage only a subset of all resources are considered when dominance is checked between two labels. In the first stage for example, a pure shortest path problem could be solved by only keeping the cheapest label at each node. Since infeasible labels are removed from each node, this stage may not return a negative reduced cost path even though such a path exists. In subsequent stages more and more resources are considered in the dominance check. The user can choose to consider all resources in the dominance check of the last stage. But the user can also only use a limited number of stages not considering all resources in the last stage. The latter method cannot guarantee an optimal solution but may reduce solution times significantly. Care must be taken to consider a “good” set of resources when checking for dominance in order to obtain good quality solutions. This needs to be verified by extensive computational experiments.

4.4.3 Branch-and-Price

To solve the crew pairing problem, first the LP relaxation of (4.2) is solved using column generation. Fractional variables in the solution are caused by different crew pairings competing for the same flights. To eliminate these fractions and obtain an integer solution for (4.2), a branch-and-price algorithm with a *follow-on branching* strategy is used. In this strategy two flights must be operated consecutively by the same crew in one problem called the 1-branch. In the 0-branch the two flights must not be operated consecutively by the same crew. The branching restrictions are enforced in the column generation network by removing arcs from the network. In the first branch only the forced connection arc leaves the arriving flight node and enters the departing flight node of the connection. All other arcs leaving the departing flight node or entering the arriving flight node are removed. In the second branch the connection arc that represents the follow-on connection is removed.

Since the LP solver in ZIP is integrated into the branch-and-price framework, enforcing of branching decisions works slightly differently to CPLEX. Whenever the LP relaxation is solved at a node, the optimal basis of the LP relaxation at the node solved previously is used as a starting solution. Since new branching decisions are made some variables that are part of the previous LP basis may violate this decision. These variables are forced out of the basis with a phase I/II approach. Infeasible (with respect to the branch) non-basic variables cannot enter the basis. Column generation is used to ensure an optimal solution of the LP relaxation at each node of the branch-and-bound tree. Depth first branching is used to obtain an integer solution.

4.4.4 Cost Constraint Approach

In this section we outline an enhanced solution method for the crew pairing problem to obtain cost efficient and operationally robust solutions of the crew pairing problem. This method is introduced in Ehrgott and Ryan [2002]. In this approach the crew pairing problem is essentially solved twice. First, the original crew pairing problem problem (4.2) is solved to crew pairing cost optimality. Then, a *cost constraint* is added to the set partitioning formulation that forces the crew pairing cost c^P of the solution to be less than some value ϵ. The crew pairing problem is solved again with the objective of minimising aircraft change cost c^{AC} (4.4) and subject to the set partitioning constraints and the new constraint limiting the crew pairing cost.

The value ϵ is set to exceed the optimal IP solution value by a specified percentage: $\epsilon = (1 + o/100) \times c^{IP}$. Here c^{IP} denotes the optimal IP solution value and o denotes the percentage increase in the objective one is prepared to invest to obtain a more robust solution. The cost constraint is added to the set partitioning formulation as an elastic constraint since a hard constraint causes computational difficulties in the branch-and-bound process as described in Ehrgott and Ryan [2002]. With the cost constraint and the objective to minimise aircraft change costs the crew pairing problem can be formulated as follows:

$$\begin{array}{llcccccl}
\text{Minimise} & (\boldsymbol{c}^{AC})^T\boldsymbol{x}^P & & & + & ts_u & & \\
\text{subject to} & A^P\boldsymbol{x}^P & & & & & = & \boldsymbol{b}^P \\
 & (\boldsymbol{c}^P)^T\boldsymbol{x}^P & + & s_l & - & s_u & = & \epsilon \\
 & & & & & \boldsymbol{x}^P & \in & \{0,1\}^{n^P}.
\end{array} \tag{4.5}$$

The elements $\boldsymbol{x}^P, \boldsymbol{c}^P, \boldsymbol{b}^P$ and A^P of formulation (4.5) are identical to formulation (4.2). The aircraft change cost $\boldsymbol{c}_j^{AC}$ that is assigned to pairing j is the sum over all restricted aircraft changes contained in the pairing. Non-negative variables s_l and s_u represent slack and surplus of the cost constraint. The surplus variable is penalised in the objective function by parameter t which represents the trade-off between crew pairing costs and aircraft change costs. Details on how to obtain values for t are given in Section 4.5.1. The constraint is only elastic during the IP solution phase. During the LP solution phase the surplus variable s_u is set to 0, i.e. the cost constraint is a hard constraint. The optimal solution of (4.5) incurs minimal aircraft change costs while the crew pairing cost of the solution does not exceed the optimal crew pairing costs by more than a given percentage.

4.5 Computational Experiments

All computational results in this section are obtained using the unmodified crew pairing solver of Air New Zealand. We use the results as reference data for the experiments in the following chapters. For all schedules, crew need to be assigned to flights that are operated by a single aircraft type. Crew can travel as passengers on additional flights operated by other aircraft types. We consider the same schedules discussed in the aircraft routing chapter, summer 2005 (s05), winter 2005 (w05), summer 2006 (s06), and winter 2007 (w07). For each schedule we again solve periods of 1 day, 3 days, and 7 days. For the winter 2007 schedule we additionally solve 10 and 14 day periods. We solve the scenarios for three different crew types, namely captains (c33), first officers (f33), and cabin crew (spsr). The cost structure and rules vary for different crew types and for different schedules. *Technical crew*, i.e. captains and first officers are much more expensive than cabin crew. A duty period for technical crew may contain at most 5 flights, while a duty period for cabin crew may contain 6 flights (MaxSectors rule). No AircraftChangeCost penalties apply and the DPACLim is by default set to 1 for technical crew and 2 for cabin crew, i.e. cabin crew can change aircraft twice during a duty period and technical crew only once. All crew are located at three bases. The strengths at each base vary for each crew type and each schedule. All scenarios use a given aircraft routing solution as input. This aircraft routing solution is constructed manually by the airline. We do not consider any (lead-in) crew pairings from previous solutions.

We apply model (4.2) to various scenarios and results of the computational experiments are summarised in Table 4.1. The first two columns list scenarios and crew types. Schedules summer 2005 and winter 2005 are solved for all crew types. For schedules summer 2006 and winter 2007 the rules and base strengths for captains and first officers are identical and we only show results for first officers. No data is available for cabin crew for schedule winter 2007.

Columns “arcs”, “rows” and “cols” list the total number of arcs in each network, the number of constraints in each model, and the total number of columns generated. The next four columns display solution times. All times are given in seconds. Total solution times (“tot”), LP solution times (“lp”), branch-and-bound solution times (“ip”), and column generation times (“colgen") are shown. The LP and IP solution times contain the time spent for column generation and the column generation time is the total time of LP and IP solution phases. The total solution time contains, besides LP and IP solution times, pre- and post-processing times. We observe that run times are much faster for the cabin crew problems than the technical crew problems. The cabin crew problems are easier to solve because the set partitioning models contain fewer and less restrictive base-constraints and the crew pairing rules are easier to satisfy.

Column "BnB-nodes" shows the number of branch-and-bound nodes to obtain an integer solution. The next three columns ("crew pairing costs") list the costs of the LP and IP solution and the gap between the two solutions in percent. We observe that in most cases the actual gap is much smaller than the stopping criterion which is set to 2%.

The last set of columns gives details on the number of restricted aircraft changes of each solution. The last seven columns show the number of aircraft changes where the sit-time of the connection exceeds the minimal sit-time by 0, 5, 10, 15, 20, 25, and 30 minutes. The total cost (column "c^{AC}") for these aircraft changes is obtained by summation (4.4). For the scenarios that span one week around 30 to 40 crew members change aircraft on a connection with sit-time equal to the minimal sit-time. The total aircraft change costs vary from 344 to 639 for weekly scenarios. More restricted aircraft changes are operated by cabin crew than technical crew because 2 aircraft changes are allowed for cabin crew during a duty period instead of 1 for technical crew. Note that aircraft changes are not penalised in the objective function during the execution of the algorithm and therefore a large number of restricted aircraft changes is expected.

					times				BnB-	crew pairing costs (c^P)			aircraft changes							
														(minutes exceeding min. sit-time)						
scenario	crew	arcs	rows	cols	tot	lp	ip	colgen	nodes	lp	ip	gap	c^{AC}	0	5	10	15	20	25	30
s05, 1	c33	23058	546	1296	9.50	9.04	0.20	8.10	4	398.35	398.54	0.05	53	4	1	1	2	1	0	3
s05, 3	c33	35905	784	3568	55.00	52.99	1.54	40.42	10	977.85	978.40	0.06	212	18	3	4	5	5	2	9
s05, 7	c33	59731	1192	10289	299.62	196.68	101.95	117.62	162	1760.76	1776.49	0.89	372	31	7	7	9	4	7	16
s05, 1	f33	23032	548	1335	9.37	8.97	0.15	8.00	0	347.13	347.13	0.00	34	2	1	1	1	0	1	3
s05, 3	f33	36005	786	4079	55.32	50.75	4.11	39.55	19	892.95	900.84	0.88	145	10	4	2	2	4	5	11
s05, 7	f33	59852	1194	9488	217.13	185.88	30.19	92.99	78	1725.12	1731.77	0.38	344	29	5	5	10	4	8	18
s05, 1	spsr	21785	515	1269	8.86	8.52	0.16	7.73	2	121.62	121.83	0.17	79	7	1	1	2	1	0	8
s05, 3	spsr	34476	749	2900	24.89	23.79	0.79	18.88	12	319.23	322.48	1.01	192	18	1	3	5	1	0	22
s05, 7	spsr	56997	1151	6201	73.70	63.56	9.52	27.30	65	687.89	698.35	1.50	485	42	5	7	15	3	3	51
w05, 1	c33	21464	557	1604	22.63	22.14	0.27	20.51	4	405.25	405.65	0.10	92	7	2	2	1	1	4	6
w05, 3	c33	33413	797	4135	59.02	49.99	8.64	37.82	32	935.23	936.17	0.10	196	17	4	2	3	3	6	10
w05, 7	c33	54372	1206	10316	341.27	238.96	101.48	125.85	72	1707.69	1714.29	0.39	443	41	12	9	3	2	2	17
w05, 1	f33	21449	558	1358	10.20	9.80	0.18	8.83	2	313.74	313.93	0.06	62	4	1	2	2	1	1	5
w05, 3	f33	33398	799	3826	49.82	45.87	3.56	35.23	33	787.84	789.61	0.22	183	15	3	5	3	4	1	9
w05, 7	f33	54357	1207	9350	216.68	174.42	41.41	88.66	92	1663.62	1684.42	1.23	374	30	9	8	5	5	8	19
w05, 1	spsr	24788	517	1164	6.93	6.62	0.14	5.92	3	106.42	106.66	0.22	48	5	0	0	1	0	0	9
w05, 3	spsr	38516	753	2821	23.10	21.99	0.83	16.92	15	286.12	287.60	0.51	166	12	3	3	6	1	0	22
w05, 7	spsr	62460	1155	6270	86.65	64.79	21.28	35.44	136	663.48	680.16	2.45	448	36	4	11	14	2	1	53

					times				BnB-	crew pairing costs (c^P)				aircraft changes						
														(minutes exceeding min. sit-time)						
scenario	crew	arcs	rows	cols	tot	lp	ip	colgen	nodes	lp	ip	gap	c^{AC}	0	5	10	15	20	25	30
s06, 1	f33	24824	552	1333	11.29	9.97	1.05	9.84	5	314.36	315.65	0.41	104	10	2	1	2	1	1	4
s06, 3	f33	38413	789	3353	32.49	28.37	3.63	22.13	13	719.62	721.79	0.30	252	24	2	7	3	3	1	14
s06, 7	f33	63360	1190	8454	195.06	163.83	30.23	96.51	80	1644.17	1673.21	1.74	385	35	2	10	10	4	4	18
s06, 1	spsr	21567	519	1046	8.01	7.71	0.17	7.02	5	101.02	101.90	0.87	96	8	1	2	3	1	0	9
s06, 3	spsr	33486	754	2411	19.59	18.46	0.90	13.97	19	282.60	285.82	1.13	260	23	1	7	6	1	1	29
s06, 7	spsr	54029	1152	5520	71.56	64.65	6.45	33.24	29	696.37	709.07	1.79	639	60	7	11	14	3	4	49
w07, 1	f33	24305	529	1249	9.19	8.27	0.62	7.87	2	260.99	261.80	0.31	81	5	4	1	2	1	2	2
w07, 3	f33	38448	767	3758	49.94	44.15	5.30	38.06	13	682.85	690.05	1.04	191	19	5	2	1	1	4	3
w07, 7	f33	63322	1173	8602	195.68	165.99	28.70	104.68	62	1543.18	1553.93	0.69	469	41	10	5	11	7	6	20
w07, 10	f33	84481	1529	12491	352.52	277.52	73.41	144.72	86	2216.91	2228.80	0.53	629	57	10	9	14	7	10	28
w07, 14	f33	109395	1932	18774	822.36	638.33	181.52	271.65	149	3109.13	3132.19	0.74	789	70	15	10	20	6	11	39

Table 4.1: Computational results for captain, first officer, and cabin crew scenarios, summer 2005, winter 2005, summer 2006, and winter 2007.

Table 4.2 lists detailed results for the first officer schedule of summer 2005. In these computational experiments we vary DPACLim and AircraftChange-Cost to observe the relationship between these parameters, the crew pairing costs, and the aircraft change costs of the solutions. We solve periods of 1, 3, and 7 days. We use values of 1, 2, and 4 for the maximal aircraft changes per duty period (column "DPACLim"). Note that value 4 disables the rule since each duty period contains at most 5 sectors. Instead of minimising crew pairing costs only, we now solve the crew pairing problem with a weighted sum objective of crew pairing costs and aircraft change costs:

$$\text{Minimise } c^P + pc^{AC},$$

where c^P represents crew pairing cost and c^{AC} represents aircraft change cost of the solution and penalty $p \in \{0, 10, 50\}$ which is listed in column "p".

We list costs for LP and IP solutions. The total costs (column "tot") are split into crew pairing costs ("c^P") and costs caused by crew changing aircraft times the penalty p ("pc^{AC}"). The gap ("gap") between total LP and IP costs is listed and given as a percentage. As in Table 4.1 we list the number of aircraft changes and the costs incurred by them. The last set of columns lists the total number of duty periods in the solution classified by the number of aircraft changes within each duty period. Column "tot" lists the total number of duty periods of the solution. Columns "0", "1", "2", "3", and "4" show the number of duty periods that contain 0, 1, 2, 3, and 4 aircraft changes (no matter if restricted or not), respectively. We observe that aircraft change costs c^{AC} can be decreased by increasing penalty p. If DPACLim is relaxed, cheaper solutions can be obtained but they contain many aircraft changes. In the following we investigate these relationships in more detail.

In Tables 4.3 and 4.4 we display the change in solution quality that results from varying AircraftChangeCost penalty p and DPACLim. The first three columns of each table are identical to Table 4.2. The next two columns list the crew pairing costs ("lp costs c^P") and the aircraft change costs ("lp costs c^{AC}") for the LP solutions, respectively. We use LP solutions since statements about cost improvements are not reliable for IP solutions because we use a branch-and-bound stopping gap of 2%. Tables 4.3 and 4.4 are sorted in a different order in order to compare values of three consecutive rows in each table.

In Table 4.3 the increase in crew cost and aircraft change cost for varying penalty p and fixed DPACLIM rule can be observed in columns six and seven. In three consecutive rows the change in solution quality is given as a percentage for changing penalty p from 0 (the default) to 10 and 50, respectively. We observe that a decrease of up to 100% of aircraft change cost can be achieved by penalising aircraft changes. A solution with fewer aircraft changes comes with the price of an increase in crew pairing costs of up to 3.13%. The second row of Table 4.3 shows a simultaneous decrease in crew pairing cost and aircraft change cost. This shows that the solution displayed in row 1 is in fact slightly sub-optimal. This small error is caused by the heuristic nature of the dominance relaxed shortest path algorithm (Section 4.4.2). We accept this error and use the heuristic method rather than an optimal method since the run time of the latter is very long. We do not observe an error that exceeds 0.1%.

The last two columns of Table 4.4 show the change in solution quality when varying DPACLIM for fixed penalty p. For each period (1, 3 and 7 days) the difference is given as a percentage comparing a limit of 1 aircraft change (the default) with limits of 2 and 4 in three consecutive rows. A decrease in crew pairing cost of up to 2.27% can be achieved by relaxing the DPACLIM rule. The cheaper solutions do contain many more aircraft changes if these are not penalised. By penalising aircraft changes, a decrease in crew costs with simultaneous decrease in aircraft change costs can be achieved as for example in the last row of Table 4.4. This solution however contains 21 duty periods with 2 aircraft changes and 2 duty periods with 3 aircraft changes that are forbidden by the default settings of the algorithm.

Figures 4.2 and 4.3 show the same solutions as in Tables 4.3 and 4.4 in objective space for the 7 day scenario of the first officer schedule, summer 2005. On the horizontal axis the crew pairing costs are depicted while the vertical axis shows the aircraft change costs. For constant DPACLIM the improvements in AIRCRAFTCHANGECOST for increasing penalties are shown in Figure 4.2. We again observe that aircraft change costs can be greatly decreased but this incurs a crew pairing cost increase. We can also see that increasing DPACLIM from 1 (green) to 2 (blue) greatly reduces costs while a value of 4 (red) does not yield significant additional improvements. Figure 4.3 shows improvements in crew pairing cost for increasing DPACLim and constant AircraftChangeCost. If aircraft changes are not penalised aircraft change costs increase significantly for smaller crew pairing costs (green). If aircraft changes are penalised, aircraft change costs remain small for decreasing crew pairing costs (blue and red). For a value of $p = 50$ we observe that crew pairing costs increase by changing DPACLim from 2 to 4 but aircraft change costs decrease due to the large value of p.

			costs							aircraft changes								duty periods				
			lp ($\times 10^2$)			ip ($\times 10^2$)			gap		(minutes exceeding min. sit-time)								aircraft changes			
days	DPACLim	p	tot	c^P	pc^{AC}	tot	c^P	pc^{AC}	(%)	c^{AC}	0	5	10	15	20	25	30	tot	0	1	2	3	4
1	1	0	347.13	347.13	0.00	347.13	347.13	0.00	0.00	34	2	1	1	1	0	1	3	36	21	15	0	0	0
1	1	10	348.29	346.79	1.50	348.29	346.79	1.50	0.00	15	1	0	0	1	1	0	1	36	21	15	0	0	0
1	1	50	348.84	348.84	0.00	350.36	350.36	0.00	0.43	0	0	0	0	0	0	0	0	36	27	9	0	0	0
1	2	0	345.53	345.53	0.00	345.54	345.54	0.00	0.00	62	4	2	1	1	3	1	2	36	21	9	6	0	0
1	2	10	347.46	346.86	0.60	347.46	346.86	0.60	0.00	6	0	0	0	1	0	0	2	36	22	12	2	0	0
1	2	50	348.60	348.60	0.00	348.63	348.63	0.00	0.01	0	0	0	0	0	0	0	0	36	25	10	1	0	0
1	4	0	344.95	344.95	0.00	344.95	344.95	0.00	0.00	55	3	3	0	1	2	1	4	36	20	11	4	1	0
1	4	10	347.60	346.70	0.90	347.60	346.70	0.90	0.00	9	0	0	0	1	1	0	2	36	23	10	3	0	0
1	4	50	348.60	348.60	0.00	348.63	348.63	0.00	0.00	0	0	0	0	0	0	0	0	36	25	10	1	0	0
3	1	0	892.95	892.95	0.00	900.84	900.84	0.00	0.88	145	10	4	2	2	4	5	11	95	58	37	0	0	0
3	1	10	900.89	896.62	4.28	909.34	904.34	5.00	0.93	50	1	1	1	2	3	2	11	95	60	35	0	0	0
3	1	50	910.39	901.70	8.69	928.72	919.72	9.00	1.97	18	1	0	1	0	1	0	3	97	73	24	0	0	0
3	2	0	884.09	884.09	0.00	892.72	892.72	0.00	0.97	194	14	4	5	5	3	2	14	96	52	31	13	0	0
3	2	10	893.24	887.75	5.49	902.80	895.80	7.00	1.06	70	2	1	3	1	3	4	14	96	61	27	8	0	0
3	2	50	903.87	893.57	10.30	930.39	922.89	7.50	2.85	15	0	0	0	0	2	1	7	98	69	25	4	0	0
3	4	0	883.80	883.80	0.00	888.26	888.26	0.00	0.50	201	15	4	4	5	2	6	14	96	53	30	12	1	0
3	4	10	893.06	887.41	5.65	917.07	910.97	6.10	2.62	61	2	2	1	2	2	4	8	97	61	30	6	0	0
3	4	50	903.87	893.57	10.30	922.87	912.87	10.00	2.06	20	0	0	1	0	2	1	7	96	67	24	5	0	0

			costs							aircraft changes								duty periods					
			lp ($\times 10^2$)			ip ($\times 10^2$)			gap		(minutes exceeding min. sit-time)								aircraft changes				
days	DPACLim	p	tot	c^P	pc^{AC}	tot	c^P	pc^{AC}	(%)	c^{AC}	0	5	10	15	20	25	30	tot	0	1	2	3	4
7	1	0	1725.12	1725.12	0.00	1731.77	1731.77	0.00	0.38	344	29	5	5	10	4	8	18	209	82	127	0	0	0
7	1	10	1744.18	1729.29	14.89	1751.62	1736.82	14.80	0.42	148	11	2	2	3	4	4	17	210	100	110	0	0	0
7	1	50	1775.07	1750.93	24.13	1811.82	1781.82	30.00	2.03	60	4	0	2	2	0	1	12	210	124	86	0	0	0
7	2	0	1688.99	1688.99	0.00	1700.13	1700.13	0.00	0.66	558	49	4	12	16	3	9	40	206	80	73	53	0	0
7	2	10	1721.96	1701.78	20.18	1735.81	1711.21	24.60	0.80	246	15	4	6	8	5	4	32	206	93	77	36	0	0
7	2	50	1754.32	1735.38	18.94	1775.26	1757.26	18.00	1.18	36	2	0	1	0	0	4	19	210	119	65	25	1	0
7	4	0	1685.98	1685.98	0.00	1707.11	1707.11	0.00	1.24	577	53	4	10	15	2	13	40	206	80	66	54	5	1
7	4	10	1718.58	1698.69	19.89	1739.10	1718.90	20.20	1.18	202	14	1	4	8	4	4	26	206	104	67	33	2	0
7	4	50	1754.76	1738.75	16.02	1768.61	1753.11	15.50	0.78	31	1	0	1	0	0	2	15	210	111	76	21	2	0

Table 4.2: Variation of DPACLim rule and AircraftChangeCost penalty p for first officer scenario, summer 2005, 7 days.

days	DPACLIM	p	lp costs		increase (%), DPACLIM fixed	
			$c^P(\times 10^2)$	$c^{AC}(\times 10)$	c^P	c^{AC}
1	1	0	347.13	3.40	-	-
1	1	10	346.79	1.50	-0.10	-55.88
1	1	50	348.84	0.00	0.49	-100.00
1	2	0	345.53	6.73	-	-
1	2	10	346.86	0.60	0.38	-91.09
1	2	50	348.60	0.00	0.89	-100.00
1	4	0	344.95	5.50	-	-
1	4	10	346.70	0.90	0.51	-83.64
1	4	50	348.63	0.00	1.07	-100.00
3	1	0	892.95	16.37	-	-
3	1	10	894.41	4.28	0.16	-73.88
3	1	50	896.55	1.74	0.40	-89.39
3	2	0	884.09	21.12	-	-
3	2	10	885.43	5.49	0.15	-74.02
3	2	50	886.74	2.06	0.30	-90.25
3	4	0	883.80	22.91	-	-
3	4	10	885.32	5.65	0.17	-75.34
3	4	50	886.66	2.06	0.32	-91.01
7	1	0	1725.12	34.09	-	-
7	1	10	1729.29	14.89	0.24	-56.33
7	1	50	1750.93	4.83	1.50	-85.84
7	2	0	1688.99	57.86	-	-
7	2	10	1701.78	20.18	0.76	-65.13
7	2	50	1735.38	3.79	2.75	-93.45
7	4	0	1685.98	63.52	-	-
7	4	10	1698.69	19.89	0.75	-68.69
7	4	50	1738.75	3.20	3.13	-94.96

Table 4.3. Improvements of solutions for variation of AIRCRAFTCHANGECOST penalty p for first officer scenario, summer 2005, 7 days.

days	DPACLim		lp costs		increase (%), p fixed	
		p	$c^P(\times 10^2)$	$c^{AC}(\times 10)$	c^P	c^{AC}
1	1	0	347.13	3.40	-	-
1	2	0	345.53	6.73	-0.46	98.04
1	4	0	344.95	5.50	-0.63	61.76
1	1	10	346.79	1.50	-	-
1	2	10	346.86	0.60	0.02	-60.00
1	4	10	346.70	0.90	-0.03	-40.00
1	1	50	348.84	0.00	-	-
1	2	50	348.60	0.00	-0.07	0.00
1	4	50	348.63	0.00	-0.06	0.00
3	1	0	892.95	16.37	-	-
3	2	0	884.09	21.12	-0.99	29.00
3	4	0	883.80	22.91	-1.02	39.95
3	1	10	894.41	4.28	-	-
3	2	10	885.43	5.49	-1.00	28.30
3	4	10	885.32	5.65	-1.02	32.10
3	1	50	896.55	1.74	-	-
3	2	50	886.74	2.06	-1.09	18.56
3	4	50	886.66	2.06	-1.10	18.56
7	1	0	1725.12	34.09	-	-
7	2	0	1688.99	57.86	-2.09	69.71
7	4	0	1685.98	63.52	-2.27	86.29
7	1	10	1729.29	14.89	-	-
7	2	10	1701.78	20.18	-1.59	35.53
7	4	10	1698.69	19.89	-1.77	33.57
7	1	50	1750.93	4.83	-	-
7	2	50	1735.38	3.79	-0.89	-21.51
7	4	50	1738.75	3.20	-0.70	-33.64

Table 4.4. Improvements of solutions for variation of DPACLim for first officer scenario, summer 2005, 7 days.

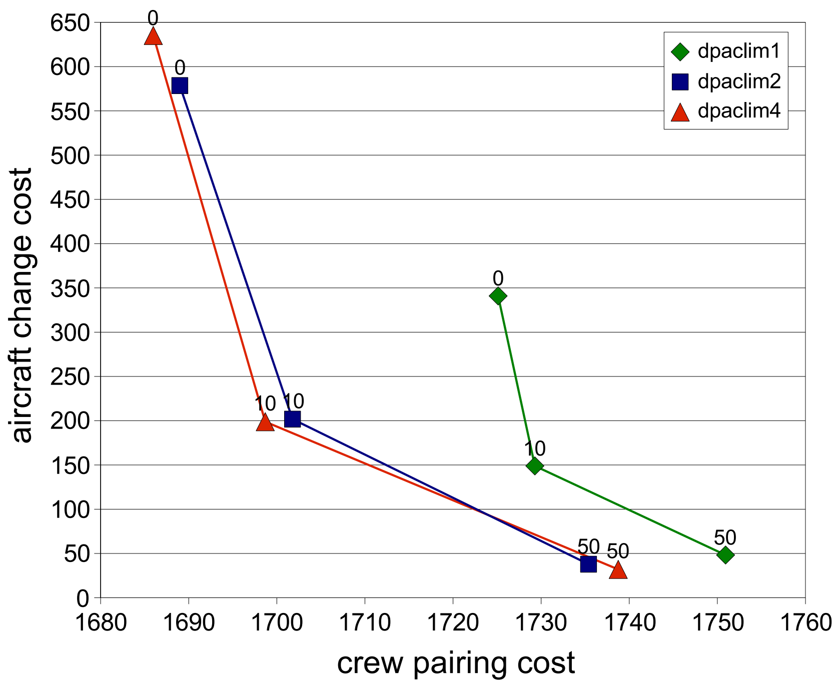


Figure 4.2. Solutions for variation of AircraftChangeCost penalty for first officer scenario, summer 2005, 7 days.

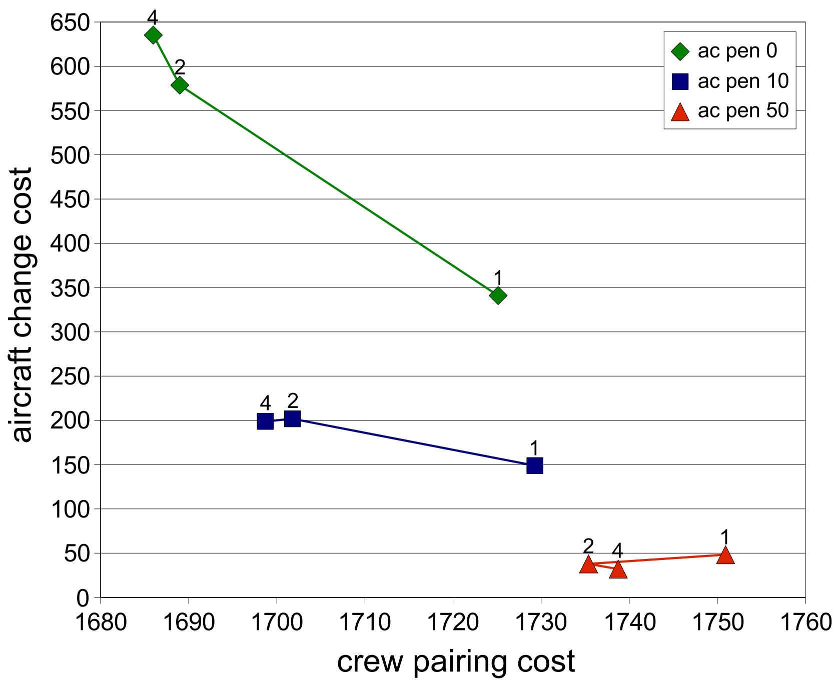


Figure 4.3. Solutions for variation of DPACLim rule for first officer scenario, summer 2005, 7 days.

4.5.1 Cost Constraint Approach

Table 4.5 summarises results of computational experiments for the cost constraint approach. We consider the first officer schedule summer 2005 with scenarios of 1, 3, and 7 days. Default settings of 0 and 1 are used for AircraftChangeCost and DPACLim, respectively. The columns "days", "o" and "t" show the number of days of the scenario and the applicable values for *o* and *t* (see Section 4.4.4). For each row value *t* was obtained by using the 2 neighbouring LP solutions (listed in the row above and below the current row). These solutions are depicted in two dimensional objective space with one dimension representing crew pairing cost and the other dimension representing aircraft change cost as in Figures 4.2 and 4.3. The value for *t* is set to the negative of the slope of the line connecting the two neighbouring solutions, hence *t* is an approximation of the trade-off between crew pairing cost and aircraft change cost. For the first and last row of each of the three scenarios *t* is set to 10. Crew pairing costs ("c^P") and aircraft change costs ("c^{AC}") are shown for the LP and IP solutions with the objective to minimise aircraft change costs. The increase ("increase") in crew pairing cost and aircraft change cost is with respect to the minimal crew pairing cost solutions listed in Table 4.1. We further list the number of restricted aircraft changes of the IP solutions and solution times in seconds. The total solution time contains pre- and post-processing times.

Improvements in aircraft change cost of more than 90% can be achieved by allowing an increase in crew pairing cost of a few percent. Even a small increase in crew pairing cost, for example of 1.2% in the second row of the 7 day scenario, enables a large decrease in aircraft change cost (78.49%). We observe that solution times roughly double compared to the standard crew pairing approach (Table 4.1). This is partly due to the fact that we solve the LP twice, once with objective to minimise crew pairing costs and once with the objective to minimise aircraft change costs. Although we start the second LP solution process from the optimal basis of the first solve, the second LP does not solve much faster than the first LP. The reason for this is that the optimal solutions of the 2 LPs are quite different due to the additional constraint. The IP is only solved in the latter case and can also be very difficult to solve. For the last five rows in Table 4.5 for example, the branch-and-bound process is stopped after the first integer solution is found and we can observe that the LP/IP gap is very large for these instances. The gap for the last instance is in excess of 300%. We stop the branch-and-bound process after the first integer solution since no significant improvement is made during further exploration of the branch-and-bound tree. As an example we explore the branch-and-bound tree up to the node limit of 2000 nodes for the second to last instance. The integer solution listed in Table 4.5 has an objective value of 14 and is found after 50 nodes. This solution exceeds the optimal LP solution value by more than 200%. No further integer solution is found within the node limit and the computation time is in excess of 4,000 seconds. The difficulties in the IP solution phase are caused by the cost constraint being active in the LP solution. Although formulation (4.2) generally yields very small LP/IP gaps, this property is destroyed by adding the cost constraint to the formulation. Even with the cost constraint being elastic, the branch-and-bound process is very difficult and time consuming.

days			lp (minimise c^{AC})				ip (minimise c^{AC})				aircraft changes										
					increase (%)				increase (%)		(minutes exceeding min sit-time)							times			
	o	t	$c^P(\times 10^2)$	c^{AC}	c^P	c^{AC}	$c^P(\times 10^2)$	c^{AC}	c^P	c^{AC}	0	5	10	15	20	25	30	tot	lp	ip	colgen
1	0.00	10.00	347.13	34.00	0.00	0.00	347.13	34.00	0.00	0.00	2	1	1	1	0	1	3	17.28	16.96	0.07	15.73
1	0.25	18.00	348.00	10.10	0.25	-70.29	348.63	4.00	0.43	-88.24	0	0	0	0	1	0	1	17.22	14.95	2.02	14.79
1	0.50	10.00	348.86	2.76	0.50	-91.87	348.81	3.00	0.48	-91.18	0	0	0	0	1	0	0	18.99	18.21	0.53	16.83
3	0.00	10.00	900.83	18.61	0.88	-88.63	916.50	18.00	1.74	-87.59	1	0	0	0	2	0	5	122.19	96.11	25.62	94.47
3	0.50	1.50	905.34	11.78	1.39	-92.80	911.87	9.00	1.22	-93.79	0	0	0	0	2	1	1	106.77	86.25	20.06	80.60
3	1.00	10.00	909.84	6.80	1.89	-95.85	942.25	3.00	4.60	-97.93	0	0	0	0	1	0	0	199.44	91.93	107.06	143.85
7	0.00	10.00	1731.76	123.89	0.39	-63.66	1753.51	134.00	1.26	-61.05	10	1	4	1	3	3	19	389.99	270.10	118.91	223.30
7	0.50	4.00	1740.42	74.65	0.89	-78.11	1752.62	74.00	1.20	-78.49	3	2	2	2	2	3	11	383.72	296.66	86.09	223.31
7	1.00	2.50	1749.08	50.84	1.39	-85.09	1762.28	45.00	1.76	-86.92	2	0	2	1	0	1	15	404.68	281.72	121.97	229.29
7	1.50	1.50	1757.74	35.26	1.89	-89.66	1774.41	28.00	2.46	-91.86	0	0	2	0	1	2	11	472.02	306.66	164.38	273.10
7	2.00	1.00	1766.40	24.56	2.39	-92.80	1782.98	27.00	2.96	-92.15	0	0	2	0	2	1	9	587.30	334.47	251.85	350.11
7	2.50	0.70	1775.06	17.21	2.90	-94.95	1788.75	21.00*	3.29	-93.90	0	0	2	0	1	0	8	424.90	312.68	111.23	230.99
7	3.00	0.50	1783.72	12.27	3.40	-96.40	1807.31	25.00*	4.36	-92.73	1	0	1	1	1	0	6	382.77	330.79	50.98	210.48
7	4.00	0.10	1801.04	8.38	4.40	-97.54	1813.61	22.00*	4.73	-93.60	1	0	1	1	1	0	3	371.62	324.57	46.07	194.52
7	5.00	0.05	1818.35	6.90	5.41	-97.98	1849.61	14.00*	6.80	-95.93	1	0	0	1	0	0	3	393.89	361.02	31.89	208.15
7	7.00	10.00	1835.67	6.75	6.41	-98.02	1862.43	21.00*	7.55	-93.90	2	0	0	1	0	0	3	318.86	290.38	27.50	143.86

* Branch-and-bound stopped at first integer solution.

Table 4.5: Variation of o and t of cost constraint approach for first officer scenario, summer 2005.

Chapter 5

Robust and Integrated Aircraft Routing and Crew Pairing

In this chapter we formulate a model that integrates the two problems aircraft routing and crew pairing. Integrating the schedule design problem is addressed in the subsequent chapter. We expect the largest gain in cost and robustness by integrating these three problems compared to the integration of other airline scheduling problems. We do not consider the problems fleet assignment or crew rostering in the integrated approach. The fleet assignment problem is important for large airlines with many different fleet types. For the problem instances considered in the context of this thesis the fleet can be regarded as homogeneous and the fleet assignment problem can be omitted. Note that we do consider a basic fleet assignment model by including the OverWater rule. We outline below how this can be generalised to consider different fleet types in our solution approaches. We do not include the crew rostering problem in our formulation. For the relevant scenarios, the crew rostering problem can be viewed as a separate optimisation problem. The main objective in rostering is maximising crew satisfaction rather than minimising cost and therefore the rostering problem has no influence on the cost of the overall airline scheduling solution [Butchers et al., 2001].

The goal of the integrated aircraft routing and crew pairing model is to generate solutions that incur low costs and are also operationally robust. We present the integrated model and describe three solution methods in the following sections. We present optimisation methods that are capable of finding optimal solutions as well as a heuristic method that finds good quality solutions quickly. We compare the approaches theoretically and conclude the chapter with a summary of the results of computational experiments.

5.1 Model

In this section we describe a model that integrates aircraft routing and crew pairing problems. In Section 4.3 we have seen that for a feasible solution, short connections are only permitted if crew stay on the same aircraft. This condition might result in suboptimal or infeasible solutions if the two problems are solved separately. If the crew pairing problem is solved for a fixed aircraft routing solution, the feasible set of connections to be used by crew is limited. But if the aircraft routing problem is solved for a fixed crew pairing solution, it may be infeasible to operate all required (short) connections with the given number of aircraft.

In order to obtain an optimal solution for the aircraft routing and crew pairing problem we need to formulate an integrated model. We enumerate all short connections that can be operated by crew and define a binary $m^B \times n^P$ matrix B^P where m^B is the number of short connections. Each pairing is associated with one column of B^P, where

$$(b_{ij})^P = \begin{cases} 1 & \text{if short connection } i \text{ is contained in pairing } j \\ 0 & \text{otherwise,} \end{cases}$$

with $1 \leq i \leq m^B, 1 \leq j \leq n^P$. For aircraft, a binary $m^B \times n^R$ matrix B^R is defined in an analogous way.

With this matrix representation the *integrated aircraft routing and crew pairing problem* can be formulated as follows:

$$\begin{array}{llcll} \text{Minimise} & (\boldsymbol{c}^P)^T \boldsymbol{x}^P & + & (\boldsymbol{c}^R)^T \boldsymbol{x}^R & \\ \text{subject to} & A^P \boldsymbol{x}^P & & & = \boldsymbol{b}^P \\ & & & A^R \boldsymbol{x}^R & = \mathbb{1} \\ & B^P \boldsymbol{x}^P & - & B^R \boldsymbol{x}^R & \leq 0, \end{array} \tag{5.1}$$

where $\boldsymbol{x}^P \in \{0,1\}^{n^P}$ and $\boldsymbol{x}^R \in \{0,1\}^{n^R}$ are binary variables. The first two sets of constraints are identical to the original single problem formulations. The third set of constraints ensures that short connections which are operated by some crew are also operated by some aircraft.

Since we are not only interested in minimal cost but also robust solutions we need to minimise the number of restricted connections in the solution (see Section 4.3). If the two problems are solved in sequence, the overall solution can be suboptimal, i.e. another solution may exist with equal or less cost that contains fewer restricted aircraft changes. It is possible to improve both objective simultaneously compared to a solution of a sequential approach. In a sequential approach the solution space of the problem solve last, and hence the overall solution space, is limited by the problem solved first, leading to globally suboptimal solutions. In order to integrate restricted connections into our formulation we enumerate all restricted connections. Analogously to short connections, we define a binary $m^D \times n^P$ matrix D^P where m^D is the number of restricted connections:

$$(d_{ij})^P = \begin{cases} 1 & \text{if restricted connection } i \text{ is contained in pairing } j \\ 0 & \text{otherwise,} \end{cases}$$

with $1 \leq i \leq m^D, 1 \leq j \leq n^P$. For aircraft, a binary $m^D \times n^R$ matrix D^R is defined in an analogous way.

With this matrix representation the *robust and integrated aircraft routing and crew pairing problem* [see also Mercier et al., 2005] can be formulated as follows:

$$\begin{array}{llllllll} \text{Minimise} & (\boldsymbol{c}^P)^T\boldsymbol{x}^P & + & (\boldsymbol{c}^R)^T\boldsymbol{x}^R & + & p(\boldsymbol{c}^{AC})^T\boldsymbol{d} & & \\ \text{subject to} & A^P\boldsymbol{x}^P & & & & & = & \boldsymbol{b}^P \\ & & & A^R\boldsymbol{x}^R & & & = & \mathbb{1} \\ & B^P\boldsymbol{x}^P & - & B^R\boldsymbol{x}^R & & & \leq & 0 \\ & D^P\boldsymbol{x}^P & - & D^R\boldsymbol{x}^R & - & \boldsymbol{d} & \leq & 0, \end{array} \tag{5.2}$$

where $\boldsymbol{x}^P \in \{0,1\}^{n^P}$, $\boldsymbol{x}^R \in \{0,1\}^{n^R}$, and $\boldsymbol{d} \in \{0,1\}^{m^D}$ are binary variables, $\boldsymbol{c}^{AC} \in \mathbb{R}_+^{m^D}$ are positive penalties for changing aircraft, and value $p \in \mathbb{R}_+$ is a weight to adjust the scale of the aircraft change cost compared to crew pairing and aircraft routing costs. Variable d_i equals 1 if restricted connection i is operated by a crew but no aircraft and 0 otherwise. The first three sets of constraints are identical to problem (5.1). The last set of constraints provokes additional aircraft change cost in the objective function if a restricted connection is operated by a crew but not by an aircraft.

The model yields an optimal solution for given aircraft change penalties $\boldsymbol{c}^{AC}$. The model assumes that the DPACLim rule is relaxed. We describe below how the DPACLim rule can be considered in each solution approach.

For the schedule data sets considered in this work, it can be assumed that the minimal sit-time of the crew is equal to the minimal turn-time of aircraft for all connections. Hence no short connections are taken into account and we remove the short connection constraints from the model. Short connections could be treated in a similar way to restricted connections. Instead of penalties $\boldsymbol{c}^{AC}$, we could use very large penalties, effectively forbidding short aircraft changes in any solution.

Aircraft routings and crew pairings must obey the rules listed in Sections 3.2 and 4.2, respectively.

5.2 Solution Methods

In this section we describe two new solution methods for the robust and integrated aircraft routing and crew pairing model: the iterative approach which is an optimisation based heuristic approach and a Dantzig-Wolfe decomposition approach which is an optimisation method. We also describe the currently most successful solution method in the literature which is a Benders decomposition approach as in Mercier et al. [2005]. We compare the characteristics of the different methods.

As described in the introduction, the goal of this thesis is to solve the robust and integrated aircraft routing and crew pairing problem without potentially damaging the set partitioning structures of the individual problems. The structure of the problems can therefore still be exploited to solve the problems efficiently. From a practical point of view, only minimal changes are required for existing crew pairing and aircraft routing solvers to be incorporated in such an approach.

There are five existing solution approaches for the integrated aircraft routing and crew pairing problem (see Section 2.2):

1. Direct solution method [Cordeau et al., 2001]
2. Benders decomposition with aircraft routing as the master problem [Cordeau et al., 2001]
3. Plane-count method [Klabjan et al., 2002]
4. Extended crew pairing method [Cohn and Barnhart, 2003]
5. Benders decomposition with crew pairing as the master problem [Mercier et al., 2005]

Cordeau et al. [2001] show that their Benders decomposition approach is superior to a direct solution method for the integrated model. Mercier et al. [2005] in turn show that the Benders decomposition approach with the crew pairing problem as the master problem is superior to the approach with the aircraft routing as the master problem [Cordeau et al., 2001]. In the plane-count constraint approach by Klabjan et al. [2002] feasibility of the aircraft routing problem cannot be guaranteed. Since we consider aircraft routing costs (see Section 3.2) we cannot find an optimal solution with this approach. Cohn and Barnhart [2003] propose to extend the crew pairing problem with an additional aircraft routing column generation problem but Mercier et al. [2005] show that this approach is computationally expensive. This leaves the Benders decomposition approach by Mercier et al. [2005] as the best approach in the literature to solve the integrated aircraft routing and crew pairing problem.

In all these approaches inequalities are added to the original set partitioning formulations of the aircraft routing or crew pairing problem. From our experience, adding base-constraints and the cost constraint to the crew pairing problem, greatly increases the complexity of the problem (see Section 4). Without these constraints the problems have “almost” integer properties. The LP/IP gaps are usually very small and integer solutions can be obtained quickly with a branch-and-bound method. After adding the constraints the gaps can be substantial and it can be very difficult to find an integer solution.

For these reasons we propose two solution methods that do not add additional constraints to the set partitioning formulation, the iterative approach and a Dantzig-Wolfe decomposition approach. Lagrange decomposition could also be used as a solution method but is not considered in this thesis. Cordeau et al. [2001] believe that Benders decomposition is superior to Lagrange relaxation since their approach is very fast. However, Lagrange relaxation could be investigated as a further approach that does not alter the structures of the subproblems.

5.2.1 Iterative Approach

In this section we describe an optimisation based heuristic solution method for the robust and integrated aircraft routing and crew pairing problem. The two individual problems are alternately solved to optimality. Each problem receives input from the previously solved problem. This process continues until a stopping criterion is reached. A predefined solution quality cannot be guaranteed but a lower bound for the optimal solution value is provided so that the quality is known once the algorithm terminates.

We assume that MinTurnSeq costs are the only aircraft routing costs and the majority of the costs of the integrated solution are crew pairing costs. In the following we denote the sum of crew pairing costs and MinTurnSeq costs simply by costs of the integrated solution c^{INT}. We consider a connection to be restricted, if the sit-time does not exceed the minimal sit-time by more than 30 minutes. The aircraft change cost c^{AC} of an integrated solution is the sum over all restricted aircraft changes (see Section 4.3):

$$c^{AC} = \sum_{ij \in \overline{RC}(x^R)} c_{ij}^{AC} \sum_{k, ij \in k} \boldsymbol{x}_k^P, \tag{5.3}$$

where $\overline{RC}(\boldsymbol{x}^R)$ is the set of restricted connections induced by the aircraft routing solution $\boldsymbol{x}^R$. Value k is used to index crew pairings and $ij \in k$ is used to indicate that connection ij is used in crew pairing k. The smaller the aircraft change cost c^{AC} of a solution the more robust we expect the solution to be.

We search for an integrated solution with small cost c^{INT} and small aircraft change cost c^{AC}. We do not attempt to solve the integrated problem to optimality. Instead we propose to solve the crew pairing problem and the aircraft routing problem iteratively. Initially, we solve the crew pairing problem to cost optimality without considering any aircraft routings. This results in the generation of a larger set of feasible crew pairings since feasibility parameters are relaxed, treating every connection in the crew pairing problem as a follow-on connection. Since this reduces the minimal sit-time on these connections there exist many more feasible arcs. The solution is likely to be infeasible for any aircraft routing solution. This initial solution yields a lower bound on the crew pairing cost of a feasible integrated solution. Then, in each iteration the aircraft routing problem is solved first. We consider all restricted connections operated in the current crew pairing solution and force the aircraft routing solution to contain as many of those connections as possible. This will force the "aircraft to follow the crew" as much as possible if the connection is restricted. In other words, we solve the aircraft routing problem using the following objective function:

$$\text{Minimise } (\boldsymbol{c}^R)^T \boldsymbol{x}^R - \sum_{ij \in RC(x^P)} c_{ij}^{AC} \sum_{k, ij \in k} \boldsymbol{x}_k^R, \tag{5.4}$$

where $RC(\boldsymbol{x}^P)$ is the set of restricted connections operated in the current crew pairing problem solution. Vectors $\boldsymbol{c}^R$ and $\boldsymbol{x}^R$ are defined as in (3.1). The first term of objective function (5.4) minimises the number of consecutive minimal turns and the second term maximises the number of restricted connections that are operated by crew in the aircraft routing solution. This is in contrast to the aircraft change cost (5.3) where we minimise the number of restricted aircraft changes. Next we solve the crew pairing problem to optimality for the current aircraft routing solution with a weighted sum objective function of crew pairing costs and aircraft change costs:

$$\text{Minimise } (\boldsymbol{c}^P)^T \boldsymbol{x}^P + pc^{AC}, \tag{5.5}$$

where $\boldsymbol{c}^P, \boldsymbol{x}^P, p$, are defined as in (5.2) and c^{AC} is defined as in (5.3). The solutions of the two problems solved in each iteration yield a feasible solution to the integrated problem. We start with penalty p equal to 0 and increase the penalty in each iteration in order to increase the robustness of the solutions we generate. Note that we do not change the ratio of weights between costs

Algorithm 2 Iterative Algorithm

```
1: set $p = 0$
2: solve crew pairing problem with objective function (5.5){Since no aircraft
   routings are taken into account a larger set of feasible pairings is gener-
   ated.}
3: while $p \leq p_{max}$ do
4:     solve aircraft routing problem with objective function (5.4){Minimise
       cost and maximise the number of restricted connections contained in the
       aircraft routing solution that are operated in the current crew pairing
       solution.}
5:     solve crew pairing problem with objective function (5.5){Minimise cost
       and the number of restricted aircraft changes.}
6:     break if robustness cannot be improved
7:     increase $p$
8: end while
```

$\boldsymbol{c}^R$ and $\sum_{ij \in RC(x^P)} c_{ij}^{AC}$ in the aircraft routing problem. Here the ratio is set to reflect the importance of the two robustness measures aircraft change cost ($\sum_{ij \in RC(x^P)} c_{ij}^{AC}$) and consecutive minimal turns ($\boldsymbol{c}^R$) and there exists no trade-off with a monetary cost objective as in the crew pairing problem.

Algorithm 2 shows the steps of the iterative approach, see also Figure 5.1 for a schematic overview. For the schedules we consider the minimal sit-time is equal to the minimal turn-time for all connections, and, hence, no short connections are taken into account. Since we always solve the crew pairing problem for a given solution of the aircraft routing problem, short connections can be considered by removing connections in the underlying network of the crew pairing problem. If short connections are present in the problem, e.g. in problem instances of American or European airlines, Step 2 of Algorithm 2 generally yields an infeasible solution that violates the short connection rules.

For our problem instances the interdependence between aircraft routings and crew pairings stated above is extended by the DPACLim rule. Since the crew pairing problem is solved for a given aircraft routing solution, this rule can simply be embedded in the resource constrained shortest path algorithm of the crew pairing problem (see Section 4).

The cost of the crew pairing solution in Step 2 yields a lower bound on the crew pairing costs of a feasible integrated solution since no aircraft routings are taken into account. In our experiments, no aircraft routing solution for

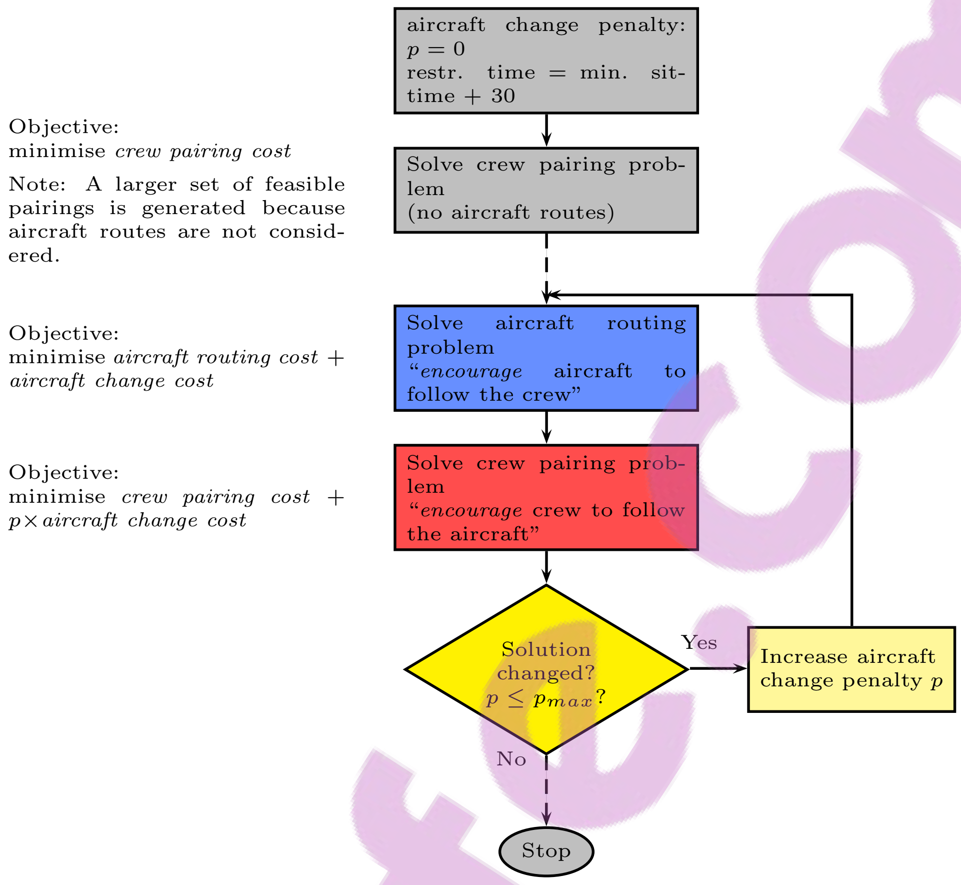

aircraft change penalty: $p = 0$
restr. time = min. sit-time + 30
Objective: minimise *crew pairing cost*
Note: A larger set of feasible pairings is generated because aircraft routes are not considered.
Solve crew pairing problem (no aircraft routes)
Objective: minimise *aircraft routing cost* + *aircraft change cost*
Solve aircraft routing problem "*encourage* aircraft to follow the crew"
Objective: minimise *crew pairing cost* + $p\times$ *aircraft change cost*
Solve crew pairing problem "*encourage* crew to follow the aircraft"
Solution changed? $p \leq p_{max}$?
Yes
Increase aircraft change penalty p
No
Stop

Figure 5.1. Schematic view iterative approach.

the crew pairing solution of Step 2 could be found to satisfy the DPACLim rule. Hence, Step 2 usually yields an infeasible crew pairing solution to the integrated problem. If the DPACLim rule is relaxed we can find an aircraft routing solution to form a feasible solution to the integrated problem. For the test instances, such an integrated solution contains a large number of restricted aircraft changes and hence accounts for large aircraft change costs.

After the initial steps of the algorithm, we obtain a feasible solution to the integrated problem in each iteration by solving the crew pairing problem (Step 5) for a given aircraft routing solution (Step 4). Once the integrated solution converges to a stable solution, i.e. all successive iterations yield identical aircraft routing and crew pairing solutions, the algorithm stops. Hence, the aircraft change cost cannot be improved. The value of p_{max} is chosen such that the aircraft change costs dominate the crew pairing costs in function (5.5) in the sense that no restricted aircraft changes are contained in the optimal solution if such a solution exists. The sequence of values p we use in this thesis is $p = \{0, 2, 5, 10, 20, 50, 100, 500, 1000\}$. In practice, we stop the algorithm once the aircraft change costs are below some threshold, e.g. less than 10.

Once the algorithm terminates, a number of different solutions to the robust and integrated aircraft routing and crew pairing problem are provided. The trade-off between cost and robustness varies between solutions and the airline can choose which solution to operate (see Figure 5.3 below).

Implementation

Note that the only modification of the aircraft routing and crew pairing algorithms presented above are different costs in the objective functions. Hence, the modifications can easily be applied to existing aircraft routing and crew pairing solvers an airline may possess. In each iteration the costs are assigned to the appropriate arcs of the column generation networks.

Non-linear Programming Formulation

It is noteworthy that characteristics of the iterative approach are similar to solution approaches in non-linear programming. Model (5.2) without the short connection constraints is equivalent to the following non-linear non-convex integer optimisation problem:

$$
\begin{array}{lllllll}
\text{Minimise} & (\boldsymbol{c}^P)^T \boldsymbol{x}^P & + & (\boldsymbol{c}^R)^T \boldsymbol{x}^R & + & p \displaystyle\sum_{ij \in RC} \left(\sum_{k, ij \in k} \boldsymbol{x}_k^P \right) \left(1 - \sum_{k, ij \in k} \boldsymbol{x}_k^R \right) & \\
\text{subject to} & A^P \boldsymbol{x}^P & & & & = & \boldsymbol{b}^P \\
 & & & A^R \boldsymbol{x}^R & & = & \mathbb{1},
\end{array}
$$

where RC denotes the set of all possible restricted connections. Formulation (5.6) can be solved by sequential linearisation methods, solving a sequence of linear approximations. We refer to Arora et al. [1994] and Li and Sun [2006] for details on non-linear programming and sequential LP solution methods. It is an interesting topic of future research to compare the performance of such a solution method with the methods proposed in this thesis.

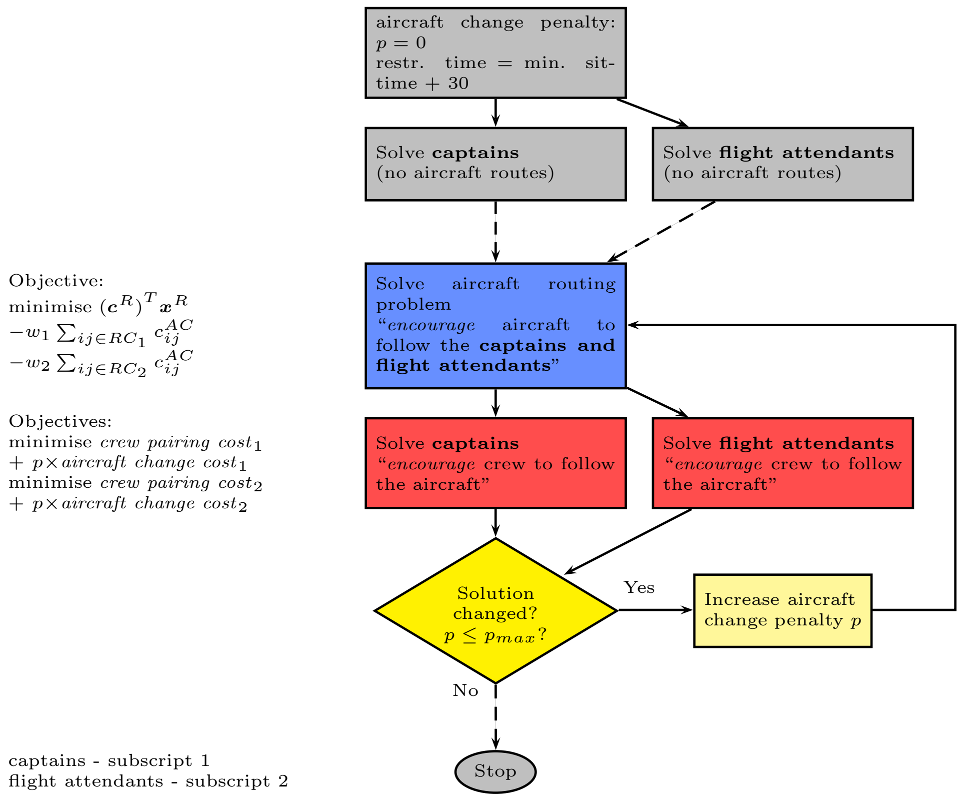

aircraft change penalty: $p = 0$
restr. time = min. sit-time + 30
Solve **captains** (no aircraft routes)
Solve **flight attendants** (no aircraft routes)
Objective:
minimise $(\boldsymbol{c}^R)^T \boldsymbol{x}^R - w_1 \sum_{ij \in RC_1} c_{ij}^{AC} - w_2 \sum_{ij \in RC_2} c_{ij}^{AC}$
Solve aircraft routing problem "*encourage* aircraft to follow the **captains and flight attendants**"
Objectives:
minimise *crew pairing cost*$_1$ + $p\times$*aircraft change cost*$_1$
minimise *crew pairing cost*$_2$ + $p\times$*aircraft change cost*$_2$
Solve **captains** "*encourage* crew to follow the aircraft"
Solve **flight attendants** "*encourage* crew to follow the aircraft"
Solution changed? $p \leq p_{max}$?
Yes
Increase aircraft change penalty p
No
Stop
captains - subscript 1
flight attendants - subscript 2

Figure 5.2. Schematic view iterative approach with two crew groups.

Iterative Approach for Multiple Crew Groups

The crew for an aircraft usually consists of multiple crew groups. Our data sets consist of three crew groups, namely captains and first officers (technical crew) and flight attendants (cabin crew). Different rules, base strengths and pay structures apply to each group. A robust aircraft routing and crew pairing solution for one crew group may enforce many restricted aircraft changes in a crew pairing solution for another crew group. Hence, considering aircraft and one crew group might lead to a suboptimal solution. Ideally we want to consider all crew groups and aircraft simultaneously. To incorporate multiple crew groups into the iterative approach we simply solve the crew pairing problem in Steps 2 and 5 for each crew group separately but use the same common aircraft routing solution. To obtain penalties for the restricted connections for the subsequent aircraft routing problem we scale the penalties for the restricted connections of the different crew pairing solutions according to weights (w_1 and w_2 in objective function (5.6) in an example with two crew groups) chosen by the airline. The aircraft routing objective function in Step 4 for two crew groups is changed to:

$$\text{Minimise } (\boldsymbol{c}^R)^T \boldsymbol{x}^R - w_1 \sum_{ij \in RC(x^P_{G1})} c^{AC}_{ij} \sum_{k, ij \in k} \boldsymbol{x}^R_k - w_2 \sum_{ij \in RC(x^P_{G2})} c^{AC}_{ij} \sum_{k, ij \in k} \boldsymbol{x}^R_k, \tag{5.6}$$

where $RC(\boldsymbol{x}^P_{G1}), RC(\boldsymbol{x}^P_{G2})$ are the restricted connections contained in crew pairing solutions of crew group 1 ($G1$) and crew group 2 ($G2$), respectively. Indices k and ij are defined as in Section 5.2.1. Non-negative weights w_1 and w_2 can be chosen to reflect the ratio of crew pairing costs between both crew groups. We then generate a new aircraft routing solution as before. The results of the *iterative algorithm with multiple crew groups* are presented in Section 5.3.3. More than two crew groups can be considered in a straightforward extension. Figure 5.2 shows a schematic overview of the iterative algorithm with two crew groups, captains and flight attendants.

Iterative Approach for Multiple Aircraft Types

Since we use a path based formulation for the aircraft routing problem, multiple fleet types can be considered in the aircraft routing problem in a straightforward way. If crew can only operate a subset of fleet types (which is common for pilots for example), the subsequent crew pairing problems are solved over subsets of flights determined by the aircraft routing problem. With this strategy the iterative approach partially integrates fleet assignment, aircraft routing, and crew pairing problems.

5.2.2 Dantzig-Wolfe Decomposition Approach

First we repeat the formulation of the robust and integrated aircraft routing and crew pairing model we try to solve:

$$\begin{array}{lccccccl}
\text{Minimise} & (\boldsymbol{c}^P)^T \boldsymbol{x}^P & + & (\boldsymbol{c}^R)^T \boldsymbol{x}^R & + & p(\boldsymbol{c}^{AC})^T \boldsymbol{d} & & \\
\text{subject to} & A^P \boldsymbol{x}^P & & & & & = & \boldsymbol{b}^P \\
 & & & A^R \boldsymbol{x}^R & & & = & \mathbb{1} \\
 & D^P \boldsymbol{x}^P & - & D^R \boldsymbol{x}^R & - & \boldsymbol{d} & \leq & 0.
\end{array} \tag{5.7}$$

Here we omit the short connection constraints because the scenarios we consider do not contain any short connections as explained in Section 5.2.1. The third set of constraints of formulation (5.2) can be considered in a similar way to the restricted connection constraints, as illustrated in the following.

We use Dantzig-Wolfe decomposition (see Section 1.5.1) to re-formulate (5.7) as one master problem and two sub-problems. The only constraints of the original formulation that are present in the master problem are the restricted connection constraints:

$$\begin{array}{lrcrcrcl} \text{Minimise} & (\boldsymbol{c}^P)^T V^P \boldsymbol{\lambda} & + & (\boldsymbol{c}^R)^T V^R \boldsymbol{\mu} & + & p(\boldsymbol{c}^{AC})^T \boldsymbol{d} & & & \\ \text{subject to} & \mathbb{1}^T \boldsymbol{\lambda} & & & & & = & 1 & \rightarrow \pi^P \\ & & & \mathbb{1}^T \boldsymbol{\mu} & & & = & 1 & \rightarrow \pi^R \\ & D^P V^P \boldsymbol{\lambda} & - & D^R V^R \boldsymbol{\mu} & - & \boldsymbol{d} & \leq & 0 & \rightarrow \boldsymbol{\pi}, \end{array} \tag{5.8}$$

where $\boldsymbol{\lambda} \in \{0,1\}^{|V^P|}$, $\boldsymbol{\mu} \in \{0,1\}^{|V^R|}$ and $\boldsymbol{d} \in \{0,1\}^{m^D}$. The columns $\boldsymbol{v}_i^P$ and $\boldsymbol{v}_i^R$ of matrices $V^P = [\boldsymbol{v}_1^P, \boldsymbol{v}_2^P, \cdots, \boldsymbol{v}_k^P]$ and $V^R = [\boldsymbol{v}_1^R, \boldsymbol{v}_2^R, \cdots, \boldsymbol{v}_k^R]$ span the respective polyhedra $P^P = \{\boldsymbol{x}^P \in \mathbb{R}_+^n | A^P \boldsymbol{x}^P = \boldsymbol{b}^P\}$, $P^P =$ conv $(\{\boldsymbol{v}_1^P, \ldots, \boldsymbol{v}_k^P\})$ and $P^R = \{\boldsymbol{x}^R \in \mathbb{R}_+^n | A^R \boldsymbol{x}^R = \mathbb{1}\}$, $P^R =$ conv $(\{\boldsymbol{v}_1^R, \ldots, \boldsymbol{v}_k^R\})$. Dual values π^P and π^R are associated with crew and aircraft convexity constraints, respectively. The entries of vector $\boldsymbol{\pi}$ are the dual values corresponding to the restricted connection constraints. The convexity constraints ensure that exactly one aircraft routing solution and exactly one crew pairing solution is chosen in an optimal integer solution.

The two subproblems contain all other constraints of the original formulation. The crew pairing subproblem is identical to the original crew pairing problem except for the objective function:

$$\begin{array}{llrcl} \text{Minimise} & ((\boldsymbol{c}^P)^T - \boldsymbol{\pi}^T D^P)\boldsymbol{x}^P & & & \\ \text{subject to} & & A^P \boldsymbol{x}^P & = & \boldsymbol{b}^P \\ & & \boldsymbol{x}^P & \in & \{0,1\}^{n^P}. \end{array} \tag{5.9}$$

It is important to note that this crew pairing subproblem assumes no aircraft routing solution. All connections are follow-on connections and minimal sit-time rules are relaxed. Similarly, the aircraft routing subproblem is identical to the original aircraft routing problem except for the objective function:

$$\begin{array}{lrcl} \text{Minimise} & ((\boldsymbol{c}^R)^T + \boldsymbol{\pi}^T D^R)\boldsymbol{x}^R & & \\ \text{subject to} & A^R\boldsymbol{x}^R & = & \mathbb{1} \\ & \boldsymbol{x}^R & \in & \{0,1\}^{n^R}. \end{array} \tag{5.10}$$

The solution process works as described in Section 1.5.1. The LP relaxation of the restricted master problem is solved and the optimal dual values are passed as input to both subproblems. These are solved and one column is generated from each subproblem solution and added to V^P or V^R of the master problem, respectively, if the reduced costs are negative. The process iterates until no columns with negative reduced cost are returned by either subproblem or a specified optimality gap is reached.

In each iteration we solve both subproblems and generate one column for V^P and one column for V^R from these solutions. Note that the subproblems are not necessarily solved to IP optimality but stopped when a specified LP bound-gap is achieved. In a first phase the master problem is only solved to LP optimality. During this phase an optimality gap can be obtained from the LP optimal solution values of the subproblems. Since the right-hand-side $\boldsymbol{b}_2$ of the second set of constraints of formulation (1.7) is equal to 0, a lower bound for an optimal LP solution of (5.7) is provided by the sum of the costs of optimal LP solutions of the two subproblems (see last paragraph of Section 1.5.1). We can get an improved lower bound for the optimal solution value of (5.7) if we solve the subproblems to IP optimality.

If fractional solution variables are contained in the optimal solution of the linear relaxation of (5.7) we branch on the restricted connections to obtain an integer solution. Since each restricted connection is associated with a connection arc i in the flight network, the branching decisions are easily incorporated into the subproblems by forcing or banning arcs to be contained in a solution. We can stop the algorithm after the LP solution phase (or at any time) prematurely and determine the best integer solution found by solving the master problem to IP optimality over the columns generated so far.

In our computational experiments we stop the algorithm after the linear relaxation of (5.7) is solved to optimality or within a specified optimality gap.

We do not use the Dantzig-Wolfe decomposition approach to obtain integer solutions to (5.7) since the integer solutions obtained by iterative approach are usually within the specified bound-gap of 2% of the LP solution. Hence, there is no additional benefit of running the IP solution phase. We refer to the computational experiments (Section 5.3) for more details.

Note that we do not include the DPACLim rule in this solution approach. The consideration of this rule requires linking particular routings and pairings which could be enforced by additional constraints or a branching strategy. This, however, is computationally difficult and inefficient to enforce. For the iterative approach on the other hand, enforcing the rule is very easy. Since the crew pairing problem is always solved for a given aircraft routing problem, the DPACLim rule can be enforced during the shortest path calculation.

We expect that an operationally robust solution will "almost" satisfy the DPACLim rule. Since the DPACLim is an artificial rule to enforce robustness (see Section 4.3), a slight violation of the rule can be tolerated. We can enforce the rule heuristically by using the aircraft routing solution of the optimal integrated IP solution and generating a DPACLim rule feasible crew pairing solution as in the iterative approach.

Implementation

Similar to the iterative approach, only small changes are required to existing aircraft routing and crew pairing algorithms. Again, in each iteration, costs are assigned to the appropriate arcs of the column generation networks. We also modify the crew pairing algorithm so that each connection is treated as a follow-on connection, i.e. minimal sit-times are relaxed. Note that the rows of matrices D^R and D^P are not generated a priori for all connections. Instead, the rows are populated during the algorithm for connections that are part of a solution that is returned by the crew pairing subproblem.

5.2.3 Benders Decomposition Approach

Currently, the most successful approach in the literature to solve the robust and integrated aircraft routing and crew pairing problem seems to be Benders decomposition (see Mercier et al. [2005]). In this section we outline their solution approach. We again consider problem (5.7) where short connection constraints are omitted. Such constraints can be considered in a similar way to restricted connection constraints.

First, for a given LP solution $\overline{\boldsymbol{x}}^P \in \{\boldsymbol{x}^P : A^P\boldsymbol{x}^P = \boldsymbol{b}^P, \boldsymbol{x}^P \geq 0\}$ of the crew pairing problem, the LP relaxation of (5.7) reduces to a primal subproblem that contains only aircraft routing variables:

$$\begin{array}{lllll} \text{Minimise} & (\boldsymbol{c}^R)^T\boldsymbol{x}^R & + & p(\boldsymbol{c}^{AC})^T\boldsymbol{d} & & \\ \text{subject to} & A^R\boldsymbol{x}^R & & & = & \mathbb{1} \\ & D^R\boldsymbol{x}^R & + & \boldsymbol{d} & \geq & D^P\overline{\boldsymbol{x}}^P, \end{array} \tag{5.11}$$

with $\boldsymbol{x}^R \geq 0$ and $\boldsymbol{d} \geq 0$. Note that the primal subproblem is always feasible if a feasible aircraft routing solution exists. Otherwise (e.g. if short connection constraints are included), feasibility can be achieved by adding artificial variables with large costs. Next, we formulate the dual of the primal subproblem:

$$\begin{array}{llllll} \text{Maximise} & \boldsymbol{\alpha}^T\mathbb{1} & + & \boldsymbol{\beta}^T D^P\overline{\boldsymbol{x}}^P & & \\ \text{subject to} & \boldsymbol{\alpha}^T A^R & + & \boldsymbol{\beta}^T D^R & \leq & \boldsymbol{c}^R \\ & & & \boldsymbol{\beta} & \leq & p\boldsymbol{c}^{AC}, \end{array} \tag{5.12}$$

with $\boldsymbol{\beta} \geq 0$ and dimensions of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ as appropriate. Since $\boldsymbol{\alpha} = 0$ and $\boldsymbol{\beta} = 0$ is a feasible solution for the dual subproblem, both primal and dual subproblems have bounded and feasible solutions. Let Δ denote the polyhedron defined by the constraints of (5.12) and let P_Δ be the set of extreme points of Δ. The LP relaxation of (5.7) can be reformulated as a master problem containing only crew pairing variables:

$$\begin{array}{llllll} \text{Minimise} & z & + & (\boldsymbol{c}^P)^T\boldsymbol{x}^P & & \\ \text{subject to} & & & A^P\boldsymbol{x}^P & = & \boldsymbol{b}^P \\ & z & - & \boldsymbol{\beta}^T D^P\boldsymbol{x}^P & \geq & \boldsymbol{\alpha}^T\mathbb{1} \qquad (\boldsymbol{\alpha},\boldsymbol{\beta}) \in P_\Delta \\ & & & \boldsymbol{x}^P & \geq & 0. \end{array} \tag{5.13}$$

The free variable z is restricted to be larger than the optimal value of the dual subproblem for any $\overline{\boldsymbol{x}}^P$. In general, the master problem contains more constraints than formulation (5.7). As the variables in the Dantzig-Wolfe decomposition, these constraints are not enumerated a priori. Instead, an initially empty restricted master problem is solved and the optimal solution is used as input to the primal subproblem. If the optimal solution value of the subproblem is larger than the value of z, the constraint formed by the dual variables of the optimal subproblem solution is violated by the current solution of the master problem. It is therefore added to the master problem and the master problem is re-solved. If the objective value of the subproblem is equal to (or within a specified gap of) z, the algorithm stops with a (close to) optimal solution for the LP relaxation of problem (5.7).

To obtain integer solutions to problem (5.7), the Benders decomposition is embedded in a 3 phase approach by Mercier et al. [2005]. In the first phase all integer requirements are dropped and master problem and subproblem are solved to LP optimality with the use of column generation. The first phase of the algorithm can be stopped when the gap between lower and upper bound of the LP solution is sufficiently small. In the second phase the master problem is solved to integer optimality with a branch-and-bound method and the subproblem is solved to LP optimality at each node. In the final phase integrality conditions are enforced on the subproblem and this is solved once. Since the subproblem is always feasible, the algorithm stops with a heuristic integer solution. However, the objective value of this solution may violate the cost constraint of the master problem. If the master problem contains not only optimality constraints for restricted connections, but also feasibility constraints for short connections, the heuristic integer solution may in fact be infeasible. In this case a constraint must be added to the master problem forbidding the set of short connections. With this new constraint the problem must be re-solved starting from phase 2.

Implementation

As in the Dantzig-Wolfe decomposition, the DPACLim rule is relaxed in the Benders decomposition approach. The following changes to the algorithms that solve the individual problems are required. An additional set of constraints must be added to the aircraft routing problem. On each iteration the right-hand sides of these constraints must be changed. On each iteration a new constraint is added to the crew pairing problem. Finally, the variable z must be added to the crew pairing model.

Identifying Strong Cuts

Mercier et al. [2005] improve their algorithm by modifying the subproblem to generate strong cuts. If the primal subproblem is degenerate, more than one optimal solution to the dual subproblem may exist. Although all of these solutions generate valid cuts, some may be stronger than others. A cut generated from the optimal solution $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ dominates another cut generated from solution $(\boldsymbol{\alpha}', \boldsymbol{\beta}')$ if and only if $\boldsymbol{\alpha}^T\mathbb{1} + \boldsymbol{\beta}^T D^P \boldsymbol{x}^P \geq \boldsymbol{\alpha}'^T\mathbb{1} + \boldsymbol{\beta}'^T D^P \boldsymbol{x}^P$ for all $\boldsymbol{x}^P \in \{\boldsymbol{x}^P : A^P\boldsymbol{x}^P = \boldsymbol{b}^P, \boldsymbol{x}^P \geq 0\}$. A cut that is not dominated is called *Pareto optimal*. The authors solve a dual auxiliary subproblem to obtain a Pareto optimal cut:

$$
\begin{array}{llcll}
\text{Maximise} & \boldsymbol{\alpha}^T\mathbb{1} & + & \boldsymbol{\beta}^T D^P \boldsymbol{x}_0^P & \\
\text{subject to} & \boldsymbol{\alpha}^T\mathbb{1} & + & \boldsymbol{\beta}^T D^P \overline{\boldsymbol{x}}^P & = v(\overline{\boldsymbol{x}}^P) \\
 & \boldsymbol{\alpha}^T A^R & + & \boldsymbol{\beta}^T D^R & \leq \boldsymbol{c}^R \\
 & & & \boldsymbol{\beta} & \leq p\boldsymbol{c}^{AC},
\end{array}
\tag{5.14}
$$

with $\boldsymbol{\beta} \geq 0$. Vector $\boldsymbol{x}_0^P$ is chosen in the relative interior $ri(\mathbb{X}^{LP})$ of $\mathbb{X}^{LP} = \{\boldsymbol{x}^P : A^P\boldsymbol{x}^P = \boldsymbol{b}^P, \boldsymbol{x}^P \geq 0\}$ (see below how to choose such an $\boldsymbol{x}_0^P$). Vector $\overline{\boldsymbol{x}}^P \in \mathbb{X}^{LP}$ is a given solution for which the primal subproblem is feasible and value $v(\overline{\boldsymbol{x}}^P)$ denotes the optimal value of the primal subproblem. Only the first constraint is added to the original dual subproblem (5.12) to obtain (5.14). This constraint ensures that the solution will be an extreme point of the set of optimal solutions of the original dual subproblem. In the objective function the strengths of the cuts are compared with respect to some primal feasible point $\boldsymbol{x}_0^P$. The Pareto optimal cut is added to the master problem.

One can solve the primal version of the dual auxiliary problem instead of the dual problem. Since the master problem is solved by column generation, an interior point $\boldsymbol{x}_0^P \in ri(\mathbb{X}^{LP})$ may not be available. Other points can be chosen for $\boldsymbol{x}_0^P$ and the dual auxiliary problem still yields a valid cut since the choice of $\boldsymbol{x}_0^P$ only changes the objective function. The choice may, however, affect the strength of the cut. The authors arbitrarily fix the coefficients of $\boldsymbol{\beta}$ close to 0 to generate strong cuts.

The primal auxiliary subproblem is solved to obtain a strong cut once the primal subproblem is solved in phase one of the algorithm. All other steps of the algorithm remain unchanged. We do not implement the strong cut approach. The main purpose of our implementation is to obtain a lower bound for the optimal solution value and speed of the algorithm is not very important.

5.2.4 Discussion of Approaches

In both optimisation approaches, Dantzig-Wolfe and Benders decomposition, a weight must be attached to aircraft change cost a priori. This weight represents the trade-off between costs and operational robustness and is difficult to estimate. In the iterative approach the user can choose a solution after the algorithm terminates depending on the trade-off observed between crew pairing cost and aircraft change cost, no weight is needed a priori.

All solution methods previously discussed in the literature, including Benders decomposition, add constraints to the set partitioning polytopes of the aircraft routing and crew pairing problems. These additional constraints can cause computational difficulties. In the iterative approach and the Dantzig-Wolfe decomposition approach the original set partitioning structures are not disturbed by additional constraints. Apart from the computational difficulties there are two further advantages of such an approach. Firstly, it is possible to solve aircraft routing and crew pairing problems efficiently with the methods described in Chapters 3 and 4. In both approaches only the objective function is changed to influence characteristics of the solutions. The calculation of objective function coefficients is easily implemented into the shortest path computations of the column generators for both problems. An airline usually uses aircraft routing and crew pairing solvers as part of the traditional sequential solution approach. Existing solvers can be used with only minor modifications. For both approaches only a master problem needs to be added, that controls the two subproblems. Secondly, in both approaches the aircraft routing and crew pairing problems must be solved repeatedly. The solution of a previous iteration can be used as a starting basis for the simplex algorithm. If only the objective function changes, the previous solution is still feasible and we expect that only very few iterations are needed until the new optimal solution is found.

The iterative approach and the Dantzig-Wolfe decomposition approach are structurally very similar. In both approaches identical subproblems are solved. Only additional aircraft routing information is used in the crew pairing subproblem of the iterative approach. The penalties used in the iterative approach to penalise aircraft changes can be thought of as duals π corresponding to the restricted connection constraints in the Dantzig-Wolfe master problem. This in fact gives the motivation for the iterative approach: instead of using optimal duals from an LP solution, heuristically constructed duals are used to guide the solution process of the subproblems in the iterative approach.

We think it is not possible to efficiently integrate the DPACLim rule into the Dantzig-Wolfe or Benders decomposition approaches. This would require comparing particular pairs of routings and pairings and is computationally expensive.

All three approaches provide lower bounds on the optimal solution. For the two decomposition approaches the optimal solution values of the LP relaxations of the subproblems provide lower bounds on the objective value of an optimal solution. In the iterative approach a lower bound for the crew pairing cost is calculated in the very first iteration. The minimal aircraft routing costs can be added to obtain a lower bound for the cost of an integrated solution. Benders and Dantzig-Wolfe decomposition provide a guarantee of the solution quality of the LP relaxation of problem (5.7) while the iterative approach does not.

In practice, it is beneficial to combine the iterative approach and the Dantzig-Wolfe decomposition approach. An initial solution is found by the iterative approach. The solution is added to matrices V^P and V^R of the Dantzig-Wolfe decomposition approach. This approach can then be used to obtain a lower bound for the solution and to improve the solution quality. The weight p of the iterative approach starting solution is used as weight p in the objective function of the decomposition approach. Using a starting solution can significantly speed up the solution process of the optimisation approach.

Mercier et al. [2005] show that in the Benders decomposition approach only very few iterations are required to obtain optimal LP solutions. In their computational experiments they do not consider base-constraints except for limiting the total number of duties. They also use an approximation of the crew cost function. In our experience, such relaxations greatly simplify the crew pairing problem.

Finally, FAM can easily be partially integrated into the iterative approach. For Dantzig-Wolfe and Benders decomposition methods, additional constraints are needed to ensure that crew with the correct qualifications operate on each aircraft type.

5.3 Computational Experiments

In this section we compare computational results of our implementations of the iterative approach, the Dantzig-Wolfe decomposition approach, and Benders decomposition approach. All program code is written in C, C++, and FORTRAN. We use basic implementations of Dantzig-Wolfe and Benders decomposition without any speed-up procedures. We also solve both optimisation approaches to LP optimality only and compare the results and run times with the iterative approach.

5.3.1 Iterative Approach for a Single Crew Group

Figure 5.3 displays a typical set of results of the iterative approach. As in the crew pairing chapter, the horizontal axis displays crew pairing costs and the vertical axis shows aircraft change costs. The costs of the solution operated by the airline and the solutions generated by the iterative algorithm are compared for the first officer schedule, summer 2005. The green diamond shows the objective value of the crew pairing solution that was manually generated and operated by Air New Zealand. This solution is obtained by using the aircraft routing solution that was operated by the airline and generating a cost minimal crew pairing solution with the traditional method described in Chapter 4. The lower bound for the crew pairing cost is shown which is obtained from the initial step of the iterative approach. The blue squares show the objective values of the solutions generated by the iterative approach. The labels show the iteration in which the solution is obtained. Starting with very cheap solutions the solutions become more robust during the algorithm and also more expensive. The first six solutions are all cheaper and more robust than the solution obtained by the traditional approach. These results are remarkable since Air New Zealand is using sophisticated optimisation methods for crew planning, described in detail in Butchers et al. [2001], a finalist entry for the Franz Edelman Award in 2000.

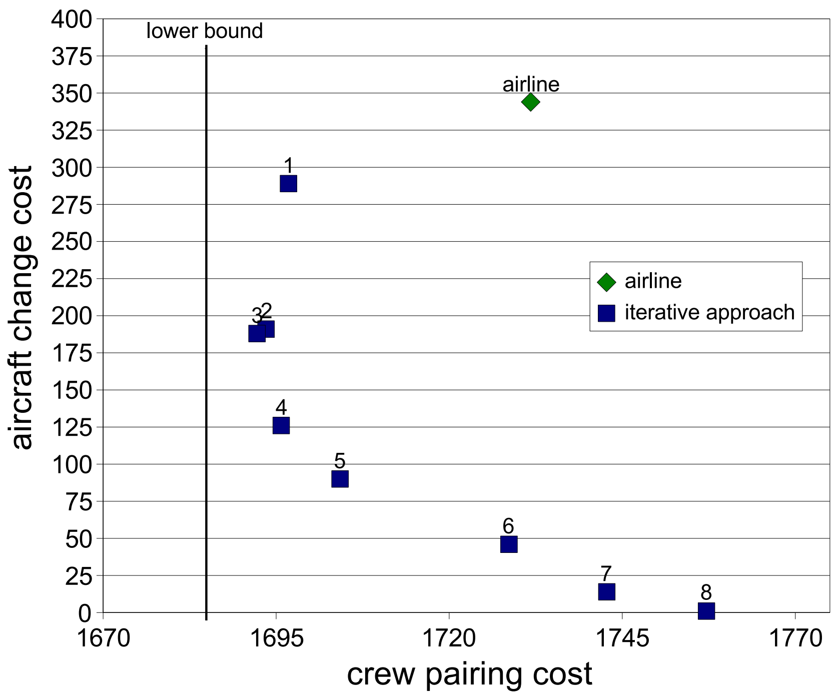


Figure 5.3. Iterative approach solutions for first officer scenario, summer 2005, 7 days.

In Table 5.1 results of the iterative approach are listed in detail for one, three, and seven day scenarios of the first officer schedule, summer 2005. The first column lists the scenario name. The next column lists the iteration in which the results are obtained. For comparison we list the solution obtained by the traditional sequential approach as "airline" (see Chapter 4). Column p shows the value of p that applies to the iteration. The next four columns show LP and IP values of crew pairing costs, the gap between LP and IP solution values ("gap") as a percentage, and the improvement of LP solutions ("impr.") compared to the sequential "airline" solution. The aircraft routing costs are displayed in column "c^R". The IP costs of aircraft changes are listed in column "c^{AC}" and the improvements compared to the "airline" solution are listed in the following column ("impr.") as a percentage. All restricted aircraft changes are listed in the following columns. Finally, the total time elapsed since the start of the algorithm is shown for each iteration in seconds. We do not show any aircraft change costs for the lower bound solution of iteration 0 since we cannot find an integrated solution satisfying the DPACLIM rule and hence this solution is infeasible.

Note that in the crew pairing problem a weighted sum objective of crew pairing costs and aircraft change costs is used. The ratio of crew pairing costs and aircraft change costs in the solution value of LP and corresponding IP solution can differ. Since we only display crew pairing costs we may observe smaller IP solution values than the corresponding LP solution values. Since the ratio also differs from iteration to iteration we observe that crew pairing costs are not strictly increasing during the iterative algorithm.

The cost of the crew pairing solution obtained in Step 2 of Algorithm 2 (iteration 0) is a lower bound for the crew pairing cost of the optimal solution value. In Table 5.1 we can observe for the 7 day first officer solutions, that this lower bound solution incurs up to 2.34% less crew pairing cost than the airline solution (LP). The solution is infeasible since we cannot find aircraft routings to satisfy the DPACLIM rule.

The cheapest feasible solution we find (iteration 1) incurs 2.32% less cost than the airline solution (LP) and its cost is almost at the lower bound (0.02% gap). Also, the aircraft change cost of this solution is only 289 compared to 344 for the airline solution which is an improvement of 15.99%. The most robust solution that is still cheaper than the airline solution improves the aircraft change cost by 86.63% (iteration 6). Note that for all technical crew DPACLIM is set to 1 for the airline approach while for the iterative algorithm DPACLIM is set to 2. Different settings are investigated below (see Figure 5.4) where we show that for setting DPACLIM to 1 the solutions of the iterative approach are also cheaper than the airline solution.

In Tables 5.2 and 5.3 we present further results for first officer schedules of winter 2005 and summer 2006, respectively. The overall trend is similar to the solutions for summer 2005 shown in Table 5.1. We find solutions that incur up to 2.23% (w05, 7 days, iter. 1) less crew pairing cost than the corresponding airline solution. For all 7 day scenarios we obtain solutions with no increase of crew pairing cost but a decrease of aircraft change cost exceeding 90% (w05, 7 days, iter. 7 and s06, 7 days, iter. 7). Aircraft routing costs c^R remain on a similar level during the iterations of the algorithm. This is caused by the constant ratio of c^R and c^{AC} in the objective function of the aircraft routing problem. Note that for the summer 2005 scenarios the aircraft routing costs are much higher for the airline solutions than for all other scenarios. For these scenarios the airline did not take the objective of minimising minimal turn sequences into account when constructing the aircraft routings.

In Tables 5.4 and 5.5 we list results for the summer 2005 schedule for captain and cabin crew groups, respectively. For the captain scenarios, results look very similar to the first officer scenarios. We achieve a decrease in crew pairing cost of up to 2.1% with a simultaneous decrease of aircraft change cost of 13.44%. We observe that for cabin crew scenarios the airline crew pairing cost is very close to the LP lower bound. The reason for this is that a relaxed DPACLim rule of 2 is used for all cabin crew scenarios in airline and iterative approach solutions. Nevertheless, we obtain a decrease of almost 60% in aircraft change cost without an increase of crew pairing cost.

Although the integer bound gap is set to 2.0% for all scenarios the average bound gap observed over all solutions of the iterative approach listed is 0.44%.

Finally, we observe that the total running time of the iterative approach for a scenario of one week ranges in between 570 seconds for the easier cabin crew problem up to 2172 seconds for the captain problem. These running times are very reasonable for the types of planning problem we try to solve. Note that the increase of weight p during the algorithm is chosen to be conservative. A faster increase of p will decrease the number of iterations and hence the running time of the algorithm considerably. We choose to generate many solutions to allow for better judgement of the trade-off between crew pairing costs and aircraft change costs.

			crew pairing cost (c^P)				aircraft	aircraft changes									
				impr.		gap	routing		impr.	(minutes exceed. min. sit-time)							run time
scenario	iteration	p	lp ($\times 10^2$)	(%)	ip ($\times 10^2$)	(%)	cost (c^R)	c^{AC}	(%)	0	5	10	15	20	25	30	elapsed (s)
s05, 1	airline	-	347.13	-	347.13	0.00	2250	34	-	2	1	1	1	0	1	3	-
	0	-	345.36	0.51	345.37	0.00	260	-	-	-	-	-	-	-	-	-	13.97
	1	0	345.80	0.38	346.37	0.16	300	60	-76.47	6	0	2	1	0	1	2	21.59
	2	2	345.96	0.34	346.34	0.11	310	27	20.59	3	0	0	0	1	1	1	26.66
	3	5	346.44	0.20	346.44	0.00	320	10	70.59	1	0	0	0	0	1	1	31.90
	4	10	346.54	0.17	346.34	-0.06	320	13	61.76	1	0	0	0	1	0	3	36.50
	5	20	346.68	0.13	346.68	0.00	270	8	76.47	1	0	0	0	0	0	1	42.20
	6	50	349.37	-0.65	346.62	-0.79	270	8	76.47	1	0	0	0	0	0	1	48.96
	7	100	349.53	-0.69	352.07	0.72	270	2	94.12	0	0	0	0	0	0	2	55.53
	8	500	353.76	-1.91	353.76	0.00	270	0	100.00	0	0	0	0	0	0	0	61.58
s05, 3	airline	-	892.95	-	900.84	0.88	4020	145	-	10	4	2	2	4	5	11	-
	0	-	883.69	1.04	893.03	1.05	580	-	-	-	-	-	-	-	-	-	62.87
	1	0	884.06	1.00	892.65	0.96	860	159	-9.66	13	2	1	6	3	4	10	106.40
	2	2	884.44	0.95	890.66	0.70	960	65	55.17	3	1	1	2	2	4	11	152.94
	3	5	885.06	0.88	897.89	1.43	920	85	41.38	8	0	0	2	2	2	11	196.10
	4	10	885.71	0.81	891.05	0.60	980	60	58.62	3	1	2	1	3	2	6	233.72
	5	20	888.06	0.55	887.63	-0.05	890	29	80.00	2	0	0	0	3	1	4	269.05
	6	50	888.89	0.45	891.75	0.32	930	13	91.03	0	0	0	0	3	1	2	304.36

			crew pairing cost (c^P)				aircraft	aircraft changes									
				impr.		gap	routing		impr.	(minutes exceed. min. sit-time)							run time
scenario	iteration	p	lp ($\times 10^2$)	(%)	ip ($\times 10^2$)	(%)	cost (c^R)	c^{AC}	(%)	0	5	10	15	20	25	30	elapsed (s)
	7	100	894.24	-0.14	899.37	0.57	930	8	94.48	0	0	0	0	1	1	3	340.20
	8	500	908.73	-1.77	910.67	0.21	880	1	99.31	0	0	0	0	0	0	1	368.17
	9	1000	914.31	-2.39	921.44	0.77	880	0	100.00	0	0	0	0	0	0	0	416.42
s05, 7	airline	-	1725.12	-	1731.77	0.38	8210	344	-	29	5	5	10	4	8	18	-
	0	-	1684.77	2.34	1702.27	1.03	1470	-	-	-	-	-	-	-	-	-	268.26
	1	0	1685.10	2.32	1696.78	0.69	2250	289	15.99	22	3	6	7	6	8	25	455.82
	2	2	1686.39	2.25	1693.53	0.42	2230	191	44.48	14	3	1	3	5	8	27	604.79
	3	5	1687.53	2.18	1692.23	0.28	2290	188	45.35	14	3	0	4	4	5	34	734.08
	4	10	1690.88	1.98	1695.73	0.29	2250	126	63.37	8	2	1	3	4	3	23	902.01
	5	20	1696.27	1.67	1704.22	0.47	2210	90	73.84	5	2	0	2	3	1	24	1052.88
	6	50	1712.21	0.75	1728.64	0.95	2190	46	86.63	1	2	1	1	1	1	13	1242.15
	7	100	1729.58	-0.26	1742.76	0.76	2140	14	95.93	0	1	0	1	0	0	4	1392.27
	8	500	1748.06	-1.33	1757.16	0.52	2050	1	99.71	0	0	0	0	0	0	1	1593.49

Table 5.1: Results of iterative approach for first officer scenario, summer 2005.

			crew pairing cost (c^P)				aircraft	aircraft changes									
				impr.		gap	routing		impr.	(minutes exceed. min. sit-time)							run time
scenario	iteration	p	lp ($\times 10^2$)	(%)	ip ($\times 10^2$)	(%)	cost (c^R)	c^{AC}	(%)	0	5	10	15	20	25	30	elapsed (s)
w05, 1	airline	-	313.74	-	313.93	0.06	320	62	-	4	1	2	2	1	1	5	-
	0	-	310.28	1.10	310.28	0.00	220	-	-	-	-	-	-	-	-	-	12.96
	1	0	310.56	1.01	310.66	0.03	240	55	11.29	6	0	0	1	0	2	5	19.14
	2	2	310.74	0.96	310.77	0.01	310	23	62.90	2	0	0	0	0	2	5	25.74
	3	5	311.03	0.87	311.03	0.00	310	16	74.19	1	0	0	0	0	2	5	34.49
	4	10	311.19	0.81	311.19	0.00	310	13	79.03	1	0	0	0	0	1	4	43.06
	5	20	311.32	0.77	311.32	0.00	270	12	80.65	1	0	0	0	0	1	3	50.21
	6	50	311.91	0.58	311.91	0.00	270	8	87.10	1	0	0	0	0	0	1	55.86
	7	100	311.91	0.58	311.91	0.00	270	8	87.10	1	0	0	0	0	0	1	61.33
	8	500	322.32	-2.74	322.36	0.01	270	0	100.00	0	0	0	0	0	0	0	67.72
w05, 3	airline	-	787.84	-	789.61	0.22	960	183	-	15	3	5	3	4	1	9	-
	0	-	779.51	1.06	785.69	0.79	600	-	-	-	-	-	-	-	-	-	59.56
	1	0	779.70	1.03	780.91	0.15	1060	158	13.66	13	1	3	5	4	1	12	92.96
	2	2	780.18	0.97	781.09	0.12	1120	93	49.18	7	0	2	2	3	2	13	130.44
	3	5	780.68	0.91	781.29	0.08	1040	55	69.95	4	0	1	0	3	1	11	159.81
	4	10	782.51	0.68	782.51	0.00	920	33	81.97	1	0	1	0	3	1	10	191.35
	5	20	782.34	0.70	782.34	0.00	920	33	81.97	1	0	1	0	3	1	10	218.63
	6	50	788.03	-0.02	791.54	0.44	910	15	91.80	0	0	0	0	3	1	4	267.80

			crew pairing cost (c^P)				aircraft	aircraft changes									
				impr.		gap	routing		impr.	(minutes exceed. min. sit-time)							run time
scenario	iteration	p	lp ($\times 10^2$)	(%)	ip ($\times 10^2$)	(%)	cost (c^R)	c^{AC}	(%)	0	5	10	15	20	25	30	elapsed (s)
	7	100	789.02	-0.15	793.27	0.54	870	11	93.99	0	0	0	0	3	0	2	313.38
	8	500	803.73	-2.02	806.24	0.31	860	0	100.00	0	0	0	0	0	0	0	358.93
w05, 7	airline	-	1663.62	-	1684.42	1.23	2220	374	-	30	9	8	5	5	8	19	-
	0	-	1625.19	2.31	1632.37	0.44	1410	-	-	-	-	-	-	-	-	-	234.53
	1	0	1626.56	2.23	1636.52	0.61	2330	289	22.73	29	2	3	3	2	6	29	405.46
	2	2	1627.27	2.18	1630.94	0.23	2280	205	45.19	19	1	2	4	1	3	31	513.00
	3	5	1628.49	2.11	1629.31	0.05	2370	165	55.88	14	1	2	2	2	4	29	614.41
	4	10	1631.85	1.91	1634.70	0.17	2200	112	70.05	8	0	2	2	2	3	26	744.60
	5	20	1635.72	1.68	1645.23	0.58	2190	58	84.49	4	0	1	2	0	1	15	972.41
	6	50	1642.06	1.30	1654.36	0.74	2190	47	87.43	3	0	1	1	1	0	14	1207.04
	7	100	1653.67	0.60	1670.09	0.98	2190	17	95.45	1	0	0	1	0	0	6	1422.26
	8	500	1678.67	-0.90	1700.83	1.30	2190	1	99.73	0	0	0	0	0	0	1	1666.60

Table 5.2: Results of iterative approach for first officer scenario, winter 2005.

			crew pairing cost (c^P)				aircraft	aircraft changes									
				impr.		gap	routing		impr.	(minutes exceed. min. sit-time)							run time
scenario	iteration	p	lp ($\times 10^2$)	(%)	ip ($\times 10^2$)	(%)	cost (c^R)	c^{AC}	(%)	0	5	10	15	20	25	30	elapsed (s)
s06, 1	airline	-	314.36	-	315.65	0.41	270	104	-	10	2	1	2	1	1	4	-
	0	-	312.70	0.53	313.09	0.12	130	-	-	-	-	-	-	-	-	-	12.54
	1	0	312.89	0.47	314.06	0.37	200	61	41.35	5	1	0	1	1	1	11	17.26
	2	2	313.09	0.40	313.75	0.21	230	22	78.85	1	0	0	1	1	0	8	23.08
	3	5	313.75	0.20	313.76	0.01	230	21	79.81	1	0	0	1	1	0	7	29.08
	4	10	313.83	0.17	313.83	0.00	230	13	87.50	0	0	0	1	1	0	6	37.66
	5	20	314.26	0.03	314.26	0.00	230	10	90.38	0	0	0	1	0	0	6	43.73
	6	50	314.75	-0.12	314.75	0.00	230	8	92.31	0	0	0	1	0	0	4	52.11
	7	100	315.01	-0.21	315.01	0.00	180	8	92.31	0	0	0	1	0	0	4	57.05
	8	500	322.13	-2.47	322.13	0.00	180	4	96.15	0	0	0	0	0	0	4	61.74
	9	1000	343.83	-9.37	356.41	3.53	180	0	100.00	0	0	0	0	0	0	0	68.92
s06, 3	airline	-	719.62	-	721.79	0.30	860	252	-	24	2	7	3	3	1	14	-
	0	-	710.81	1.22	710.94	0.02	410	-	-	-	-	-	-	-	-	-	54.62
	1	0	710.81	1.23	711.83	0.14	760	98	61.11	10	0	0	3	1	1	11	79.24
	2	2	710.85	1.22	710.85	0.00	760	48	80.95	3	0	0	3	1	1	10	102.66
	3	5	711.43	1.14	711.43	0.00	750	31	87.70	1	0	0	3	1	1	7	128.07
	4	10	712.08	1.05	712.08	0.00	740	30	88.10	1	0	0	3	1	1	6	159.78
	5	20	711.16	1.18	711.16	0.00	740	23	90.87	0	0	0	3	1	1	6	185.73

			crew pairing cost (c^P)				aircraft	aircraft changes									
				impr.		gap	routing		impr.	(minutes exceed. min. sit-time)							run time
scenario	iteration	p	lp ($\times 10^2$)	(%)	ip ($\times 10^2$)	(%)	cost (c^R)	c^{AC}	(%)	0	5	10	15	20	25	30	elapsed (s)
	6	50	714.04	0.78	711.28	-0.39	690	15	94.05	0	0	0	1	1	2	4	215.91
	7	100	715.36	0.59	714.27	-0.15	650	12	95.24	0	0	0	1	1	1	3	246.88
	8	500	730.64	-1.53	731.12	0.07	650	0	100.00	0	0	0	0	0	0	0	273.81
s06, 7	airline	-	1644.17	-	1673.21	0.55	1890	385	-	35	2	10	10	4	4	18	-
	0	-	1615.21	1.76	1617.23	0.12	850	-	-	-	-	-	-	-	-	-	229.73
	1	0	1614.43	1.81	1625.28	0.67	1450	364	5.45	36	3	7	6	1	4	24	370.05
	2	2	1618.73	1.55	1631.20	0.76	1540	212	44.94	18	2	2	7	3	5	17	538.57
	3	5	1618.96	1.53	1620.44	0.09	1480	107	72.21	7	0	3	4	3	1	16	624.55
	4	10	1619.83	1.48	1619.53	-0.02	1450	93	75.84	7	0	1	3	3	2	14	705.58
	5	20	1621.36	1.39	1621.36	0.00	1420	68	82.34	4	0	1	3	2	2	13	796.41
	6	50	1632.11	0.73	1634.23	0.13	1380	28	92.73	0	0	0	3	2	0	10	919.37
	7	100	1638.91	0.32	1638.29	-0.04	1370	21	94.55	0	0	0	2	1	0	10	1000.55
	8	500	1673.11	-1.76	1681.33	0.49	1370	1	99.74	0	0	0	0	0	0	1	1151.35
	9	1000	1674.63	-1.85	1674.73	0.01	1360	0	100.00	0	0	0	0	0	0	0	1230.94

Table 5.3: Results of iterative approach for first officer scenario, summer 2006.

			crew pairing cost (c^P)				aircraft	aircraft changes									
				impr.		gap	routing		impr.	(minutes exceed. min. sit-time)							run time
scenario	iteration	p	lp ($\times 10^2$)	(%)	ip ($\times 10^2$)	(%)	cost (c^R)	c^{AC}	(%)	0	5	10	15	20	25	30	elapsed (s)
s05, 1	airline	-	398.35	-	398.54	0.05	2250	53	-	4	1	1	2	1	0	3	-
	0	-	397.63	0.18	397.76	0.03	260	-	-	-	-	-	-	-	-	-	13.46
	1	0	397.85	0.13	397.85	0.00	340	70	-32.08	6	0	1	2	2	2	5	18.46
	2	2	398.03	0.08	398.03	0.00	370	21	60.38	1	0	0	0	2	2	4	25.84
	3	5	398.37	0.00	398.37	0.00	370	10	81.13	0	0	0	0	2	1	2	32.96
	4	10	398.37	0.00	398.37	0.00	370	10	81.13	0	0	0	0	2	1	2	37.18
	5	20	398.64	-0.07	398.64	0.00	370	8	84.91	0	0	0	0	2	0	2	42.93
	6	50	399.47	-0.28	399.10	-0.09	370	7	86.79	0	0	0	0	2	0	1	48.07
	7	100	400.53	-0.55	399.87	-0.17	370	6	88.68	0	0	0	0	2	0	0	53.60
	8	500	419.37	-5.28	419.37	0.00	370	0	100.00	0	0	0	0	0	0	0	60.80
s05, 3	airline	-	977.85	-	978.40	0.06	4020	212	-	18	3	4	5	5	2	9	-
	0	-	969.85	0.82	970.51	0.07	580	-	-	-	-	-	-	-	-	-	57.61
	1	0	970.01	0.80	970.22	0.02	960	136	35.85	11	1	1	3	5	2	17	88.25
	2	2	970.13	0.79	970.29	0.02	900	104	50.94	7	2	0	2	5	2	16	122.51
	3	5	970.66	0.73	970.66	0.00	910	77	63.68	5	1	0	1	4	2	16	147.65
	4	10	971.41	0.66	971.41	0.00	910	58	72.64	4	0	0	1	4	1	12	182.76
	5	20	974.38	0.35	979.21	0.49	910	36	83.02	1	1	0	1	5	0	4	221.25
	6	50	979.72	-0.19	977.64	-0.21	850	24	88.68	1	0	0	1	3	0	4	252.36

			crew pairing cost (c^P)				aircraft	aircraft changes									
				impr.		gap	routing		impr.	(minutes exceed. min. sit-time)							run time
scenario	iteration	p	lp ($\times 10^2$)	(%)	ip ($\times 10^2$)	(%)	cost (c^R)	c^{AC}	(%)	0	5	10	15	20	25	30	elapsed (s)
	7	100	979.94	-0.21	984.49	0.46	820	15	92.92	1	0	0	0	2	0	2	278.79
	8	500	1004.02	-2.68	1004.43	0.04	820	1	99.53	0	0	0	0	0	0	1	314.15
	9	1000	1011.28	-3.42	1011.39	0.01	820	0	100.00	0	0	0	0	0	0	0	336.35
s05, 7	airline	-	1760.76	-	1776.49	0.89	8210	372	-	31	7	7	9	4	7	16	-
	0	-	1724.06	2.08	1747.61	1.35	1470	-	-	-	-	-	-	-	-	-	376.85
	1	0	1723.77	2.10	1742.42	1.07	2390	322	13.44	30	4	1	7	4	7	29	588.92
	2	2	1724.36	2.07	1749.42	1.43	2300	238	36.02	22	2	0	6	4	5	26	827.49
	3	5	1725.63	2.00	1753.86	1.61	2300	180	51.61	13	3	1	4	4	5	28	1007.15
	4	10	1728.95	1.81	1748.85	1.14	2320	134	63.98	9	4	0	2	5	2	20	1183.44
	5	20	1732.19	1.62	1747.31	0.86	2360	84	77.42	4	3	0	1	3	2	21	1363.53
	6	50	1745.13	0.89	1758.74	0.77	2360	44	88.17	2	2	0	1	2	0	8	1525.07
	7	100	1769.71	-0.51	1790.51	1.16	2230	11	97.04	0	1	0	0	0	0	5	1724.39
	8	500	1784.43	-1.34	1817.22	1.80	2160	3	99.19	0	0	0	0	0	0	3	1960.90
	9	1000	1790.96	-1.72	1813.45	1.24	2160	0	100.00	0	0	0	0	0	0	0	2172.89

Table 5.4: Results of iterative approach for captains scenario, summer 2005.

			crew pairing cost (c^P)				aircraft	aircraft changes									
				impr.		gap	routing		impr.	(minutes exceed. min. sit-time)							run time
scenario	iteration	p	lp ($\times 10^2$)	(%)	ip ($\times 10^2$)	(%)	cost (c^R)	c^{AC}	(%)	0	5	10	15	20	25	30	elapsed (s)
s05, 1	airline	-	121.62	-	121.83	0.17	2250	79	-	7	1	1	2	1	0	8	-
	0	-	121.62	0.00	123.75	1.72	260	-	-	-	-	-	-	-	-	-	8.18
	1	0	121.62	0.00	122.17	0.45	410	42	46.84	4	0	0	1	1	0	7	12.38
	2	2	121.89	-0.22	124.13	1.80	300	15	81.01	1	0	0	1	0	0	4	16.58
	3	5	122.35	-0.60	122.35	0.00	300	3	96.20	0	0	0	0	0	0	3	20.00
	4	10	122.38	-0.62	122.35	-0.02	300	3	96.20	0	0	0	0	0	0	3	23.12
	5	20	122.49	-0.71	122.35	-0.11	300	3	96.20	0	0	0	0	0	0	3	27.43
	6	50	122.65	-0.85	123.07	0.33	300	1	98.73	0	0	0	0	0	0	1	31.58
	7	100	123.83	-1.81	123.83	0.00	270	1	98.73	0	0	0	0	0	0	1	36.55
	8	500	125.02	-2.79	125.26	0.19	270	0	100.00	0	0	0	0	0	0	0	40.06
s05, 3	airline	-	319.23	-	322.48	1.01	4020	192	-	18	1	3	5	1	0	22	-
	0	-	319.18	0.01	324.00	1.49	580	-	-	-	-	-	-	-	-	-	22.33
	1	0	319.50	-0.09	320.77	0.40	750	133	30.73	12	0	3	2	1	1	21	35.13
	2	2	319.76	-0.17	320.34	0.18	780	59	69.27	6	0	1	0	0	0	12	46.17
	3	5	320.29	-0.33	320.19	-0.03	760	31	83.85	1	0	1	2	0	0	11	64.58
	4	10	320.19	-0.30	320.19	0.00	760	26	86.46	1	0	0	2	0	0	11	80.49
	5	20	322.67	-1.08	322.14	-0.16	760	13	93.23	0	0	0	1	0	0	9	92.95
	6	50	323.40	-1.31	325.11	0.52	750	4	97.92	0	0	0	0	0	0	4	109.77

			crew pairing cost (c^P)				aircraft	aircraft changes									
				impr.		gap	routing		impr.	(minutes exceed. min. sit-time)							run time
scenario	iteration	p	lp ($\times 10^2$)	(%)	ip ($\times 10^2$)	(%)	cost (c^R)	c^{AC}	(%)	0	5	10	15	20	25	30	elapsed (s)
	7	100	329.33	-3.16	336.78	2.21	750	0	100.00	0	0	0	0	0	0	0	124.36
s05, 7	airline	-	687.89	-	698.35	1.50	8210	485	-	42	5	7	15	3	3	51	-
	0	-	686.72	0.17	701.33	2.08	1470	-	-	-	-	-	-	-	-	-	90.18
	1	0	687.63	0.04	700.34	1.81	2010	198	59.18	15	2	1	4	3	4	43	143.47
	2	2	688.51	-0.09	696.41	1.13	2040	149	69.28	9	2	2	3	2	6	34	199.97
	3	5	690.45	-0.37	700.78	1.47	2000	162	66.60	12	1	0	3	4	5	38	250.57
	4	10	692.02	-0.60	702.83	1.54	2010	86	82.27	6	0	1	1	2	2	25	306.46
	5	20	695.59	-1.12	704.81	1.31	2040	26	94.64	0	0	0	0	2	1	18	356.66
	6	50	701.21	-1.94	705.36	0.59	1990	20	95.88	0	0	0	0	2	1	12	403.56
	7	100	713.37	-3.70	712.16	-0.17	1950	9	98.14	0	0	0	0	0	1	7	442.76
	8	500	728.22	-5.86	731.69	0.47	1950	2	99.59	0	0	0	0	0	0	2	510.01
	9	1000	726.26	-5.58	737.09	1.47	1910	0	100.00	0	0	0	0	0	0	0	570.70

Table 5.5: Results of iterative approach for cabin crew scenario, summer 2005.

Figure 5.4 shows the impact of the DPACLim rule for the first officer scenario, schedule summer 2005, 7 days. Relaxing the rule from 1 to 2 for the airline approach results in a solution with less crew pairing costs but many more restricted aircraft changes and hence it is less robust. In the iterative approach cheaper solutions can be generated if the DPACLim rule is relaxed from 1 to 2. Here the solutions are equally robust for both settings since restricted aircraft changes are penalised in both approaches. Since the DPACLim rule is only used by Air New Zealand to increase the robustness of the solutions the rule is relaxed to 2 for the iterative approach since robustness is achieved by means of aircraft change costs. From a practical point of view, multiple aircraft changes can be tolerated if this does not affect robustness, i.e. the aircraft changes occur on connections with long ground times. Note that because of the LP/IP gaps, LP values are displayed in Figure 5.4 to obtain a more consistent representation.

Table 5.6 lists some more details about the characteristics of the solutions. Statistics are shown for setting DPACLim to 1 and 2, respectively. We list the airline solutions as well as the solutions generated by the iterative approach. For DPACLim equal to 1 we list the total number of duty periods in the solution ("all"). We show the number of duty periods with 1 aircraft change ("1 ac") and the number of restricted aircraft changes ("(rac)") within these duty periods. Column c^{AC} shows the aircraft change cost for each solution. For a DPACLim setting of 2 we again show the total number of duty periods. Additionally, we list the number of duty periods with 1 ("ac 1") and 2 ("ac 2") aircraft changes and the number of restricted aircraft changes contained in both types of duty periods, respectively. Finally, column c^{AC} again displays the aircraft change costs of the solutions. We observe that the relaxation of the rule has no negative impact on the aircraft change cost of the solutions of the iterative algorithm. The reduction in crew pairing cost is achieved by increasing the number of duty periods with two aircraft changes. However, the aircraft change costs of these solutions do not increase significantly. In iteration 3 for example, aircraft change costs are almost identical but we observe (see Figure 5.4) a decrease in crew pairing cost of 0.77 % for setting DPACLim to 2. This cheaper solution does contain 16 duty periods with 2 aircraft changes but only 11 restricted aircraft changes. This demonstrates that we can achieve a better crew pairing cost and a more robust solution if we allow a small number of duty periods to contain two aircraft changes.

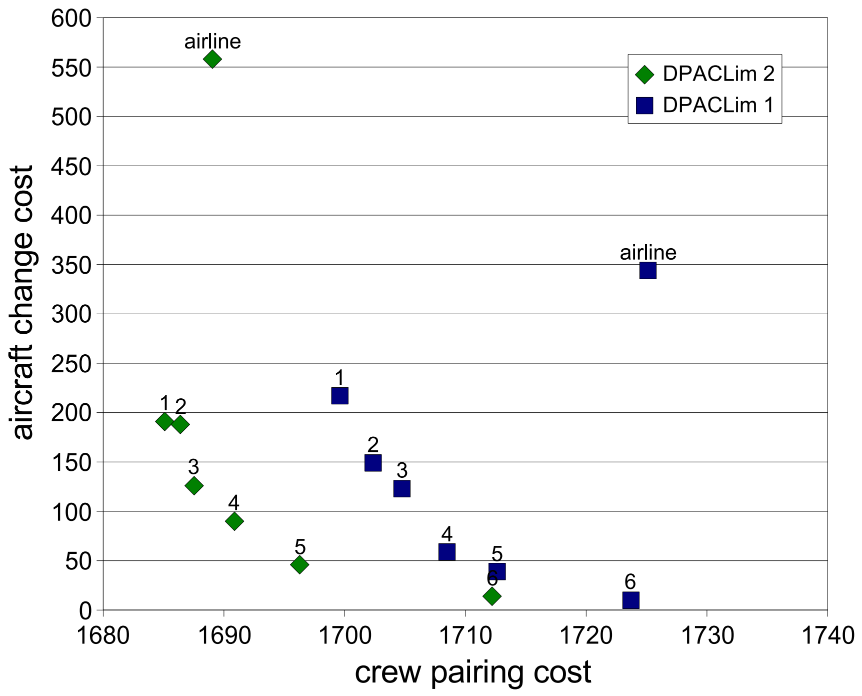


Figure 5.4. Variation of DPACLim for first officer scenario, summer 2005, 7 days.

	DPACLim 1				DPACLim 2					
	no. duty periods				no. duty periods					
iteration	all	1 ac	(rac)	c^{AC}	all	1 ac	(rac)	2 ac	(rac)	c^{AC}
airline	210	115	72	344	206	74	52	56	76	558
0	204	101	54	255	202	77	46	25	24	289
1	203	98	52	217	202	81	43	17	14	191
2	205	83	35	149	202	83	40	21	19	188
3	204	86	34	123	202	76	26	16	11	126
4	204	81	17	59	203	73	24	16	10	90
5	205	77	14	39	205	60	12	12	3	46
6	205	81	5	10	206	69	5	10	1	14
7	207	83	1	2	207	66	1	17	0	1

Table 5.6. Variation of DPACLim for first officer scenario, summer 2005, 7 days.

5.3.2 Comparison of Iterative Approach and Optimisation Approaches

In this section we use Dantzig-Wolfe and Benders decomposition methods to solve the LP relaxation of the robust and integrated aircraft routing and crew pairing problem. The purpose of the computational experiments is twofold. Firstly, we want to establish lower bounds for the solution values of the integrated problem. We compare the integer solutions of the iterative approach with the LP lower bound from the decomposition methods. Secondly, we compare the running times of Dantzig-Wolfe and Benders decomposition approaches for solving the LP relaxation to establish which algorithm performs better.

For each of the scenarios investigated in the previous section we apply each decomposition approach twice. In one run we set weight p equal to 2 and in the other run we set weight p equal to 20. This corresponds to the value of weight p in iterations 2 and 5 of the iterative approach.

In the Benders decomposition approach the master problem and subproblem are only solved to LP optimality. In the Dantzig-Wolfe approach the master problem is solved to LP optimality and integer solutions for the subproblems are found. Both decomposition approaches provide lower bounds for the optimal solution of the optimal LP solution (see Section 1.5). Once the gap between the lower bound and the best LP solution value found is below 0.5% we stop the algorithm.

Table 5.7 summarises the results of the experiments. We list crew group, scenario name, solution method and value of p for all experiments we perform. In the next two columns the number of iterations and running time needed to reach the stopping criterion are listed. If the optimality gap of 0.5% is not reached in 50 iterations, we stop the algorithm and report the gap that is obtained after 50 iterations. For each optimisation run we list lower and upper bound of the optimal LP solution and the gap between the two values. Note that because of the heuristic nature of the crew pairing solver (i.e. the dominance relaxed shortest path) that is used, we can sometimes observe two different intervals for the same optimal solutions that are not overlapping (see for example Table 5.7 scenarios f33, s05, 7 days, $p = 20$ and c33, s05, 7 days,

$p = 20$). Of course, this cannot occur if the solution approach for the crew pairing problem is truly optimal. The observed gaps between two intervals for the same optimal solution is usually very small ($< 0.1\%$). In two instances, however, the observed gap is 0.32% (c33, s05, 7 days, $p = 20$) and 0.33% (f33, s05, 7 days, $p = 20$), respectively. For the iterative approach, we show the best solution that is found with respect to the objective function that is used in the optimisation approaches. We apply this objective function to the integer solutions of the iterative approach and display the objective value and the gap with respect to the lower bound obtained by the optimisation approach. We also list the time needed for the iterative approach to obtain the solution.

Note that the lower bound LP relaxation solutions do not necessarily satisfy the DPACLim rule restrictions since the rule is not integrated into the model. The LP/IP gaps observed are partially caused by the violation of the DPACLim rule. Also, more than two aircraft changes in a duty period are not desirable for the problem instances we consider. Hence, the iterative approach has the advantage of easily satisfying the DPACLim rule while the optimisation approaches cannot guarantee to satisfy the rule.

The average gap between iterative approach solution and LP lower bound is very small (0.9%). In most cases this integer solution is found before the stopping criterion of either optimisation approach is reached. The observed gap is also well below the branch-and-bound gap of 2%. Using the iterative approach, we always find an integer solution within the 2% gap and hence, we do not solve the optimisation approaches to IP optimality. The additionally required run time cannot result in significantly improved solution quality. In two cases the Dantzig-Wolfe decomposition indicates a gap exceeding 2% but the gap is smaller than 2% for the lower bound of Benders decomposition in both cases.

In terms of run time, Benders decomposition seems to be superior to Dantzig-Wolfe decomposition. However, integer subproblems are solved for the Dantzig-Wolfe decomposition approach while the subproblem is only solved to LP optimality for Benders decomposition. It is noteworthy that the contribution by Mercier et al. [2005] indicates fewer iterations and much faster run times for Benders decomposition than observed in our computational experiments. We believe that this is caused by the authors solving a slightly relaxed problem, while in our approach a real world application is addressed together with all applicable rules. It also seems to be much harder to obtain an optimal solution for both decomposition approaches when the weight p is large. For Benders decomposition, this is caused since larger costs are associated with the subproblem and hence the subproblem must not only transfer feasibility but also optimality information back to the master problem. The Dantzig-Wolfe decomposition is also more difficult when p is large since the sum of crew pairing cost and aircraft routing cost as naturally available lower bound for the objective value of the relaxed master problem is further away from an optimal solution than for small values of p.

Note that both optimisation approaches are implemented in a basic fashion. No dual stabilisation method is used to speed up Dantzig-Wolfe decomposition, and Benders decomposition is not enhanced with stronger cut generation. We omit these improvements since we do not believe that the run times of the optimisation approaches can be improved sufficiently to compete with those of the iterative approach. We use the results of the optimisation approaches to verify the quality of the solutions of the iterative approach. In practice we suggest only running the decomposition approach to verify the solution quality of the iterative approach. If this is only done rarely and run time is not very important then Dantzig-Wolfe decomposition can be chosen as the solution method since it is much easier to implement into existing aircraft routing and crew pairing optimisation algorithms than Benders decomposition.

We investigated hot-starting the Dantzig-Wolfe decomposition approach by generating columns for the restricted master problem from all solutions of the iterative approach but this procedure did not improve run times significantly.

									iterative approach		
crew				number of		lp cost ($\times 10^2$)			ip solution	gap	run time
group	scenario	method	p	iterations	run time (s)	lower bound	upper bound	gap (%)	value ($\times 10^2$)	(%)	elapsed (s)
f33	s05, 1	DW	2	2	24.05	348.62	349.49	0.25	349.48	0.24	48.96
		DW	20	10	119.05	348.98	349.98	0.29	350.92	0.55	48.96
		Benders	2	2	24.28	348.35	349.17	0.23	349.48	0.32	48.96
		Benders	20	11	77.64	349.22	350.89	0.48	350.92	0.48	48.96
	s05, 3	DW	2	2	103.46	890.20	892.96	0.31	897.11	0.77	269.05
		DW	20	19	812.15	892.97	896.49	0.39	902.33	1.04	269.05
		Benders	2	2	89.48	890.88	892.79	0.21	897.11	0.69	269.05
		Benders	20	16	407.30	896.60	899.95	0.37	902.33	0.64	269.05
	s05, 7	DW	2	2	343.12	1702.51	1709.76	0.42	1718.89	0.95	734.08
		DW	20	36	3329.00	1709.81	1716.92	0.41	1743.43	1.93	902.01
		Benders	2	2	352.04	1706.92	1708.51	0.09	1718.89	0.70	734.08
		Benders	20	15	1420.18	1722.59	1729.63	0.41	1743.43	1.20	902.01
c33	s05, 1	DW	2	2	22.34	400.23	401.85	0.40	402.15	0.48	25.84
		DW	20	14	108.17	401.24	402.40	0.29	403.94	0.67	42.93
		Benders	2	2	22.01	400.82	401.72	0.22	402.15	0.33	25.84
		Benders	20	11	76.45	402.59	404.22	0.40	403.94	0.33	42.93
	s05, 3	DW	2	2	115.33	976.94	979.68	0.28	981.30	0.44	147.65
		DW	20	17	522.12	980.65	984.07	0.35	990.94	1.04	252.36

									iterative approach		
crew				number of		lp cost ($\times 10^2$)			ip solution	gap	run time
group	scenario	method	p	iterations	run time (s)	lower bound	upper bound	gap (%)	value ($\times 10^2$)	(%)	elapsed (s)
		Benders	2	2	93.10	977.40	979.75	0.24	981.30	0.40	147.65
		Benders	20	16	493.09	984.30	988.39	0.41	990.94	0.67	252.36
	s05, 7	DW	2	2	421.12	1741.17	1748.54	0.42	1772.59	1.77	1363.53
		DW	20	26	2911.00	1746.81	1754.37	0.43	1787.71	2.29	1363.53
		Benders	2	2	321.19	1745.84	1747.69	0.11	1772.59	1.51	1363.53
		Benders	20	15	1674.29	1759.96	1768.44	0.48	1787.71	1.55	1363.53
spsr	s05, 1	DW	2	3	26.22	124.63	124.96	0.26	125.41	0.62	20.00
		DW	20	16	88.12	124.98	125.57	0.47	125.95	0.78	20.00
		Benders	2	2	14.39	124.61	124.90	0.23	125.41	0.64	20.00
		Benders	20	15	75.52	125.37	125.89	0.41	125.95	0.46	20.00
	s05, 3	DW	2	3	66.23	326.15	327.72	0.48	328.31	0.66	80.49
		DW	20	40	478.12	329.25	330.87	0.49	332.34	0.93	92.95
		Benders	2	3	53.07	326.30	327.36	0.32	328.31	0.61	80.49
		Benders	20	32	410.94	331.04	332.54	0.45	332.34	0.39	92.95
	s05, 7	DW	2	3	178.12	704.46	707.49	0.43	719.79	2.13	199.97
		DW	20	51	2912.00	708.45	718.04	1.34	729.26	2.85	403.56
		Benders	2	2	123.16	705.86	707.88	0.29	719.79	1.94	199.97
		Benders	20	44	1916.30	718.77	722.26	0.48	729.26	1.44	403.56

									iterative approach		
crew				number of		lp cost ($\times 10^2$)			ip solution	gap	run time
group	scenario	method	p	iterations	run time (s)	lower bound	upper bound	gap (%)	value ($\times 10^2$)	(%)	elapsed (s)
f33	w05, 1	DW	2	2	24.18	312.89	313.80	0.29	314.16	0.40	19.14
		DW	20	17	140.08	313.70	314.93	0.39	316.21	0.80	55.86
		Benders	2	3	28.40	312.50	313.67	0.37	314.16	0.53	19.14
		Benders	20	14	102.74	314.22	315.49	0.40	316.21	0.63	55.86
	w05, 3	DW	2	3	145.12	786.81	789.06	0.29	792.20	0.68	218.63
		DW	20	34	1005.07	789.02	792.41	0.43	798.14	1.14	218.63
		Benders	2	2	80.67	788.05	789.11	0.13	792.20	0.52	218.63
		Benders	20	21	521.62	791.77	795.60	0.48	798.14	0.80	218.63
	w05, 7	DW	2	3	523.43	1641.59	1647.30	0.35	1656.31	0.89	614.41
		DW	20	51	4956.00	1643.77	1652.91	0.55	1678.73	2.08	972.41
		Benders	2	2	324.71	1643.62	1648.26	0.28	1656.31	0.77	614.41
		Benders	20	19	2017.75	1653.82	1661.20	0.44	1678.73	1.48	972.41
f33	s06, 1	DW	2	2	22.76	314.29	315.70	0.45	316.39	0.66	37.66
		DW	20	14	111.65	314.95	316.50	0.49	318.41	1.09	57.05
		Benders	2	2	19.53	314.56	316.03	0.46	316.39	0.58	37.66
		Benders	20	16	109.84	315.93	317.28	0.43	318.41	0.78	57.05
	s06, 3	DW	2	3	104.08	715.41	718.78	0.47	718.48	0.43	215.91

crew				number of		lp cost ($\times 10^2$)			iterative approach		
group	scenario	method	p	iterations	run time (s)	lower bound	upper bound	gap (%)	ip solution value ($\times 10^2$)	gap (%)	run time elapsed (s)
		DW	20	49	1323.56	715.88	719.45	0.50	721.18	0.74	215.91
		Benders	2	4	124.72	715.55	717.48	0.27	718.48	0.41	215.91
		Benders	20	26	584.09	718.37	721.87	0.49	721.18	0.39	215.91
	s06, 7	DW	2	3	388.76	1625.66	1632.10	0.39	1635.89	0.63	705.58
		DW	20	51	4456.00	1623.71	1636.55	0.78	1649.16	1.54	796.41
		Benders	2	2	328.79	1623.52	1631.08	0.46	1635.89	0.76	705.58
		Benders	20	34	2821.00	1634.72	1642.83	0.49	1649.16	0.88	796.41

Table 5.7: Comparison of iterative approach, Dantzig-Wolfe decomposition approach, and Benders decomposition approach.

5.3.3 Iterative Approach for Multiple Crew Groups

In this section we show results for considering two crew groups in the iterative approach instead of one. We extend the iterative algorithm to solve the aircraft routing problem together with the crew pairing problems for technical crew and flight attendants. The goal is to show that the benefits of solving one crew group together with the aircraft are not lost if we add another crew group. Clearly, using a solution that is very robust for one crew group but results in many restricted aircraft changes for another crew group is not desirable. Figures 5.5 and 5.6 summarise the results. Note that LP solution values are displayed in both figures to obtain a more consistent overview.

In Figure 5.5 the results of applying the iterative algorithm to first officers only are shown as blue squares. Similarly, in Figure 5.6 the results of considering cabin crew only are also marked as blue squares. Additionally, we solve the first officer scenario as before and in each iteration use the aircraft routing solution to generate solutions for the cabin crew problem. This corresponds to setting $w_1 = 1$ and $w_2 = 0$ in objective function (5.6). The results are shown as red diamonds in Figure 5.6. The solutions incur more crew pairing cost and more aircraft change cost than the solutions obtained from focusing on cabin crew and aircraft only. But the solutions follow the same pattern and are of good quality compared to the airline solution. Hence we do not need to sacrifice solution quality of the cabin crew problem in order to improve the first officer solution and hence improve the overall solution to the integrated problem. Note that rules for captains, the third crew group, are very similar to the first officer scenario and hence we expect to obtain almost identical solutions for captains compared to first officers.

In a second experiment we again solve two crew pairing problems in each iteration but we now use feedback information from the first officer problem as well as the cabin crew problem to generate the aircraft routing solution. In the aircraft routing problem we scale the penalties for the restricted connections of the first officer solution and of the cabin crew solution with the same weight. This corresponds to setting $w_1 = 1$ and $w_2 = 1$ in objective function (5.6). We then generate aircraft routings subject to this objective function. The resulting solutions are represented as light blue triangles in both figures. In Figure 5.5, we observe that the first officer solutions do not deteriorate sig-

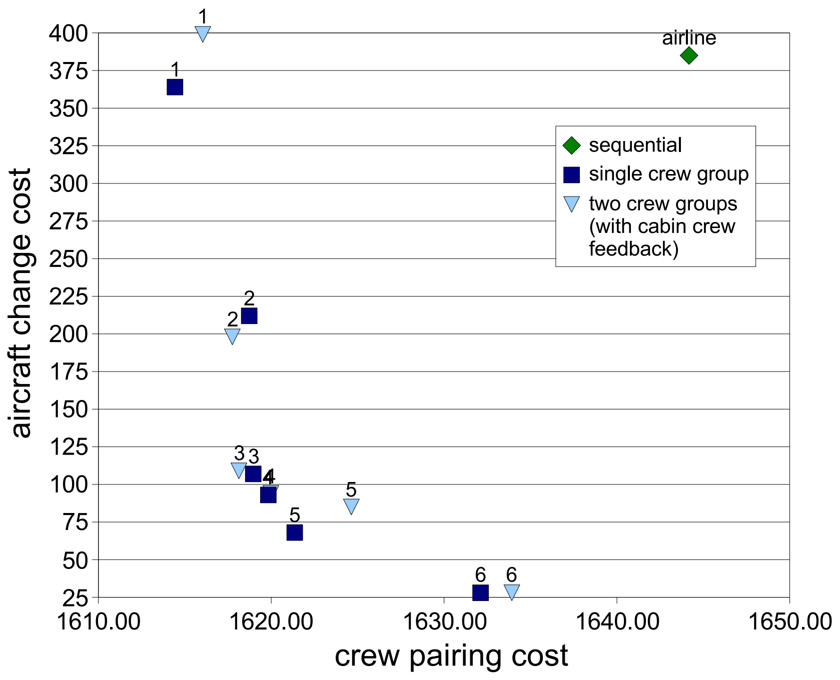


Figure 5.5. Results for first officer scenario, summer 2006, 7 days, with cabin crew solved simultaneously.

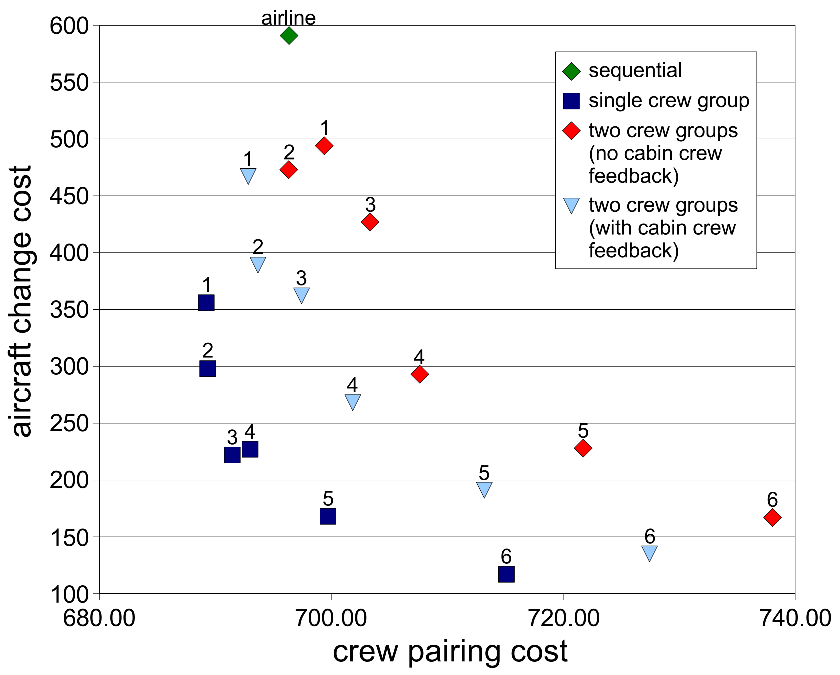


Figure 5.6. Results for cabin crew scenario, summer 2006, 7 days, with first officers solved simultaneously.

nificantly. In Figure 5.6, the solutions for cabin crew are not as good as the best cabin crew solutions (squares), but better than the solutions generated without using any feedback from the cabin crew solutions (diamonds). Note that for the more robust of these solutions, both crew groups rarely change aircraft if the connection is restricted. This also implies that both crew groups usually stay together on restricted connections, which is of great advantage not only from a robustness point of view but also from an operational point of view since keeping the crew as one unit greatly simplifies business procedures. The solutions satisfy this property without the need to focus on the property in the algorithm.

We only use two basic configurations of the weights ($(1, 0)$ and $(1, 1)$) to scale the penalties. The weights can be adjusted according to the airline's preferences, e.g. relative to the cost incurred by each crew group. We can now generate solutions for the aircraft routing and the two crew pairing problems in one integrated procedure. Likewise, additional crew groups or multiple aircraft types can be added in a straightforward way.

5.4 Simulation

In this thesis we consider sequences of minimal turns within an aircraft routing and aircraft change costs as two measures of the operational robustness of the integrated solution. We use simulation to estimate whether more robust solutions with respect to our measures are indeed solutions where disruptions are less likely to propagate onto other flights causing additional delays. The final test of robustness of the improved solutions can only be made once the solutions are operated in practice.

The aircraft routing solution and two crew solutions, one for technical crew and one for cabin crew, are used as input for the simulation. We assume that the two technical crew group solutions, for captains and first officers, are identical. During the simulation we loop over all flights which are ordered by increasing departure time. For each flight a possible initial disruption delay is generated randomly depending on origin and hour of departure. The distribution that is used to calculate delays is derived from actual delay data over two years. A simple *push-back recovery* is used: each delayed flight departs as soon as the aircraft and both crew types are available to operate this flight. The time at which aircraft and crew are available depends on the actual departure times of the flights they are operating prior to the current flight. For each flight the actual delay is calculated as the maximum of three values: delay caused by waiting for technical crew, delay caused by waiting for cabin crew, and the sum of delay caused by waiting for aircraft and new initial disruption delay. The delay is added to the original departure time of the flight to obtain the actual departure time. Iteratively, all flights are considered in the same way. We repeat this simulation 1000 times with different random seeds.

As a result, we obtain estimates of the on-time performance (OTP) of the solutions, i.e. what percentage of the flights depart within 10 minutes of the originally scheduled departure time, and the number of minutes of delay. In Table 5.8 we display the simulation results for the "airline" solution and for the solutions from iterations 2 and 5 of the iterative approach. We use the iterative approach for two crew groups with identical weights w_1, w_2 for both crew groups as in the previous section to obtain the solutions. For each solution we list crew pairing costs and aircraft change costs for both crew groups, technical crew ("f33") and cabin crew ("spsr"). We also display aircraft routing costs. The last column shows the OTP percentage. For the airline solution 78.34% of all flights depart within 10 minutes of the scheduled departure time. For iteration 2 and 5 the values are 80.03% and 81.16%, respectively. All three robustness measures are better for the iterative approach solutions compared to the airline solution. This increase in robustness is reflected in better OTP, where the OTP value increases by up to 3.6%. We only list results for a single data set, results are very similar for all scenarios considered in this chapter.

	f33		spsr			
iteration	$c^P(\times 10^2)$	c^{AC}	$c^P(\times 10^2)$	c^{AC}	c^R	OTP (%)
airline	1673.21	385	706.56	591	1890	78.34
2	1624.29	198	713.37	389	1540	80.03
5	1627.32	85	713.07	191	1460	81.16

Table 5.8. On-time performance for iterative approach solutions, first officer and cabin crew scenarios, summer 2006, 7 days.

Table 5.9 lists the minutes of delay that occurred for the different solutions. We list the number of minutes of delay that are occurring during a week.

Delays are listed for new initial disruptions ("new"), delays caused by waiting for aircraft, delays caused by waiting for technical crew ("f33"), and delays caused by waiting for cabin crew ("spsr"). The final column of Table 5.9 shows the total number of passenger delay minutes ("PDM") per year. The values are obtained by multiplying the occurring delays by average passenger numbers for every pair of origin and hour of departure. We observe a significant decrease in all *reactionary* delay figures. These delays are all delays caused by waiting for aircraft and crew. The annual passenger delay minutes decrease by more than 4.2 million minutes or almost 14%. All values are average numbers over all simulations performed.

	weekly				annual
iteration	new	aircraft	f33	spsr	total PDM
airline	2456.76	2551.12	382.84	413.07	30360061.95
2	2454.06	2473.75	137.76	228.88	27838728.10
5	2456.88	2304.04	68.18	131.70	26143271.69

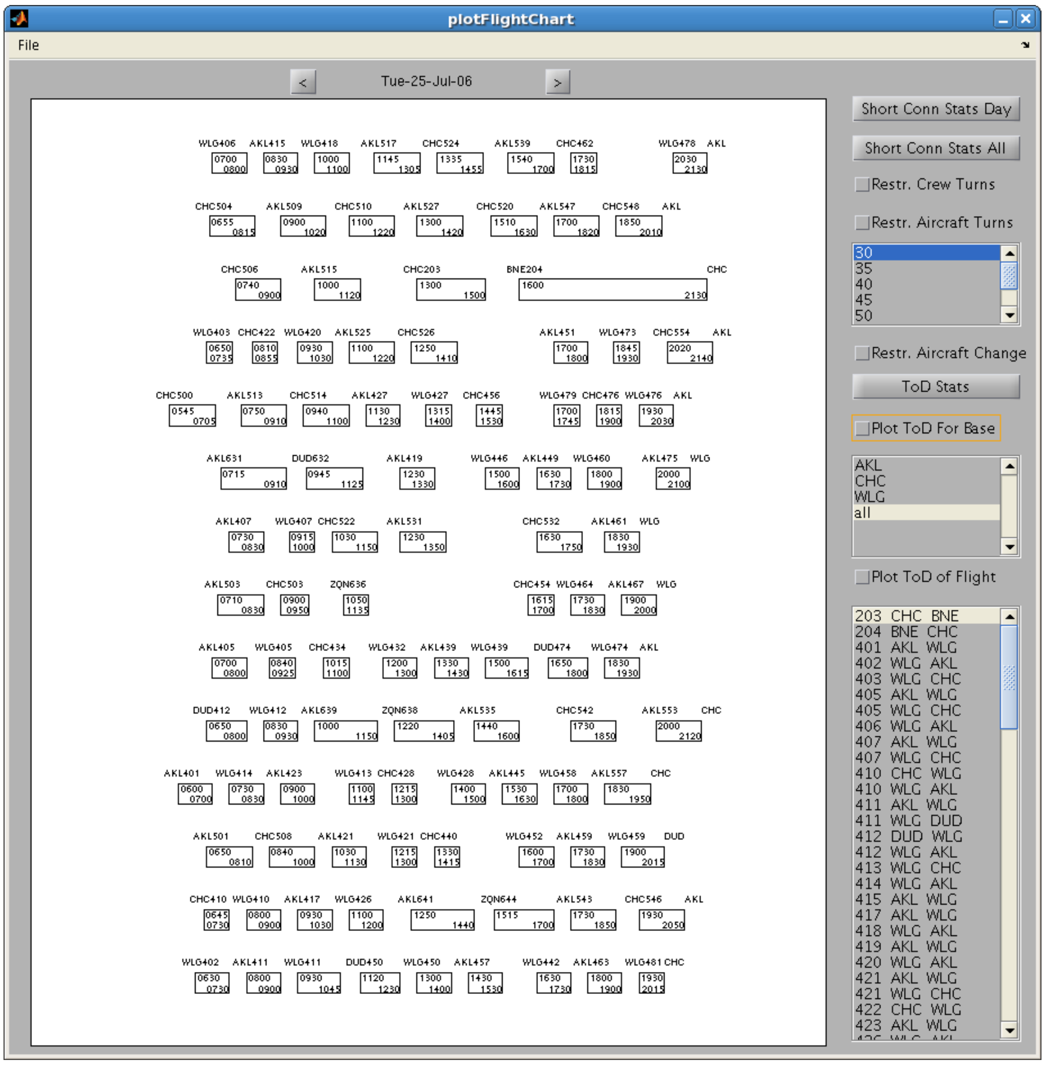
Table 5.9. Minutes of delay listed by reason for iterative approach solutions, first officer and cabin crew scenarios, summer 2006, 7 days.

5.5 Visualisation

We use a visualisation GUI to compare solutions and verify the solution quality. The visualisation GUI is implemented in MATLAB 7.3.0 (The MathWorks Inc. [2003]). Aircraft routings for all aircraft for the whole solution period are displayed together with crew pairings of one crew pairing solution. It is possible to view selected pairings or pairings originating from particular bases. Minimal aircraft turns, restricted aircraft changes, and various statistics can also be displayed. It is also possible to print solutions over any time period.

In the following screen-shots a single day of solutions of the first officer schedule of summer 2006 is shown. Each flight is represented by a rectangle associated with flight number, origin, destination, departure time, and arrival time. Aircraft routings are displayed as rows of flights. Crew pairings are represented as red lines connecting flights. The crew pairings are labelled with id numbers, the start ("ST") and end ("ET") of the crew pairings are identified as well as overnight rest breaks ("R"). If a crew pairing starts with a blue line, the crew is passengering on the blue flights until the red line of the pairing starts. Similarly, if the crew pairing ends with a green line, the crew is passengering at the end of the pairing back to their home base.

Screen-shot 5.7 displays an aircraft routing solution only, for Tuesday, July 25, 2006. All times are given in local Auckland time. The difference between a robust and non-robust solution becomes obvious in screen-shots 5.8 and 5.9. Figure 5.8 displays the solution that is generated by using the traditional sequential approach which is an aircraft routing solution that was operated by Air New Zealand. Figure 5.9 shows the solution of iteration 5 of the iterative approach for the same day. Only very few aircraft changes can be observed in the second Figure. Aircraft changes can easily be identified in this representation as diagonal lines connecting one aircraft with another in a different row of flights.



```
30
35
40
45
50
```

```
AKL
CHC
WLG
all
```

```
203  CHC  BNE
204  BNE  CHC
401  AKL  WLG
402  WLG  AKL
403  WLG  CHC
405  AKL  WLG
405  WLG  CHC
406  WLG  AKL
407  AKL  WLG
407  WLG  CHC
410  CHC  WLG
410  WLG  AKL
411  AKL  WLG
411  WLG  DUD
412  DUD  WLG
412  WLG  AKL
413  WLG  CHC
414  WLG  AKL
415  AKL  WLG
417  AKL  WLG
418  WLG  AKL
419  AKL  WLG
420  WLG  AKL
421  AKL  WLG
421  WLG  CHC
422  CHC  WLG
423  AKL  WLG
```

Figure 5.7. Screen-shot of aircraft routing solution for first officer scenario, summer 2006.

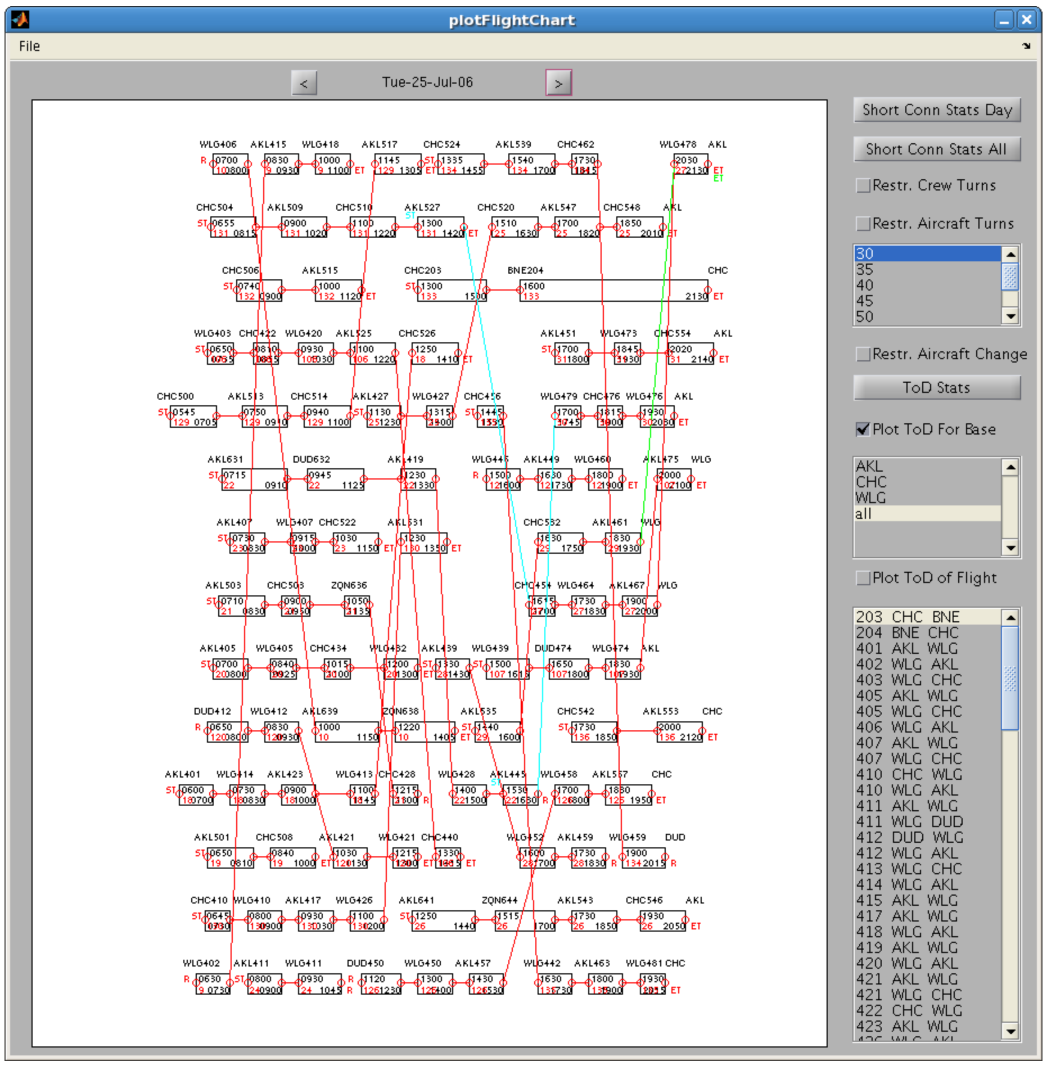
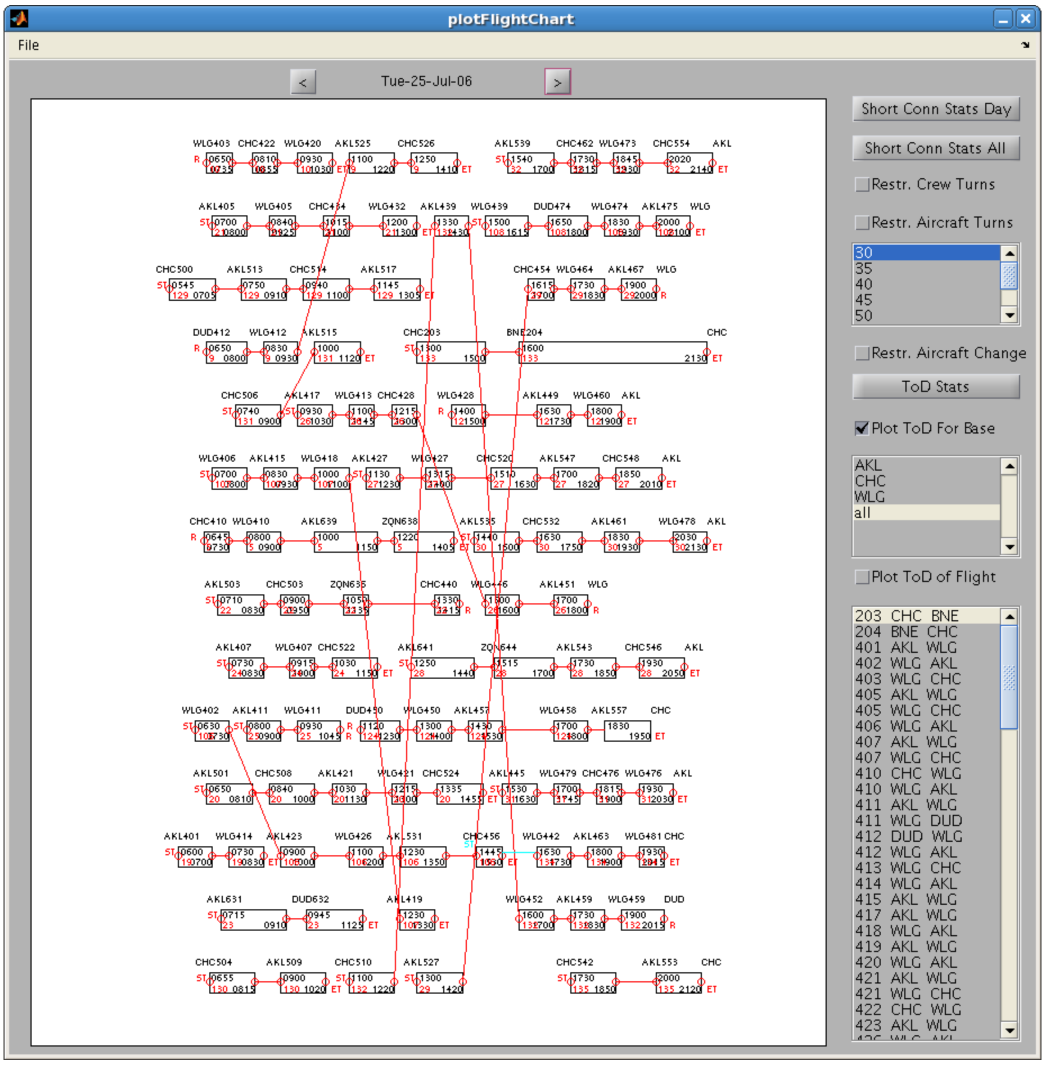


Figure 5.8. Screen-shot of traditional approach crew pairing solution for first officer scenario, summer 2006.



Restr. Crew Turns [ ]
Restr. Aircraft Turns [ ]
30 [✓] 35 40 45 50
Restr. Aircraft Change [ ]
Plot ToD For Base [✓]
AKL CHC WLG [✓ all]
Plot ToD of Flight [ ]

```
203  CHC  BNE
204  BNE  CHC
401  AKL  WLG
402  WLG  AKL
403  WLG  CHC
405  AKL  WLG
405  WLG  CHC
406  WLG  AKL
407  AKL  WLG
407  WLG  CHC
410  CHC  WLG
410  WLG  AKL
411  AKL  WLG
411  WLG  DUD
412  DUD  WLG
412  WLG  AKL
413  WLG  CHC
414  WLG  AKL
415  AKL  WLG
417  AKL  WLG
418  WLG  AKL
419  AKL  WLG
420  WLG  AKL
421  AKL  WLG
421  WLG  CHC
422  CHC  WLG
423  AKL  WLG
```

Figure 5.9. Screen-shot of iterative approach (iteration 5) crew pairing solution for first officer scenario, summer 2006.

Chapter 6

Robust and Integrated Aircraft Routing and Crew Pairing with Time Windows

In this chapter we describe a robust and integrated model for aircraft routing and crew pairing problems that also allows re-timing of departure times of flights in the schedule. The problem is called the *time window problem* because the departure time of each flight can vary within some specified interval, i.e. a time window. We formulate a model that integrates the three problems of aircraft routing, crew pairing, and time windows and propose two different solution approaches. We present computational experiments for both solution approaches and highlight problems and challenges.

In contrast to the robust and integrated problem described in the previous chapter, the type of problem considered in this chapter is a long term planning problem. The schedule is usually published at least six months or even a year prior to operation. Once the schedule is published only minor adjustments to the schedule can be made because passengers start booking flights and rely on the published departure times of the flights, e.g. because of business travel or international connections. The aircraft routing and crew pairing problems are solved much closer to the day of operations, e.g. two months in advance. We cannot change departure times of the flights at this time. Instead, we solve the robust and integrated problem with time windows at the time the schedule is constructed as a long term planning problem. We try to explore benefits in cost or robustness of slightly modified schedules compared to the originally proposed schedule. We expect the modified schedule to possess characteristics that will enable low cost and robust crew pairing and aircraft routing solutions. These solutions will be constructed with the iterative approach shortly before the day of operations, as discussed in the previous chapter.

6.1 Model

The robust and integrated aircraft routing and crew pairing problem with time windows is a generalisation of model (5.2). Flexibility is added to the departure times (and hence arrival times) of the flights in the schedule, i.e. the departure time of each flight can vary within some lower and upper bounds. We are solving the robust and integrated aircraft routing and crew pairing problem, and the model must ensure that the same departure time is assigned to the same flight in both problems. As in all previous models a dated schedule is considered. Although the schedule is different on each day of the week, a number of flights are offered by Air New Zealand at the same time each day, e.g. a business flight between the same origin and destination each morning at 7 o'clock. Some flights are repeated on every day of the schedule while others are only operated on some days, e.g. only on weekdays. As a requirement to the model, every flight that repeats throughout the schedule must depart at the same time. For example, the model may change the departure time of the business flight above to depart at 7:05 a.m., but it must do so on each day the flight is operated. We refer to these requirements as *schedule synchronisation constraints*. Additionally, we must ensure all departure times are within the applicable lower and upper bounds and all minimal turn-time and minimal sit-time requirements are met.

To incorporate departure time flexibility into the model, we add a set of variables $\boldsymbol{t}$ to the formulation: variable t_i represents the departure time of flight i. Variable t_i is limited by the lower and upper bound of the departure time window:

$$t_i^{min} \leq t_i \leq t_i^{max}. \tag{6.1}$$

Since we assume fixed flying time, the departure time also determines the arrival time of each flight. If connection ij is operated in the solution by some aircraft, we need to enforce the following inequality to ensure minimal turn-time restrictions are satisfied:

$$t_i + (flightTime_i + minTurnTime_{ij}) \leq t_j. \tag{6.2}$$

Equivalently, the same restrictions need to be satisfied for crew connections. If connection ij is operated by some crew, the following inequality must hold:

$$t_i + (flightTime_i + minSitTime_{ij}) \leq t_j. \tag{6.3}$$

To integrate these constraints into our model, we define an integer $m^T \times n^R$ matrix T^R. The value m^T is the number of all possible connections in the schedule and n^R the number of all feasible aircraft routings. Each column in matrix T^R corresponds to exactly one column of matrix A^R, where

$$(t_{\overline{ij},k})^R = \begin{cases} flightTime_i + minTurnTime_{ij} & \text{if connection } ij \text{ is contained in} \\ & \text{routing } k \\ -M & \text{otherwise,} \end{cases}$$

with $1 \leq \overline{ij} \leq m^T, 1 \leq k \leq n^R$. Note that we use value $\overline{ij}$ to identify the single row of matrix T^R that is associated with connection ij. A large constant M (called *big-M*) is chosen such that the inequality $t_i - M \leq t_j$ holds for all feasible connections ij, independently of the values of t_i and t_j. This ensures that if a connection ij is not part of the solution, constraints (6.2) are always satisfied. If connection ij is part of the solution, constraints (6.2) ensure that the MinTurnTime rule is satisfied.

We also define an $m^T \times m$ node-arc incidence matrix T, where m is the number of flights. Each row of matrix T represents a possible connection and each column represents a flight. Each row has exactly two non-zero entries: the value 1 in the column of the arriving flight and the value -1 in the column of the departing flight of the connection:

$$(t_{\overline{ij},k}) = \begin{cases} 1 & \text{if } k = i, \text{ i.e. flight } k \text{ is the arriving flight of connection } ij \\ -1 & \text{if } k = j, \text{ i.e. flight } k \text{ is the departing flight of connection } ij \\ 0 & \text{otherwise,} \end{cases}$$

with $1 \leq \overline{ij} \leq m^T, 1 \leq k \leq m$. For crew, we define an $m^T \times n^P$ matrix T^P analogously to T^R:

$$(t_{\overline{ij},k})^P = \begin{cases} flightTime_i + minSitTime_{ij} & \text{if connection } ij \text{ is contained in} \\ & \text{pairing } k \\ -M & \text{otherwise,} \end{cases}$$

with $1 \leq \overline{ij} \leq m^T, 1 \leq k \leq n^P$. Constant M ensures that constraints (6.3) are satisfied if connection ij is not part of the solution, otherwise the constraints ensure that the MinSitTime rule is satisfied.

In order to satisfy the schedule synchronisation constraints, we partition the flights in the schedule into groups of flights G that must depart at the same time. We add one departure time variable τ_g for each group of flights $g \in G$ determining the departure time of all flights in the group and we add the following constraints to the model:

$$t_i = \tau_g, \qquad \text{for all } g \in G \text{ and all } i \in g.$$

We define an $m \times |G|$ matrix S that maps the time variables to the appropriate group:

$$(s_{ij}) = \begin{cases} 1 & \text{if flight } i \text{ is in group } j \\ 0 & \text{otherwise,} \end{cases}$$

with $1 \leq i \leq m, 1 \leq j \leq |G|$. With this matrix representation the *robust and integrated aircraft routing and crew pairing problem with time windows* can be formulated as follows:

$$\begin{array}{lccccccccccl}
\text{Minimise} & (\boldsymbol{c}^P)^T\boldsymbol{x}^P & + & (\boldsymbol{c}^R)^T\boldsymbol{x}^R & + & (\boldsymbol{c}^D)^T\boldsymbol{d} & & & & & & \\
\text{subject to} & A^P\boldsymbol{x}^P & & & & & & & & = & \mathbb{1} & \\
 & & & A^R\boldsymbol{x}^R & & & & & & = & \mathbb{1} & \\
 & B^P\boldsymbol{x}^P & - & B^R\boldsymbol{x}^R & & & & & & \leq & 0 & \\
 & D^P\boldsymbol{x}^P & - & D^R\boldsymbol{x}^R & - & \boldsymbol{d} & & & & \leq & 0 & \\
 & T^P\boldsymbol{x}^P & & & & & - & T\boldsymbol{t} & & \leq & 0 & \\
 & & & T^R\boldsymbol{x}^R & & & - & T\boldsymbol{t} & & \leq & 0 & \\
 & & & & & & & \boldsymbol{t} & - S\boldsymbol{\tau} & = & 0 & \\
 & & & & & \boldsymbol{t}^{min} & \leq & \boldsymbol{t} & & \leq & \boldsymbol{t}^{max}, &
\end{array} \tag{6.4}$$

where $\boldsymbol{x}^P \in \{0,1\}^{n^P}$, $\boldsymbol{x}^R \in \{0,1\}^{n^R}$, $\boldsymbol{d} \in \{0,1\}^{m^D}$ are binary variables and $\boldsymbol{t} \in \mathbb{Z}^m$ and $\boldsymbol{\tau} \in \mathbb{Z}^{|G|}$ are integer departure time variables. All departure times are represented as minutes from the start of the day.

The first four sets of constraints and associated variables are identical to those in Model (5.2). Constraint sets five and six ensure that minimal turn-time and minimal sit-time requirements are met, respectively. Constraints seven are the schedule synchronisation constraints and the last set of constraints enforces the lower and upper bounds on all departure time variables.

Variables $\boldsymbol{\tau}$ are included in the model for simplicity and can be removed from the formulation and expressed by variables $\boldsymbol{t}$. The time variables can be disaggregated from one variable for each flight to one variable for each connection to avoid the big-M constraints, see van Eijl [1995]. We do not present this formulation since the big-M constraints are not included in our solution approaches, as discussed in Section 6.3. Instead, the constraints are implicitly satisfied in the pairing and routing generation subproblems. Note that other departure time dependant restrictions such as maximal duty time limits are also implicitly satisfied in the column generation problems.

6.2 Rules

The following restrictions for re-timing flights are imposed:

- AklWlg
  Departure times for all flights between the airports Auckland and Wellington (in both directions) are fixed to the original departure time. This restriction is a requirement imposed by Air New Zealand. All other flights are allowed to be re-timed within a window of $\pm$ 10 minutes of the original departure time. All departure times occur at 5 minute intervals starting from midnight.

- FollowThrough
  A pair of follow-through flights must be re-timed by the same amount. The turn-time between two follow-through flights is usually equal to the minimal turn-time and must remain constant because passengers expect minimal turn-times between follow-through flights.

- Synchronisation
  Flights with the same flight number and origin must depart at the same time on each weekday. On the weekend, however, the flights with the same flight number and origin are allowed to depart at different times. We refer to a group of flights that must depart at the same time on different days as a *departure time group*. Additionally, aircraft, operating crew, and passengering crew that operate the same flight must all depart at the same time. Since the departure time of a flight is only determined by the routing and pairing that contain the flight, we must make sure that all routings and pairings use the same departure time for all flights in the same departure time group.

The departure time flexibility is incorporated into the shortest path algorithms of aircraft routing and crew pairing problems. The departure time of each flight contained in a routing or pairing is determined by the labelling algorithm: an attribute is associated with the departure time of each flight. Similarly to deciding when a meal break occurs during a pairing, it is decided in the labelling process which departure time attribute is chosen for each flight. The attributes that are stored at a label determine the departure times of all flights contained in the path that is associated with the label.

The algorithm finds re-timings for all flights in the routing or pairing that result in the most negative reduced cost. Different routings or pairings covering the same flight may assign different departure times to the flight. Additionally, a pairing that contains a crew that is passengering on a flight may determine a different departure time for that flight than a pairing in which the flight is operated. We propose two solution methods that ensure that the departure times determined by a solution are the same for all flights in the same departure time group.

6.3 Solution Methods

Solving the robust and integrated model with time windows is challenging for the problem instances we consider. The main reason is that synchronisation constraints are needed to ensure the same departure time for all flights in the same group on different days. Since the schedule is different on every day, a particular re-timing may yield an improvement of crew pairing cost or robustness on one day, but may worsen the solution or render the solution infeasible on another day. Clearly, a daily problem is much easier to solve because such constraints are not needed. In this section we present two optimisation based heuristic solution approaches: *time window branching* approach and *re-timing of flights for fixed aircraft routings and crew pairings* approach.

Other recent approaches considering time windows include Ioachim et al. [1999] and Mercier and Soumis [2007]. Ioachim et al. [1999] solve the fleet assignment and aircraft routing problem with time windows and synchronisation constraints. They use stepwise linear cost functions on the time windows for the shortest path calculations and find that this approach works better than discretised departure times for very large time windows (100 minutes and more). For small time windows however, faster solution times are achieved with discretised departure times. Mercier and Soumis [2007] include time windows in their integrated aircraft routing and crew pairing model. They consider time windows of $\pm$ 5 minutes around the original departure time and add binary variables that indicate if the flight leaves 5 minutes earlier, later, or at the same time as originally scheduled. Synchronisation constraints are added to the model of Cordeau et al. [2001] to ensure the same departure time is used for aircraft, crew, as well as deadheading crew. Because of the additional binary variables the number of short connection constraints is much larger than in the original formulation. Mercier and Soumis [2007] propose an equivalent model aggregating the short connection constraints but in our opinion the feasible solution space is reduced by the aggregation. They do not consider robustness in the approach and solve a daily problem, so no further synchronisation constraints are needed to ensure the same departure time of flights that are repeated on multiple days. They use Benders decomposition and the three phase solution approach as described in Cordeau et al. [2001] and Mercier et al. [2005] to solve the problem.

6.3.1 Time Window Branching Approach

Starting from a solution of the iterative approach, we allow time windows for departure times of the flights and solve crew pairing and aircraft routing independently. This yields a new (possibly infeasible) integrated solution with *un-synchronised* departure times. In each routing or pairing that is covering a flight, the departure time for the flight is determined by the particular routing or pairing. Hence, departure times may be different for aircraft, operating crew, and passengering crew as well as for flights of the same departure time group on different days of the week. We penalise deviations from the original departure times of the flights in the objective function by linearly increasing penalties. We hereby avoid a large number of unnecessarily re-timed flights that do not yield an improvement in crew pairing cost or aircraft change cost. The ratio of weights between crew pairing cost and robustness in the objective function remain the same as used for the iterative approach starting solution. The un-synchronised solutions usually show significant gains in cost and robustness compared to the starting solution of the iterative approach.

If in an un-synchronised solution, departure times are different for flights within the same departure time group, we employ a branching scheme on the time windows to synchronise the departure times, but only in the aircraft routing problem, i.e. we force all departure times for flights in the same departure time group to be equal in the aircraft routing solution. We guide this branching procedure in the aircraft routing problem by the departure times of the operating crew: for each departure time group we find the departure time that is preferred by crew, that is the departure time over all flights of the group that is used the most by crew. In case of a tie, we choose the departure time that is closest to the original departure time. We then force this common departure time for all flights of the group by setting the time window bounds accordingly. All other flights are allowed to be re-timed but each re-timing is penalised in the objective function. We solve the aircraft routing problem for the new time windows. If no feasible solution can be found, we increase the time windows of the flights with fixed departure time in 5 minute steps until we find a feasible aircraft routing solution. Subsequently, we branch on time windows for which the aircraft routing solution uses different departure times for flights of the same departure time group.

Once all departure times for all groups are synchronised, we use this aircraft routing solution and solve the crew pairing problem once. All departure times in the crew pairing problem are fixed to the departure times used in the aircraft routing problem. Once the crew pairing problem is solved, we obtain a robust and integrated aircraft routing and crew pairing solution where some of the departure times of flights may differ from the original departure times.

We use this heuristic procedure because branching on time windows is difficult and time consuming. We would prefer to incorporate both the aircraft routing and crew pairing problems within the time window branching process and explore the branch-and-bound tree until an optimal solution is found. However, this is very time consuming for two reasons. Firstly, each iteration takes a considerable amount of time. Note that because of the departure time flexibility the run times of the column generation problems increase greatly since many more labels exist at each node. Secondly, branching on time windows is difficult, especially when time windows are large, and leads to a large number of branch-and-bound nodes that must be explored. Ideally, we want to make only few branching decisions that lead to a good quality solution. We discuss some difficulties with such a branching procedure in the following:

- It is not obvious what a good branching strategy might be. In the following examples each table lists the departure times of flights of one departure time group. The top line of each table shows the possible offsets in minutes from the original departure time, while the bottom line displays how many flights of the departure time group depart with each offset. Suppose 7 flights depart 15 minutes earlier and 1 flight departs 10 minutes earlier than originally scheduled. Then, the table showing the offsets has the following form:

offset	−15	−10	−5	0	5	10	15
flights	7	1	0	0	0	0	0

In this case the branch is explored first where all flights must depart 15 minutes earlier than originally scheduled. However, if departure times are obtained as in the following examples, the branching decision becomes much more difficult:

offset	−15	−10	−5	0	5	10	15
flights	0	5	0	0	2	2	2

offset	−15	−10	−5	0	5	10	15
flights	1	1	1	1	1	1	1

offset	−15	−10	−5	0	5	10	15
flights	1	0	0	0	0	1	1

A sensible branching strategy in order to divide a time window into 2 smaller intervals might be the following. For each possible partition of the time window calculate the ratios between the number of departures in a sub-interval and the number of different offsets that contribute to the sub-interval for each side of the partition. The sub-interval with the largest ratio over all possible partitions is explored first. The example

offset	−15	−10	−5	0	5	10	15
flights	0	0	5	0	0	0	3

shows that one has to decide if the contributing offsets of the sub-interval for the ratio calculation is determined from the beginning and end of the time window or if one starts counting from the first nonzero entry in the partition. In the first case the 6 possible partitions result in the ratios $(0/1, 8/6), (0/2, 8/5), (5/3, 3/4), (5/4, 3/3), (5/5, 3/2)$ and $(5/6, 3/1)$ and, hence, the preferred branch is the second interval of

offset	−15	−10	−5	0	5	10
flights	0	0	5	0	0	0

and

offset	15
flights	3

.

In the latter case, however, the preferred branch is the first interval of

offset	-15	-10	-5
flights	0	0	5

and

offset	0	5	10	15
flights	0	0	0	3

with a largest ratio of 5/1. Note that branching on time-windows is equivalent to branching on *special ordered sets*. Special ordered sets are introduced in Beale and Tomlin [1970] and are used to branch on sets of variables rather than on individual variables. The departure time variable of a single flight can be expressed as a special ordered set of binary variables, each representing a single departure time. For a feasible solution, exactly one variable must have value 1 and all other variables must have value 0. During each branching decision a subset of these variables is set to 0. Because of these similarities the extensive literature on special ordered sets should be analysed thoroughly in a future research project in order to improve the branching decisions on the time-windows.

- It is unclear how to decide how restrictive the branches should be, i.e. how small the time window in the preferred branch should be. More restrictive branching quickly leads to infeasibility in the aircraft routing or crew pairing problem. If less restrictive branches are employed branching multiple times on the same time window may be necessary resulting in many nodes that must be explored. Note that for a weekly schedule, there are around 120 departure time groups that contain more than 1 flight and hence many nodes must be explored even if there is only a single branch required for each time window.

- It is very difficult to predict the impact on crew pairing cost and aircraft change cost an imposed branch may have. Re-timings that yield improvements on some days of the week may cause much worse or infeasible solutions on other days of the schedule. To keep crew pairing costs low, the branching decisions could take only (or mostly) crew pairing solutions into account. However, in our experiments this strategy quickly leads to infeasibility of the aircraft routing problem.

- After solving the un-synchronised problem, one can shrink the time windows to only include departure times that are actually used by the solution. This decreases the number of branches required but quickly leads to infeasibility.

To overcome these difficulties the following two strategies seem to be useful:

- We introduce costs on the time windows to guide the branching process. If o_i denotes the difference in time compared to the original departure time of flight i, costs can be used to discourage any re-timing ($cost = k * abs(o_i)$) for some positive constant k or to force all flights in a departure time group to depart at a certain time t within the window ($cost = k * abs(o_i - t)$). This strategy proves to be very useful in finding good quality solutions quickly because it results in less re-timings and the re-timings are more homogeneous across the same departure time group.

- Only considering the aircraft routing problem within the time window branching process is promising. For our problem instances the aircraft routing problem is easier and hence faster to solve than the crew pairing problem. Also, the aircraft routing problem is much more likely to become infeasible within the branching process than the crew pairing problem because of the limited number of available aircraft. If the branching decisions are guided by the crew pairing departure times of the un-synchronised version, the synchronised and re-timed solution of the aircraft routing problem generally leads to a crew pairing solution with low crew pairing cost and aircraft change cost. This, however, cannot be guaranteed.

Implementation

Compared to the approaches described in previous chapters, significant changes are made to aircraft routing and crew pairing algorithms to implement the time window branching approach. A large amount of code development is needed to allow flexible departure times in crew pairing and aircraft routing optimisers. All data structures and functions of the commercial crew pairing solver were based on the assumption that the departure time of a flight is determined by the flight itself. This assumption is no longer valid in our implementation of the algorithm. Instead, the departure time of each flight is determined in the label setting resource constrained shortest path algorithm. Each path represents a pairing and contains the sequence of flights as well as the departure time for each flight. These departure times must be considered in many rule and cost calculations which requires code changes to most parts of the commercial crew pairing optimiser. All restrictions on the departure times are satisfied by the shortest path algorithm.

6.3.2 Re-timing of Flights for Fixed Aircraft Routings and Crew Pairings

In this section we describe how to find a re-timed solution that incurs minimal aircraft change cost for fixed aircraft routings and crew pairings of a solution of the iterative approach. We do not change the sequences of flights in the routings and pairings and hence we do not alter the total number of aircraft changes in the solution but improve aircraft change costs by increasing buffer times. Whenever it is possible to re-time a flight before or after a restricted aircraft change, we re-time this flight in order to improve aircraft change cost. If the crew pairings are still feasible, the crew pairing cost of the re-timed solution only changes very slightly as long as the changes in departure times are not too large.

Re-timing flights for fixed aircraft routing and crew pairing solutions to improve aircraft change cost is very easy compared to the problem formulated in the previous section. As all connections operated by crew and aircraft are given, the effects of re-timing a particular flight on all other flights in the schedule can easily be calculated. We formulate the problem of finding a re-timing of flights that incurs minimal aircraft change costs for given and fixed aircraft routings and crew pairings as an integer program in the following way.

We define integer variables o for the offset from the original departure time for each departure time group. Each variable o_i is bounded by the time window imposed on departure time group i and must be an integer multiple of 5 minutes. Additional binary variables q are used to penalise restricted aircraft changes.

The constraints are constructed by considering all connections operated by aircraft and crew in the given solution: for all departure time groups i and j operated in sequence by some aircraft we add a constraint

$$o_i = o_j + (connectionTime_{ij} - minTurnTime_{ij}),$$

if connection ij is a follow-through connection (as we are not allowed to change the duration of this connection) and

$$o_i \leq o_j + (connectionTime_{ij} - minTurnTime_{ij}),$$

otherwise. The value of $connectionTime_{ij}$ refers to the connection duration without re-timings. Time $(connectionTime_{ij} - minTurnTime_{ij}) \geq 0$ is the buffer by which we can shorten connection ij without violating the minimal turn time. Note that i can be followed by multiple departure time groups j because different connections may be operated by aircraft on different days.

For all departure time groups i and j operated in sequence by some crew we add the following constraints if connection ij is an aircraft change:

$$o_i = o_j + (connectionTime_{ij} - minSitTime_{ij}) - \sum_{k=0}^{l} f_k q_k^{ij},$$

and

$$\sum_{k=0}^{l} q_k^{ij} = 1,$$

Binary variables q_k^{ij} are used to penalise the aircraft change in the objective function if the connection time is smaller than the restricted time. The coefficients are defined by $f_k = 5k$ since all departure times occur at 5 minute intervals. Index $k \in \{0, \dots, l\}$ is used to determine by how many minutes ($5k$) the minimal turn-time is exceeded. The constant l is determined a priori such that $5l$ is equal to the maximal possible duration (in minutes) of any aircraft change connection: $l = (abs(min(o_i)) + abs(max(o_j)) + restrictedTime)/5 + 1$. Functions $min(o_i)$ and $max(o_i)$ determine the minimal and maximal possible value of o_i, respectively. The GUB constraint $\sum_{k=0}^{l} q_k^{ij} = 1$ ensures that exactly one variable $q_{k'}^{ij}$ for some $k' \in \{0, \dots, l\}$ equals 1 and all others are equal to 0. This is equivalent to the duration of the re-timed connection exceeding the minimal sit time by $5k'$ minutes. We add positive (linearly decreasing with increasing time) cost c_k^{ij} to the objective function for each q_k^{ij} if $5k < restrictedTime$, otherwise the cost c_k^{ij} of q_k^{ij} is set to 0.

If connection ij is not an aircraft change we add constraint

$$o_i \leq o_j + (connectionTime_{ij} - minSitTime_{ij}),$$

similarly to aircraft connection constraints.

All constraints combined form the *re-timing problem* as follows:

$$\begin{array}{llrll}
\text{Minimise} & & \displaystyle\sum_{ij\in AC^P}\sum_{k=0}^{l} c_k^{ij} q_k^{ij} & & \\
\text{subject to} & o_i - o_j - (ct_{ij} - mtt_{ij}) & & = 0 & \forall ij \in TC^R \\
& o_i - o_j - (ct_{ij} - mtt_{ij}) & & \leq 0 & \forall ij \in C^R \setminus TC^R \\
& o_i - o_j - (ct_{ij} - mst_{ij}) & \displaystyle + \sum_{k=0}^{l} f_k q_k^{ij} & = 0 & \forall ij \in AC^P \\
& & \displaystyle\sum_{k=0}^{l} q_k^{ij} & = 1 & \forall ij \in AC^P \\
& o_i - o_j - (ct_{ij} - mst_{ij}) & & \leq 0 & \forall ij \in C^P \setminus AC^P,
\end{array}$$

where C^R is the set of all connections operated in the solution by an aircraft and TC^R the set of follow-through connections. Sets C^P and AC^P denote all connections operated by crew and all aircraft change connections, respectively. Values ct_{ij}, mtt_{ij}, and mst_{ij} denote the connection time, minimal turn time and minimal sit time of connection ij, respectively. All other parameters and variables are defined as above. The objective minimises the total aircraft change cost of the solution. The first set of constraints ensures that the connection times of all follow-through connections remain constant. The second set ensures that the minimal turn time rule is obeyed for all other aircraft connections. Constraint sets three and four impose a penalty in the objective function for aircraft change connections where the sit time is less than the restricted time and the last set of constraints ensures that the minimal sit time rule is obeyed by all connections operated by crew.

By substituting variables o with $o = o^+ - o^-, o^+ \geq 0, o^- \geq 0$ we can add linearly increasing penalties to the objective function for increasing deviation from original departure times. The model then yields a solution with small aircraft change cost that also only re-times as few flights as possible.

Since the only rules considered by the re-timing IP involve minimal turn-times and minimal sit-times, we must make sure that other rules such as meal breaks or maximal duty time limits are not violated. We check the solutions with the crew pairing solver for feasibility (the re-timings are guaranteed to be feasible for the aircraft routing problem). In our experiments all solutions turn out to be feasible crew pairing solutions. If this is not the case, we change the re-timing limits for an infeasible crew pairing, re-solve the re-timing IP and check the new solution for feasibility. Meal break rules and duty time limits can be added as constraints to the IP if necessary.

Implementation

This simple approach proves to be very effective. Very little implementation effort to construct the re-timing IP is needed in order to improve the robustness of the solutions by orders of magnitude. Note that no time window coding is necessary in the crew pairing solver or the aircraft routing optimiser. We find solutions that are optimal for a weighted sum objective function of aircraft change costs and penalties for re-timing flights for given aircraft routings and crew pairings. Note that there is hardly any computation time needed to solve the re-timing IP to optimality (with CPLEX). As a drawback we do not know the solution quality compared to a globally re-timed optimum where aircraft routings and crew pairings are allowed to be changed.

6.4 Computational Experiments

In this section we present computational experiments of solving the time window problem. We show results of applying the time window branch-and-bound approach and the re-timing approach to the first officer scenarios of schedule winter 2005 and summer 2006.

Figure 6.1 shows the results for re-timing solutions of the iterative approach for the first officer scenario, schedule summer 2006, 7 days. Original iterative approach solutions without re-timing are represented as blue squares. Red diamonds show a solution with identical aircraft routings and crew pairings but with an optimal re-timing with respect to a weighted sum objective function of aircraft change cost and re-timing penalties. Departure times for all AklWlg flights are fixed while all other flights can be re-timed by ± 10 minutes around the original departure time. The aircraft change cost decreases by around 30% for all solutions (e.g. from 93 to 62 for the solution of iteration 4). Only a modest amount of re-timings (e.g. 46 flights in 23 departure time groups in iteration 4) is required to achieve this improvement.

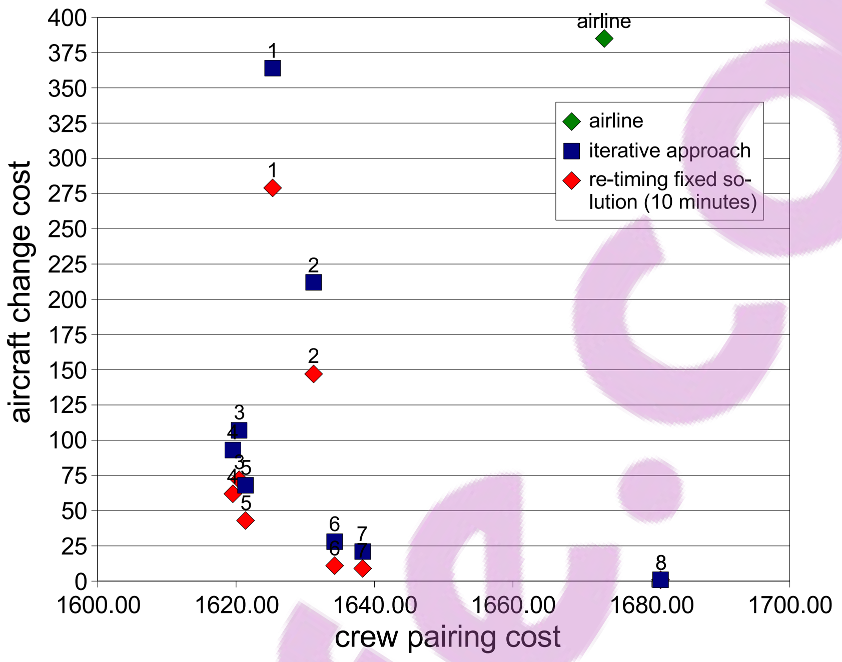


Figure 6.1. Re-timed solutions of iterative approach, AklWlg flights fixed, ±10 minute windows, first officer scenario, summer 2006, 7 days.

To estimate the full potential of this method we relax the AklWlg rule and allow ±10 minute time windows for these flights as well. The results are shown in Figure 6.2. The robustness of all solutions improves dramatically. Aircraft change costs are roughly reduced by a factor of 3 (e.g. aircraft change cost decreases from 93 to 27 for the solution of iteration 4). Since the AklWlg rule was imposed by Air New Zealand in an attempt to describe important flights which cannot be re-timed, this rule could be revised. As an example we could only fix business flights in the morning and the afternoon or assign a different time window to each individual departure time group.

Tables 6.1 and 6.2 list statistics for re-timed solutions with and without the possibility of re-timing AklWlg flights for summer 2006 and winter 2005 scenarios. The first column displays the iteration number of the iterative approach. The next two columns show crew pairing costs ("c^P") and aircraft change costs ("c^{AC}") for the iterative approach and the airline integer solutions. The next four columns show results for the scenario when departure times for

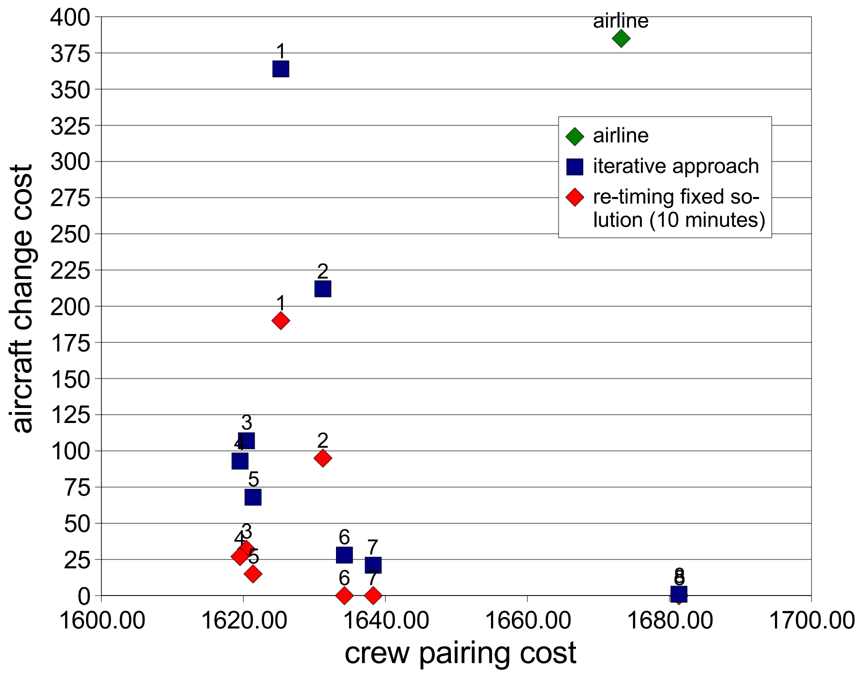


Figure 6.2. Re-timed solutions of iterative approach, AklWlg flights flexible, ±10 minute windows, first officer scenario, summer 2006, 7 days.

AklWlg flights are fixed. Column "c^{AC}" shows the aircraft change cost and columns "groups", "flights", and "mins." display the number of re-timed groups and flights, and the sum of re-timing minutes over all flights for each solution, respectively. The same values are shown in the next four columns for the scenario when AklWlg can also be re-timed within a window of ± 10 minutes around the original departure time. We observe that aircraft change costs can be reduced significantly by only re-timing a small number of flights by a few minutes. Results look similar for both scenarios for fixed AklWlg flights. Results are not as good for the winter 2005 scenario since for flexible AklWlg flights, a much larger number of flights is re-timed in this scenario to reach a similar level of robustness as in the summer 2006 scenario.

Figure 6.3 shows two solutions for the time window branch-and-bound approach for the scenario of 2006. We run the approach twice, starting from solutions of iterations 3 and 5 of the iterative approach and hence the value of p is set to 5 and 20, respectively. Solution 3 of the time window branch-and-bound approach incurs 1.83% less cost than the solution of the iterative approach with the lowest crew pairing cost. The gap for the corresponding LP

	iter. approach		AklWlg fixed				AklWlg flexible			
	c^P	c^{AC}	c^{AC}	groups	flights	mins.	c^{AC}	groups	flights	mins.
airline	1673.21	385	-	-	-	-	-	-	-	-
1	1625.28	364	279	47	139	1110	190	36	127	990
2	1631.20	212	147	45	131	1075	95	26	81	655
3	1620.44	107	72	27	62	495	32	14	38	275
4	1619.53	93	62	23	46	375	27	14	34	285
5	1621.36	68	43	21	42	335	15	10	21	155
6	1634.23	28	11	18	31	245	0	5	8	55
7	1638.29	21	9	13	26	205	0	4	7	45
8	1681.33	1	1	0	0	0	0	0	0	0

Table 6.1. Results for re-timed solutions of iterative approach for first officer scenario, summer 2006, 7 days.

	iter. approach		AklWlg fixed				AklWlg flexible			
	c^P	c^{AC}	c^{AC}	groups	flights	mins.	c^{AC}	groups	flights	mins.
airline	1684.42	374	-	-	-	-	-	-	-	-
1	1636.52	289	226	45	139	1120	152	114	350	2630
2	1630.94	205	156	38	130	990	93	104	333	2595
3	1629.31	165	130	24	75	555	68	80	233	1780
4	1634.70	112	80	24	77	580	37	75	225	1610
5	1645.23	58	42	16	43	315	15	42	113	840
6	1654.36	47	35	12	44	325	13	33	105	810
7	1670.09	17	13	4	8	45	3	16	40	310
8	1700.83	1	1	0	0	0	0	1	1	5

Table 6.2. Results for re-timed solutions of iterative approach for first officer scenario, winter 2005, 7 days.

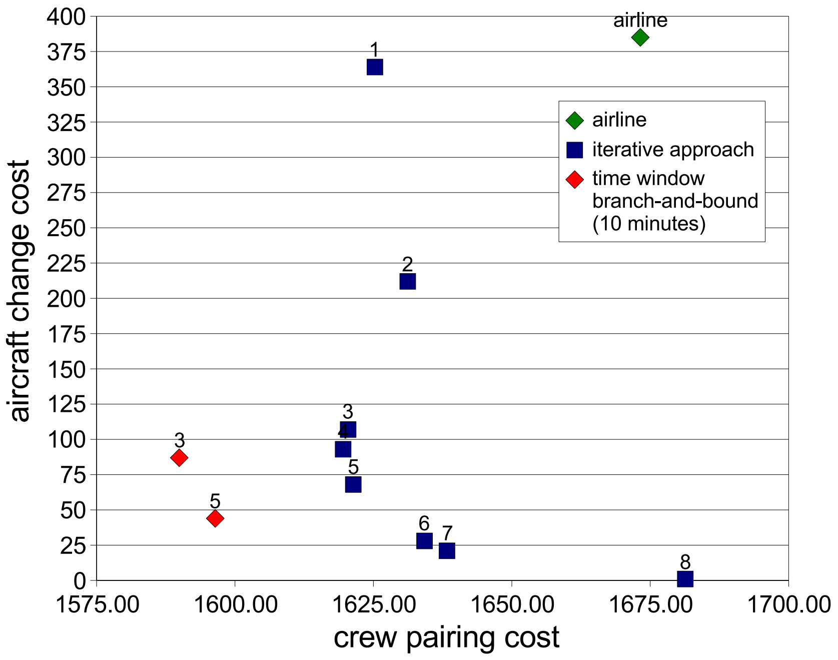


Figure 6.3. Time window branch-and-bound solutions of iterative approach solutions (iterations 3 and 5), AklWlg flights fixed, ±10 minute windows, first officer scenario, summer 2006, 7 days.

solution values is even bigger with 2.4%. The value of the un-synchronised IP solution incurs 2.3% less cost than the best solution of the iterative approach. When we solve the aircraft routing problem where departure times are driven by the departure times that are operated by the majority of the crew, no branching on time windows is necessary in the aircraft routing problem since all departure times are equal for flights within the same group. This is caused by penalising deviations of departure times in the aircraft routing problem from the departure times that are operated mostly by the crew. The running time is 1325 seconds. Only four departure time groups are re-timed for this solution, two groups containing five flights where each flight is re-timed by 10 minutes, and two groups containing two flights where each flight is re-timed by only 5 minutes. The cost saving results from four duty periods that are required less in the re-timed solution.

The results for the winter 2005 scenario are not as promising. Although the un-synchronised solution incurs 2.2% less cost than the minimal crew pairing cost solution of the iterative approach, no solution with synchronised departure times can be found that incurs less costs than the solutions of the iterative approach. The time window branching procedure evaluates 13 nodes before the departure times in the aircraft routing problem are synchronised. The crew pairing problem for this set of re-timed flights does not yield any improvements in crew pairing cost. The departure times the crew would “like” to operate are infeasible from an aircraft routing perspective and hence it is difficult to find departure times in the aircraft routing problem that are “similar” to the ones preferred by crew. The set of re-timings finally determined in the aircraft routing problem does not allow a cost decrease for the crew pairing solution. For a single day however, crew pairing costs can be reduced by 4.6% compared to the best solution of the iterative approach. This is achieved by re-timing one flight by 5 minutes and five flights by 10 minutes. The improvement is enabled by only solving a single day and hence not needing any further synchronisation constraints. This demonstrates the difficulties caused by synchronisation constraints for solving the time window problem over multiple days for a varying schedule.

These results show that solutions can be greatly improved by changing the departure times of only very few flights. This, however, can only be achieved if the departure times that are preferably operated by the crew are likely to result in a feasible (or almost feasible) aircraft routing solution. If departure times of many other flights must be changed to enable a feasible and synchronised aircraft routing solution it is likely that the positive effects on the crew pairing costs of some days are annihilated by negative effects on other days. If no improved synchronised solution can be found, the un-synchronised solution can nevertheless give good insights into the structure of a schedule that is efficient to operate from a crew pairing cost and robustness perspective.

Figure 6.4 shows one day of solution 3 of the time window branch-and-bound approach with AklWlg flights fixed, 10 minute windows, for the first officer scenario, summer 2006. As described above, 2 time groups are re-timed by 10 minutes in this solution, resulting in 10 re-timed flights, 2 on each weekday. The screen-shot displays the 2 flights that are departing 10 minutes later than originally scheduled in pink. The re-timing enables a feasible crew connection between flight 402 from WLG (Wellington) to AKL (Auckland) and flight 513 from AKL to CHC (Christchurch). Screen-shot 6.5 shows the solution without re-timing. Flight 402 is connected to flight 415 which results in a very short and, hence, inefficient crew pairing. Additionally, the crew must stay overnight in Wellington before operating flight 402. The same behaviour can be observed on 4 of the 5 weekdays. By re-timing the flights as shown in Figure 6.4 the short and inefficient crew pairing can be eliminated from the solution and hence the solution requires 4 man days less to operate the schedule. This example demonstrates clearly how the cost of the solution can be improved by almost 2% by making only minor adjustment to the schedule.

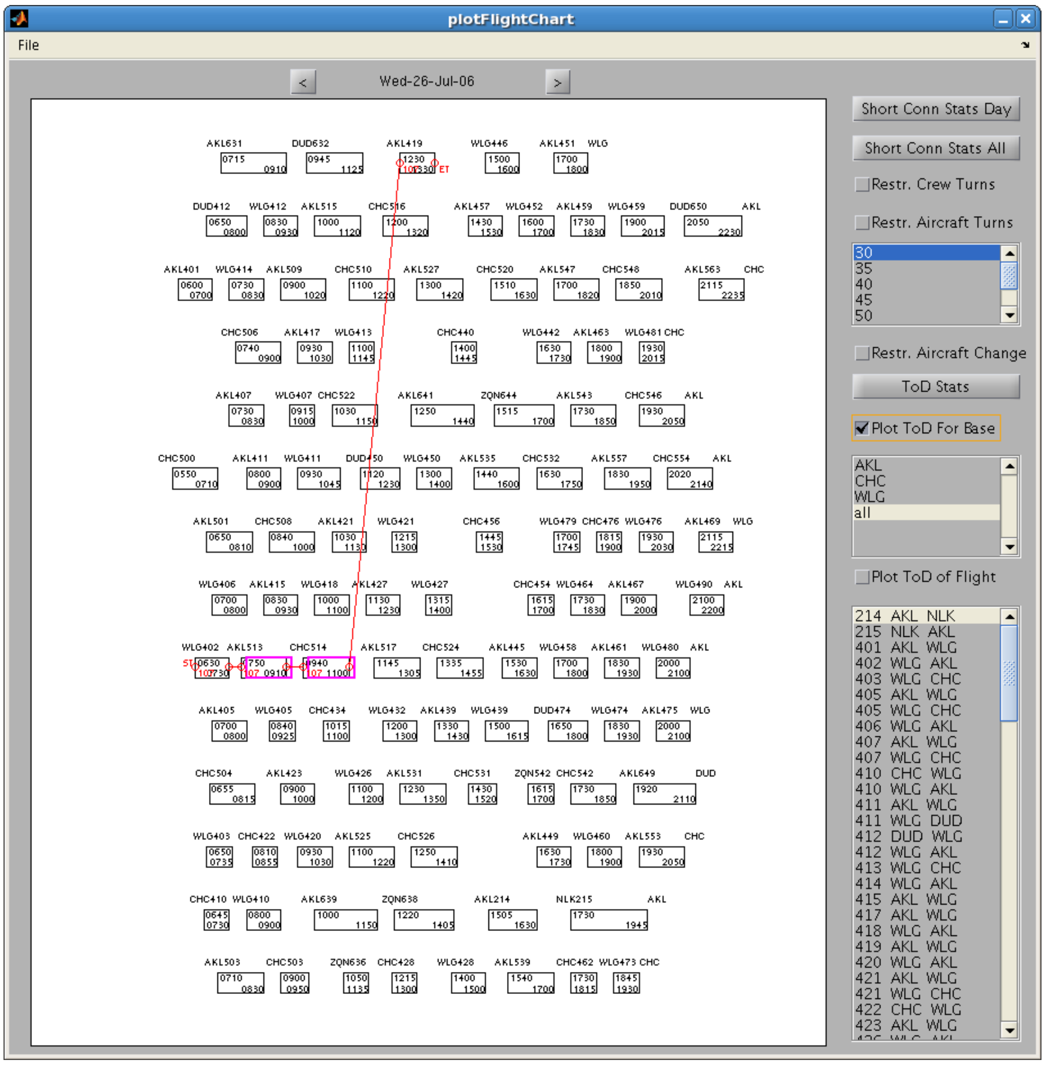
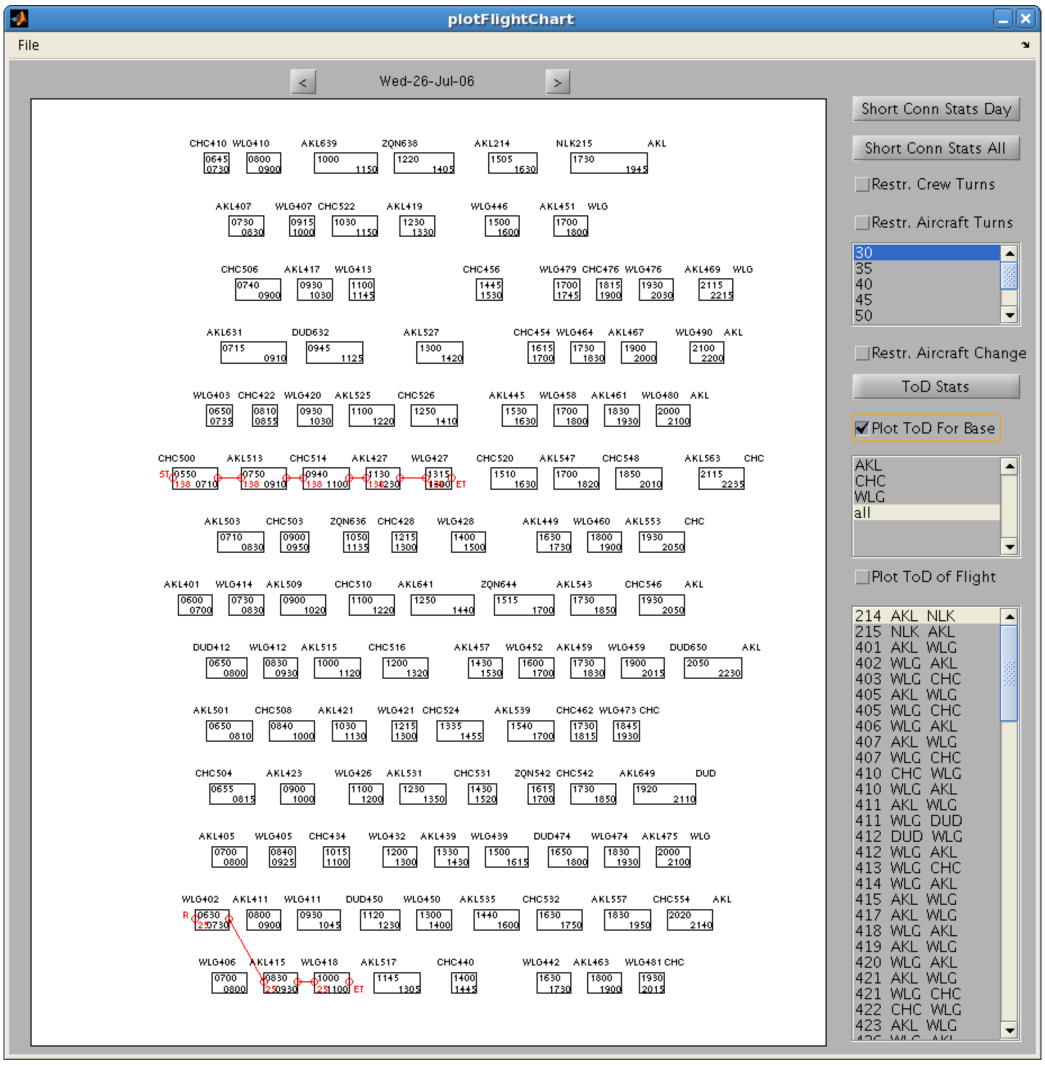


Figure 6.4. Screen-shot of time window branch-and-bound solution (iteration 3), AklWlg flights fixed, ±10 minute windows, for first officer scenario, summer 2006.



```
214  AKL  NLK
215  NLK  AKL
401  AKL  WLG
402  WLG  AKL
403  WLG  CHC
405  AKL  WLG
405  WLG  CHC
406  WLG  AKL
407  AKL  WLG
407  WLG  CHC
410  CHC  WLG
410  WLG  AKL
411  AKL  WLG
411  WLG  DUD
412  DUD  WLG
412  WLG  AKL
413  WLG  CHC
414  WLG  AKL
415  AKL  WLG
417  AKL  WLG
418  WLG  AKL
419  AKL  WLG
420  WLG  AKL
421  AKL  WLG
421  WLG  CHC
422  CHC  WLG
423  AKL  WLG
```

Figure 6.5. Screen-shot of iterative approach solution (iteration 3) without time windows for first officer scenario, summer 2006.

Conclusion

We present a model and two new solution methods to solve the robust and integrated aircraft routing and crew pairing problem. We propose an iterative approach that is coupling the two problems heuristically and can quickly generate a series of solutions with low crew pairing costs and low aircraft change costs. We therefore expect the solutions to be operationally robust. Additionally, no monetary value needs to be attached to robustness a priori. Instead, the trade-off between costs and robustness can be observed and a preferred solution can be implemented. Although optimality of the solutions cannot be guaranteed, a lower bound on the optimal crew pairing cost is provided by the algorithm. We obtain solutions that incur less crew pairing costs and are significantly more robust than solutions currently used in practice. This is a great improvement compared to the sequential method where the crew pairing problem is solved for a fixed aircraft routing solution. We have seen in Chapter 4 that in such a solution approach robustness can only be improved by accepting an increase in crew pairing cost. In an extension of the iterative approach, we are able to consider multiple crew groups with only minor modifications.

We propose a Dantzig-Wolfe decomposition approach to solve the robust and integrated aircraft routing and crew pairing problem to optimality. Solving the problem to optimality is computationally expensive. Also, to identify a robust solution in the approach, we need to associate a monetary value with non-robustness. The run times of an optimisation approach are much longer than the run times of the iterative approach. This is the case even though only a single problem is solved in the optimisation approach with fixed weights for cost and robustness while the iterative approach generates multiple solutions with varying trade-off between cost and robustness. The optimisation approach is useful in determining that the iterative approach solutions are of very good quality with an average optimality gap over all problem instances of less than 1%. The iterative approach can be substituted by an optimisation approach once run times decrease due to improvements in the optimisation algorithm or computer hardware. It remains complicated to incorporate a rule such as limiting the number of aircraft changes per duty period (DPACLim rule) into any optimisation approach because the rule requires to compare individual routings and pairings. Due to the results presented in this thesis the rule was relaxed by Air New Zealand and robustness is now ensured by imposing penalties for restricted aircraft changes. We observe that there is no significant disadvantage in using Dantzig-Wolfe decomposition compared to Benders decomposition in terms of running time. We also show that the problem becomes much harder to solve if the weight for robustness is increased in the objective function.

In Chapter 6 we enhance the formulation by allowing flexibility for the departure times of some flights. We show that large additional gains in crew pairing cost and robustness can be made when aircraft routing and crew pairing problems are considered in the schedule design phase. A slightly perturbed schedule can lead to significant improvements of aircraft routing and crew pairing solutions. This problem is complex due to the requirement to synchronise departure times on different days of the schedule. We therefore propose two heuristic solution methods that provide good solutions to the problem. Whenever the problem is very hard to solve, the heuristic may fail to improve solution quality. Nevertheless, an un-synchronised solution is useful in indicating possible improvements in crew pairing cost.

We demonstrate in the computational experiments of Chapters 5 and 6 that it is indeed possible to solve the integrated formulations without disturbing the set partitioning structures of the individual problems. We therefore can employ existing and efficient solution methods to solve the individual problems in an integrated model.

As the main focus of this thesis is to solve a real world application, data sets provided by Air New Zealand were used to measure the performance of the solution approaches. All rules imposed by Air New Zealand are satisfied in the solutions we generate. At the time this thesis is finalised, Air New Zealand is using the iterative approach in their production environment. This enables them to operate highly efficient schedules without facing the risk of a major operational breakdown as experienced by Easyjet in 2002. The implementation would not have been possible without considering all rules of a real world application in this research project. If instead a mathematical model is considered that simplifies some of the restrictions, an actual implementation in practice becomes much more difficult. Until optimisation methods are improved, we need to use sensible heuristic decision making methods to some extent in combination with optimisation methods in order to include all restrictions. Despite using data from a New Zealand domestic schedule, the proposed methods are general enough so that we expect similar results for other airlines that operate in a similar environment, i.e. a domestic schedule with routings and pairings containing many flights per day and many short turn times.

Future research includes the integration of other airline scheduling problems, i.e. fleet assignment and crew rostering into an integrated problem. Most importantly, passenger flow should also be considered in an integrated model. Often, an aircraft is delayed because of passengers connecting to the flight are arriving on a delayed flight. Considering passenger flow in the model therefore may greatly improve the robustness of the solutions.

We consider two different robustness measures in this thesis: consecutive minimal turns operated by the aircraft and aircraft changes operated by the crew when turn time is below some restricted time. Simulations show that these measures are good indicators for on-time performance of the operated schedule. It can be useful to consider additional robustness measures such as the number of move-up crews. More sophisticated measures could be considered that take complicated recovery procedures into account.

Finally, the operational counterpart of the problem should be investigated. Since the iterative approach is very fast, it can be used to calculate alterations of routings and pairings once disruptions occur in practice and the planned solutions become invalid. In such a case decisions on how to change the schedule to recover from the disruption must be made quickly. Ideally, the approach could be used to simultaneously re-route aircraft, crew, and passengers in an automated fashion.

References

J. Abara. Applying integer linear programming to the fleet assignment problem. *Interfaces*, 19(4):20–28, 1989.

Y. Ageeva. Approaches to incorporating robustness into airline scheduling. Master's thesis, Massachusetts Institute of Technology, 2000.

S. AhmedBeygi, A. Cohn, and M. Lapp. Decreasing airline delay propagation by re-allocating schedule slack. Technical report, AGIFORS, 2008. http://www.agifors.org/award/submissions2008/Ahmadbeygi_paper.pdf.

R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

R.K. Ahuja, J. Goodstein, A. Mukherjee, J.B. Orlin, and D. Sharma. A very large-scale neighborhood search algorithm for the combined through and fleet assignment model. Technical report, University of Florida, 2001.

R.K. Ahuja, J. Liu, J.B. Orlin, J. Goodstein, and A. Mukherjee. A neighborhood search algorithm for the combined through and fleet assignment model with time windows. *Networks*, 44(2):160–171, 2004.

R. Anbil, E. Gelman, B. Patty, and R. Tanga. Recent advances in crew pairing optimization at american airlines. *Interfaces*, 21:62–74, 1991.

R. Anbil, R. Tanga, and E.L. Johnson. A global approach to crew-pairing optimization. *IBM Systems Journal*, 31(1):71–78, 1992.

R. Anbil, J.J. Forrest, and W.R. Pulleyblank. Column generation and the airline crew pairing problem. In *Proceedings of the International Congress of Mathematicians Berlin*, pages 677–686, 1998. Extra Volume ICM 1998 of Doc. Math. J. DMV.

E. Andersson, E. Housos, N. Kohl, and D. Wedelin. Crew pairing optimization. In G. Yu, editor, *Operations Research in the Airline Industry*, pages 228–258. Kluwer Academic Publishers, 1998.

A.P. Armacost, C. Barnhart, and K.A. Ware. Composite variable formulations for express shipment service network design. *Transportation Science*, 36 (1):1–20, 2002.

J. S. Arora, M. W. Huang, and C. C. Hsieh. Methods for optimization of nonlinear problems with discrete variables: A review. *Structural and Multidisciplinary Optimization*, 8(2):69–85, October 1994.

M. Ball, C. Barnhart, G.L. Nemhauser, and A. Odoni. Air transportation: Irregular operation and control. In C. Barnhart and G. Laporte, editors, *Handbook in OR & MS*, volume 14. Elsevier B.V., 2007.

C. Barnhart and A.M. Cohn. Airline schedule planning: Accomplishments and opportunities. *Manufacturing & and Sevice Operations Management*, 6 (1):3–22, 2004.

C. Barnhart and R.G. Shenoi. An approximate model and solution approach for the long-haul crew pairing problem. *Transportation Science*, 32(3):221–231, 1998.

C. Barnhart, E.L. Johnson, R. Anbil, and L. Hatay. A column generation technique for the long-haul crew assignment problem. In T. Ciriani and R.C. Leachman, editors, *Optimization in Industry: Vol. II*, pages 7–22. Wiley, 1994.

C. Barnhart, L. Hatay, and E.L. Johnson. Deadhead selection for the long-haul crew pairing problem. *Operations Research*, 43(3):491 – 499, 1995. ISSN 0030364X.

C. Barnhart, N.L. Boland, L.W. Clarke, E.L. Johnson, G.L. Nemhauser, and R.G. Shenoi. Flight string models for aircraft fleeting and routing. *Transportation Science*, 32(3):208–220, 1998a.

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

C. Barnhart, F. Lu, and R. Shenoi. Integrated airline schedule planning. In G. Yu, editor, *Operations Research in the Airline Industry*, pages 384–403. Kluwer Academic Publishers, 1998c.

C. Barnhart, T.S. Kniker, and M. Lohatepanont. Itinerary-based airline fleet assignment. *Transportation Science*, 36(2):199–217, 2002a.

C. Barnhart, N. Krishnan, D. Kim, and K. Ware. Network design for express shipment delivery. In *Computational Optimization and Applications*, volume 21, pages 239–262. Kluwer Academic Publishers, 2002b.

C. Barnhart, P. Beloba, and A.R. Odoni. Applications of operations research in the air transport industry. *Transportation Science*, 37(4):368–391, 2003a.

C. Barnhart, A.M. Cohn, E.L. Johnson, D. Klabjan, G.L. Nemhauser, and P.H. Vance. Airline crew scheduling. In R.W. Hall, editor, *Handbook of Transportation Science*, pages 517–560. Kluwer Academic Publishers, 2 edition, 2003b.

C. Barnhart, A. Farahat, and M. Lohatepanont. Airline fleet assignment with enhanced revenue modeling. Technical report, Massachusetts Institute of Technology, 2006.

E.M.L. Beale and J.A. Tomlin. Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables. In J. Lawrence, editor, *Proceedings of the Fifth IFORS Conference*, pages 447–454, Tavistock, London, 1970.

P.P. Belobaba and A. Farkas. Yield management impacts on airline spill estimation. *Transportation Science*, 33(2):217 – 232, 1999.

J.F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252, 1962.

C. Berge. Färbung von graphen, deren sämtliche bzw. deren ungerade kreise starr sind (zusammenfassung). In *Wissenschaftliche Zeitschrift, Mathematisch-Naturwissenschaftliche Reihe*. Martin Luther Universität Halle-Wittenberg, 1961.

M.E. Berge and C.A. Hopperstad. Demand driven dispatch: A method for dynamic aircraft capacity assignment, models and algorithms. *Operations Research*, 41(1):153–168, 1993.

F. Bian, E.K. Burke, S. Jain, G. Kendall, G.M. Koole, J.D. Landa Silva, J. Mulder, M.C.E. Paelinck, C. Reeves, I. Rusdi, and M.O. Suleman. Making airline schedules more robust. Technical report, Free University of Amsterdam, 2003.

R. Borndörfer, A. Löbel, and S. Weider. Integrierte umlauf- und dienstplanung im nahverkehr. Technical Report ZIB-Report 02-10, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 2002.

R. Borndörfer, A. Löbel, and S. Weider. A bundle method for integrated multi-depot vehicle and duty scheduling in public transit. Technical Report ZIB-Report 04-14, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 2004.

E.K. Burke, P. Causmaecker, G. Maere, J. Mulder, M. Paelinck, and G.V. Berghe. A multi-objective approach for robust airline scheduling. Technical report, School of Computer Science, Nottingham, 2007.

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

N. Bélanger, G. Desaulniers, F. Soumis, and J. Desrosiers. Periodic airline fleet assignment with time windows, spacing constraints, and time dependent revenues. *European Journal of Operational Research*, 175(3):1754–1766, December 2006.

K. Büdenbender, T. Grünert, and H.-J. Sebastian. A hybrid tabu search/branch-and-bound algorithm for the direct flight network design problem. *Transportation Science*, 34(4):364–380, 2000.

P. Cappanera and G. Gallo. On the airline crew rostering problem. Technical Report TR-01-08, Department of Computer Science, University of Pisa, 2001.

I.T. Christou, A. Zakarian, J.-M. Liu, and H. Carter. A two-phase genetic algorithm for large-scale bidline-generation problems at delta air lines. *Interfaces*, 29(5):51 – 65, 1999.

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

V. Chvátal. *Linear Programming.* W. H. Freeman, 1983.

L.W. Clarke, C.A. Hane, E.L. Johnson, and G.L. Nemhauser. Maintenance and crew considerations in fleet assignment. *Transportation Science*, 30(3): 249–260, 1996.

L.W. Clarke, E.L. Johnson, G.L. Nemhauser, and Z. Zhu. The aircraft rotation problem. *Annals of Operations Research*, 69:33–46, 1997.

A.M. Cohn and C. Barnhart. Improving crew scheduling by incorporating key maintenance routing decisions. *Operations Research*, 51(3):387–396, 2003.

J.-F. Cordeau, G. Stojković, F. Soumis, and J. Desrosiers. Benders decomposition for simultaneous aircraft routing and crew scheduling. *Transportation Science*, 35(4):375–388, 2001.

J.-F. Cordeau, G. Laporte, M.W.P. Savelsbergh, and D. Vigo. Vehicle routing. In C. Barnhart and G. Laporte, editors, *Handbook in OR & MS*, volume 14, pages 367–428. Elsevier B.V., 2007.

G.B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8(1):101–111, 1960.

M.S. Daskin and N.D. Panayotopoulos. A langrangian relaxation approach to assigning aircraft to routes in hub and spoke networks. *Transportation Science*, 23(2):91–99, 1989.

H. Dawid, J. Konig, and C. Strauss. An enhanced rostering model for airline crews. *Computers & Operations Research*, 28(7):671–688, June 2001.

P.R. Day and D.M. Ryan. Flight attendant rostering for short-haul airline operations. *Operations Research*, 45(5):649–661, 1997.

G. Desaulniers, J. Desrosiers, Y. Dumas, S. Marc, B. Rioux, M.M. Solomon, and F. Soumis. Crew pairing at Air France. *European Journal of Operational Research*, 97:245–259, 1997a.

G. Desaulniers, J. Desrosiers, Y. Dumas, M.M. Solomon, and F. Soumis. Daily aircraft routing and scheduling. *Management Science*, 43(6):841–855, 1997b.

G. Desaulniers, J. Desrosiers, M. Gamache, and F. Soumis. Crew scheduling in air transportation. In T. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 169 – 185. Kluwer Academic Publishers, Boston, 1998.

M. Desrochers and F. Soumis. A column generation approach to the urban transit crew scheduling problem. *Transportation Science*, 23(1):1–13, 1989.

J. Desrosiers, Y. Dumas, M. Solomon, and F. Soumis. Time constrained routing and scheduling. In M. Ball, T. Magnanti, C. Monma, and G. Nemhauser, editors, *Handbook in Operations Research and Management Science, Network Routing*, pages 35–139. Elsevier Science Publishers, 1995.

M. Ehrgott. *Multicriteria Optimization*. Springer, 2005.

M. Ehrgott and D.M. Ryan. Constructing robust crew schedules with bicriteria optimization. *Journal of Multi-Criteria Decision Analysis*, 11:139–150, 2002.

A. Erdmann, A. Nolte, A. Noltemeier, and R. Schrader. Modeling and solving an airline schedule generation problem. *Annals of Operations Research*, 107: 117–142, 2001.

A.T. Ernst, H. Jiang, M. Krishnamoorthy, B. Owens, and D. Sier. An annotated bibliography of personnel scheduling and rostering. *Annals of Operations Research*, 127:21–144, 2004a.

A.T. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier. Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, 153:3–27, 2004b.

D. Espinoza, R. Garcia, M. Goycoolea, G.L. Nemhauser, and M. W. P. Savelsbergh. Per-Seat, On-Demand Air Transportation Part I: Problem Description and an Integer Multicommodity Flow Model. *Transportation Science*, 42(3): 263–278, 2008a.

D. Espinoza, R. Garcia, M. Goycoolea, G.L. Nemhauser, and M. W. P. Savelsbergh. Per-Seat, On-Demand Air Transportation Part II: Parallel Local Search. *Transportation Science*, 42(3):279–291, 2008b.

M.M. Etschmaier and D.F.X. Mathaisel. Airline scheduling: An overview. *Transportation Science*, 19(2):127–138, 1985.

T.A. Feo and J.F. Bard. Flight scheduling and maintenance base planning. *Management Science*, 35(12):1415–1432, 1989.

J.A. Filar, P. Manyem, and K. White. How airlines and airports recover from schedule perturbations: A survey. *Annals of Operations Research*, 108: 315–333, 2001.

M.L. Fisher. The lagrangian relaxation method for solving integer programming problems. *Management Science*, 35:1415–1432, 1981.

M.L. Fisher. An applications oriented guide to lagrangian relaxation. *Interfaces*, 15(2):10–21, 1985.

L.R. Ford and D.R. Fulkerson. A suggested computation for maximal multicommodity network flows. *Management Science*, 5:97–101, 1958.

R. Freling, D. Huisman, and A.P.M. Wagelmans. Models and algorithms for integration of vehicle and crew scheduling. *Journal of Scheduling*, 6:63–85, 2003.

B. Fuhr. Robust flight scheduling - an analytic approach to performance evaluation and optimization. Technical report, AGIFORS, 2007. http://www.agifors.org/award/submissions2007/Fuhr_paper.pdf.

R. Galia and C. Hjörring. Modeling of complex costs and rules in a crew pairing column generator. Technical Report CRTR-0304, Carmen Systems, 2003.

M. Gamache and F. Soumis. A method for optimally solving the rostering problem. In G. Yu, editor, *Operations Research in the Airline Industry*, pages 124–157. Kluwer Academic Publishers, 1998.

M. Gamache, F. Soumis, D. Villeneuve, J. Desrosiers, and É. Gélinas. The preferential bidding system at Air Canada. *Transportation Science*, 32(3): 246–255, 1998.

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

A.M. Geoffrion. Lagrangean relaxation for integer programming. *Mathematical Programming Study 2*, 2:82–114, 1974.

I. Gershkoff. Optimizing flight crew schedules. *Interfaces*, 19:29–43, 1989.

J.L. Goffin and J.P. Vial. Convex nondifferentiable optimization: A survey focused on the analytic center cutting plane method. *Optimization Methods and Software*, 17(5):805–867, 2002.

B. Gopalakrishnan and E.L. Johnson. Airline crew scheduling: State-of-the-art. *Annals of Operations Research*, 140:305–337, 2005.

R. Gopalan and K.T. Talluri. The aircraft maintenance routing problem. *Operations Research*, 46(2):260–271, 1998a.

R. Gopalan and K.T. Talluri. Mathematical models in airline schedule planning: A survey. *Annals of Operations Research*, 76:155–185, 1998b.

G.W. Graves, R.D. McBride, I. Gershkoff, D. Anderson, and D. Mahidhara. Flight crew scheduling. *Management Science*, 39(6):p736 – 745, 1993.

M. Grönkvist. Accelerating column generation for aircraft scheduling using constraint propagation. *Computers & Operations Research*, 33:2918–2934, 2006.

Z. Gu, E.L. Johnson, G.L. Nemhauser, and Y. Wang. Some properties of the fleet assignment problem. *Operations Research Letters*, 15(2):59–71, 1994.

K. Haase, G. Desaulniers, and J. Desrosiers. Simultaneous vehicle and crew scheduling in urban mass transit systems. *Transportation Science*, 35(3): 286–303, 2001.

C.A. Hane, C. Barnhart, E.L. Johnson, R.E. Marsten, G.L. Nemhauser, and G. Sigismondi. The fleet assignment problem: Solving a large-scale integer program. *Mathematical Programming*, 70:211–232, 1995.

J.B. Hiriart-Urruty and C. Lemaréchal. Convex analysis and minimization algorithms, part 2: Advanced theory and bundle methods. *Grundlehren der mathematischen Wissenschaften*, 306, 1993.

K.L. Hoffman and M. Padberg. Solving airline crew scheduling problems by branch-and-cut. *Management Science*, 39(6):657 – 682, 1993.

A.J. Hoffmann and J.B. Kruskal. Integral boundary points of convex polyhedral. In H.W. Kuhn and A.W. Tucker, editors, *Linear inequalities and related systems*, pages 223–246. Princeton University Press, Princeton, NJ, 1956.

D. Huisman, R. Freling, and A.P.M. Wagelmans. Multiple-depot integrated vehicle and crew scheduling. *Transportation Science*, 39(4):p491 – 502, 2005.

ILOG. Concert Technology 2.3 (user manual), 2006a.

ILOG. CPLEX 10.1 (user manual), 2006b.

I. Ioachim, J. Desrosiers, F. Soumis, and N. Bélanger. Fleet assignment and routing with schedule synchronization constraints. *European Journal of Operational Research*, 119:75–90, 1999.

A.I. Jarrah and J.C. Strehler. An optimization model for assigning through flights. *IIE Transactions*, 32(3):237–244, 2000.

L. Kang. *Degradable airline scheduling*. PhD thesis, Operations Research Center, Massachusetts Institute of Tech- nology, Cambridge, MA, 2003.

L.G. Khachian. A polynomial algorithm for linear programming. *Soviet Mathematics Doklady*, 20:191–194, 1979.

D. Klabjan. Large-scale models in the airline industry. In G. Desaulniers, J. Desrosiers, and M.M. Solomon, editors, *Column Generation*, pages 163–196. Kluwer Scientific Publishers, 2005.

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

D. Klabjan, E.L. Johnson, G.L. Nemhauser, E. Gelman, and S. Ramaswamy. Airline crew scheduling with regularity. *Transportation Science*, 35(4):359 – 374, 2001b.

D. Klabjan, E.L. Johnson, G.L. Nemhauser, E. Gelman, and S. Ramaswamy. Airline crew scheduling with time windows and plane-count constraints. *Transportation Science*, 36(3):337–348, 2002.

G. Kliewer. Integrating market modelling and fleet assignment. Technical report, University of Paderborn, 1996.

N. Kohl and S.E. Karisch. Airline crew rostering: Problem types, modeling and optimization. *Annals of Operations Research*, 127:223–257, 2004.

L. Kwok and L. Wu. Development of an expert system in cabin crew pattern generation. *International Journal of Expert Systems*, 9:445–464, 1996.

S. Lan, J.-P. Clarke, and C. Barnhart. Planning for robust airline operations: Optimizing aircraft routes and flight departure times to minimize passenger disruptions. *Transportation Science*, 40(1):15–28, 2006.

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*, 35:45–58, 1988.

L. Lettovský, E.L. Johnson, and G.L. Nemhauser. Airline crew recovery. *Transportation Science*, 34(4):337–348, 2000.

D. Li and X. Sun. *Nonlinear Integer Programming*, volume 84 of *International Series in Operations Research & Management Science*. Springer, 2006.

O. Listes and R. Dekker. A scenario aggregation based approach for determining a robust airline fleet composition. Technical Report EI 2002-17, Rotterdam School of Economics, 2002.

M. Lohatepanont and C. Barnhart. Airline schedule planning: Integrated models and algorithms for schedule design and fleet assignment. *Transportation Science*, 38(1):19–32, 2004.

M.E. Lübbecke and J. Desrosiers. Selected topics in column generation. Technical report, Technische Universität Berlin, 2004.

A. Makri and D. Klabjan. A new pricing scheme for airline crew scheduling. *INFORMS Journal on Computing*, 16(1):56–67, 2004.

R. Martin. *Large Scale Linear and Integer Optimization: a Unified Approach.* Kluwer Academic Publishers, 1999.

A. Mercier. *Integrated aircraft routing and crew pairing.* PhD thesis, Université de Montreal, 2006.

A. Mercier and F. Soumis. An integrated aircraft routing, crew scheduling and flight retiming model. *Computers & Operations Research*, 34:2251–2265, 2007.

A. Mercier, J.-F. Cordeau, and F. Soumis. A computational study of benders decomposition for the integrated aircraft routing and crew scheduling problem. *Computers & Operations Research*, 32:1451–1476, 2005.

M. Minoux. *Mathematical Programming: Theory and Algorithms.* Wiley-Interscience, 1986.

G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimaization.* John Wiley & Sons, 1988.

M.W. Padberg. Perfect zero-one matrices. *Mathematical Programming*, 6: 180–196, 1974.

N. Papadakos. Integrated airline scheduling. *Computers & Operations Research*, 2007. doi:10.1016/j.cor.2007.08.002.

V.L. Pilla. *Robust Airline Fleet Assignment.* PhD thesis, The University of Texas in Arlington, 2006.

T.K. Ralphs and M.V. Galati. Decomposition and dynamic cut generation in integer linear programming. *Mathematical Programming, Series A*, 106: 261–285, 2006.

B. Rexing, C. Barnhart, T.S. Kniker, A. Jarrah, and N. Krishnamurthy. Airline fleet assignment with time windows. *Transportation Science*, 34(1): 1–20, 2000.

J.M. Rosenberger, A.J. Schaefer, D. Goldsman, E.L. Johnson, A.J. Kleywegt, and G.L. Nemhauser. A stochastic model of airline operations. *Transportation Science*, 36(4):357–377, 2002.

J.M. Rosenberger, E.L. Johnson, and G.L. Nemhauser. Rerouting aircraft for airline recovery. *Transportation Science*, 37(4):408 – 421, 2003.

J.M. Rosenberger, E.L. Johnson, and G.L. Nemhauser. A robust fleet assignment model with hub isolation and short cycles. *Transportation Science*, 38 (3):357–368, 2004.

R.A. Rushmeier and S.A. Kontogiorgis. Advances in the optimization of airline fleet assignment. *Transportation Science*, 31(2):159–169, 1997.

D.M. Ryan. Zip - a zero-one integer programming package for scheduling. Technical report, A.E.R.E., Harwell, Oxfordshire, 1980.

D.M. Ryan. The solution of massive generalized set partitioning problems in air crew rostering. *Journal of the Operational Research Society*, 43:459–467, 1992.

D.M. Ryan and B.A. Foster. An integer programming approach to scheduling. In A. Wren, editor, *Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling*, pages 269–280. North-Holland, 1981.

R. Sandhu and D. Klabjan. Integrated airline fleeting and crew-pairing decisions. *Operations Research*, 55(3):439–456, 2007.

A. Sarac, R. Batta, and C.M. Rump. A branch-and-price approach for operational aircraft maintenance routing. *European Journal of Operational Research*, 175:1850–1869, 2006.

A.J. Schaefer, E.L. Johnson, A.J. Kleywegt, and G.L. Nemhauser. Airline crew scheduling under uncertainty. *Transportation Science*, 39(3):340–348, 2005.

A. Schrijver. *Theory of Linear and Integer Programming.* John Wiley & Sons, 1986.

M. Sellmann, K. Zervoudakis, P. Stamatopoulos, and T. Fahle. Crew assignment via constraint programming: Integrating column generation and heuristic tree search. *Annals of Operations Research*, 115:207–225, 2002.

S. Shebalov and D. Klabjan. Robust airline crew pairing: Move-up crews. *Transportation Science*, 40(3):300–312, 2006.

H.D. Sherali, E.K. Bish, and X. Zhu. Airline fleet assignment concepts, models, and algorithms. *European Journal of Operational Research*, 172:1–30, 2006.

B.C. Smith and E.L. Johnson. Robust airline fleet assignment: Imposing station purity using station decomposition. *Transportation Science*, 40(4): 497–516, 2006.

C. Sriram and A. Haghani. An optimization model for aircraft maintenance scheduling and re-assignment. *Transportation Research Part A*, 37:29–48, 2003.

G. Stojković, F. Soumis, J. Desrosiers, and M.M. Solomon. An optimization model for a real-time flight scheduling problem. *Transportation Research Part A: Policy and Practice*, 36(9):779–788, November 2002.

M. Stojković, F. Soumis, and J. Desrosiers. The operational airline crew scheduling problem. *Transportation Science*, 32(3):232–245, 1998.

R. Subramanian, R.P. Scheff Jr., J.D. Quillinan, D.S. Wiper, and R.E. Marsten. Coldstart: Fleet assignment at delta air lines. *Interfaces*, 24(1): p104 – 120, 1994.

K.T. Talluri. Swapping applications in a daily airline fleet assignment. *Transportation Science*, 30(3):237–248, 1996.

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

The MathWorks Inc. Matlab version 7.3.0, 2003.

M.P. Thiel. *Team-Oriented Airline Crew Scheduling and Rostering: Problem Description, Solution Approaches and Decision Support*. PhD thesis, University of Paderborn, 2005.

C. A. van Eijl. A polyhedral approach to the delivery man problem. Technical report, Eindhoven University of Technology, 1995. Memorandum COSOR 95-19.

T.J. van Roy. A cross decomposition algorithm for capacitated facility location. *Operations Research*, 34(1):145–163, 1986.

G. van Ryzin and K.T. Talluri. Revenue management. In R. Hall, editor, *Handbook of Transportation Science*, pages 599–661. Kluwer Academic Publishers, 2002.

P.H. Vance, A. Atamtürk, C. Barnhart, E. Gelman, E.L. Johnson, A. Krishna, D. Mahidhara, G.L. Nemhauser, and R. Rebello. A heuristic branch-and-price approach for the airline crew pairing problem. Technical report, Georgia Institute of Technology, 1997a.

P.H. Vance, C. Barnhart, E.L. Johnson, and G.L. Nemhauser. Airline crew scheduling: A new formulation and decomposition algorithm. *Operations Research*, 45(2):188–200, 1997b.

C.A. Wallace. *Construction of optimal tours of duty for long-haul flight attendants*. PhD thesis, University of Auckland, 2001.

D. Wedelin. An algorithm for large scale 0-1 integer programming with application to airline crew scheduling. *Annals of Operations Research*, 57:283–301, 1995.

L. Wolsey. *Integer Programming*. Wiley-Interscience publication, 1998.

C.-L. Wu. Improving airline network robustness and operational reliability by sequential optimisation algorithms. *Netw Spat Econ*, 6:235–251, 2006.

S. Yan and C.-H. Tseng. A passenger demand model for airline flight scheduling and fleet routing. *Computers and Operations Research*, 29:1559–1581, 2002.

J.W. Yen and J.R. Birge. A stochastic programming approach to the airline crew scheduling problem. *Transportation Science*, 40(1):3–14, 2006.

G. Yu, M. Argüello, G. Song, S.M. McCowan, and A. White. A new era for crew recovery at continental airlines. *Interfaces*, 33(1):5 – 22, 2003.