INTEGRATED AIRCRAFT ROUTING AND CREW PAIRING PROBLEM BY BENDERS DECOMPOSITION

LIANG ZHE

(B.Eng.(Hons.) NUS)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF INDUSTRIAL & SYSTEMS ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2003

Acknowledgments

I would like to thank Prof. Huei-Chuen Huang, my supervisor, for her many suggestions and constant support during this research. Without her, I will not understand linear and integer programming as today. Also, I learn the attitude of doing research from her, which will benefit me even more.

I am also thankful to Dr. Li Rongheng and Dr. Alexander David Morton for their help when I was struggling in understanding Benders decomposition and paper writing.

I had the pleasure of meeting Li Dong, Ivy Mok and Leong Chun How. They are wonderful people. When we together study linear and integer programming and CPLEX, we shared our knowledge and help each other. Mr Leong Chun How also shared with me his knowledge on Airline planning and operations and provided many useful references when I first join the AirCargo team. Also, I feel so lucky that I can study in ISE together with Li Dong, who becomes one of my best friends.

Of course, I am grateful to my parents for their patience and *love.* Without them this work would never have come into existence.

Finally, I wish to thank the following: Li Rujing (for her cookies); Lin Wei, Zhang Jun (for playing badminton together); Huang Peng, Sun Hainan, Gao Fei, Liu Bin and Xu Zhiyong (for taking lunch together).

Liang Zhe
November , 2003

Contents

List of Tables

List of Figures

Summary

The traditional airline planning is usually divided in several stages and solved sequentially, due to its size and complexity. The early stage results are inputs to the subsequent stage problems. Therefore, this sequential method may result in sub-optimality in planning. However, a fully integrated model is not tractable because of its enormous size. Nonetheless, benefits can be gained by partially integrating elements of the planning process. This paper uses the Benders decomposition to solve the integrated aircraft routing and crew pairing problem. Reversing the conventional approach, the crew pairing is formulated as the Benders master problem while a linear program on the selection of an aircraft maintenance routing is considered as the subproblem. We exploit the structure of the subproblem and identify two types of feasibility cuts. Test cases are generated to compare these two types of cuts. One of them is found to be stronger while the other is found to be computationally more efficient.

Chapter 1

Introduction

1.1 Traditional Airline Schedule Planning

The airline planning is usually divided in several stages and solved sequentially, due to its size and complexity. Airline schedule planning consists of four major steps, which are flight schedule design, fleet assignment, aircraft maintenance routing and crew scheduling.

The *flight schedule* (Phillips et al., 1991) is usually determined based on a few factors like traffic forecasts, airline network analysis and profitability analysis. The schedule is often built by the airline marketing department and once it is publised it will last for a number of months.

Given a flight schedule and a set of aircrafts, the *fleet assignment problem* (Hane et al., 1995) is to decide which type of aircrafts to fly the flight segments. A fleet type prescribed by the manufacture is a particular class of aircrafts which has a given seating capacity and fuel consumption. An airline usually has a variety of fleets. Considering factors such as passenger demands (both point-to-point and continuing services), revenues, operation costs etc, the fleet assignment faced by the airline is to assign a fleet to each flight of the schedule so as to maximize the total profit.

Given a fleet assignment solution, a *maintenance routing problem* (Clarke et al., 1997) is then solved to determine the individual aircraft rotation, so that enough maintenance opportunities are provided for each aircraft. There are different types of maintenance checks. The checks differ by the amount of work to be done. For example, Federal Aviation Administration (FAA) requires A, B, C and D checks (FAA, 2002). Type A checks inspect all the major systems and are performed frequently (every 65 flight hours). B checks entail a thorough visual inspection plus lubrication of all moving parts, and are performed every 300 to 600 flight hours. C and D checks require taking the aircraft out of service for up to a month at a time and are done about once every one to four years. Also, different airlines operate slightly different maintenance regulations. The maintenance routing problem is to schedule the most frequent maintenances, e.g. A type maintenance, whereas the less frequent, e.g. B, C and D type of maintenances, can be incorporated into the fleet assignment problem.

The planning process of crew scheduling consists of two steps: crew pairing and crew rostering. The *crew pairing problem* (Barnhart et al., 1999) is to determine the best set of *crew pairings* to cover the flights. A crew pairing is a crew trip spanning one or more work days separated by periods of rest. Each cockpit crew is qualified to fly a set of closely related fleet types, known as a *fleet family.* Therefore, a crew pairing problem is solved for those flights assigned to the corresponding fleet family.

The next step, called a *rostering problem* (Gamache and Soumis, 1998), is to construct personalized monthly schedules (rosters) for crew members by assigning them pairings and rest periods.

1.2 Integrated Planning

As we can see, the solutions of early stage problems are inputs to the subsequent stages. Therefore, solving these planning problems sequentially can lead to sub-optimality. However a fully integrated approach to the airline planning process is not tractable, due to its enormous size and complexity. Nonetheless, benefits can be gained by partially integrating elements of the planning process. For example, a fleet assignment problem can be solved incorporating with aircraft routing, crew scheduling or passenger flow considerations.

Particularly in this thesis, we consider the aircraft routing problem together with the crew pairing problem. One restriction on a valid pairing is that two sequential flights cannot be assigned to the same crew unless the time between the flights is sufficient (known as *minimum sit time*). This minimum sit time can be shortened if the crew follows the plane turn, which is called a *short connect*. Even though the difference between the minimum turn time and the minimum sit time is small, using more short connects in planning can significantly improve the robustness of the crew scheduling. This is because during the aircraft disruption, it is possible to absorb the delay in the crew assignment if initially the crew is assigned to follow the aircraft turn.

1.3 Research Contribution

In this thesis we solve the integrated planning problem by adopting the extended crew pairing model and approach it by the Benders decomposition. The crew pairing problem is considered as the master problem while the selection of a maintenance routing out of the collection of all maintenance routing solutions, which can be treated as a linear program, is solved as a subproblem. We provide insights into the subproblem and identify two types of feasibility cuts. One of them is a cut generated from a minimally infeasible short connect set, proposed by Cohn and Barnhart (2003). The other is a cut generated by a maximal short connect set. The first type of cuts is found to be stronger. However, in solving the integrated problem it is found to be less efficient computationally.

1.4 Organization of This Thesis

The remainder of the thesis is organized as follows. In Chapter 2, we give a literature review on the related airline planning problems. Also related models are presented in this chapter, e.g. the crew pairing problem model, the string model for the aircraft maintenance routing problem and the extended crew pairing model. In Chapter 3, we give the framework for the Benders decomposition and identify two types of feasibility cuts. Computational and implemental issues are addressed in Chapter 4. Chapter 5 concludes the thesis with a comparison study on the effectiveness and efficiency of solving the integrated problem by the two types of feasibility cuts.

Chapter 2

Literature Review

In this chapter, we first list the previous work done on aircraft maintenance routing and crew pairing problems. Then we briefly describe three previous research works on the integrated planning of aircraft maintenance routing and crew pairing problems.

2.1 Aircraft Maintenance Routing Problem

In Figures 2.1, a simple time space network is shown. The schedule contains 2 cities and 12 flights everyday.

In this network, the two horizontal lines represent the time at stations. The vertices of the network are the flight events on the stations (arrivals and departures of aircraft). The edge of the network comprised of the flight arcs and the ground arcs. A flight arc represents a flight between two stations. The ground arcs connect two subsequent flight events on an airport. We need to make time line a cycle so that the schedule can be circulated (represented by the dash line).

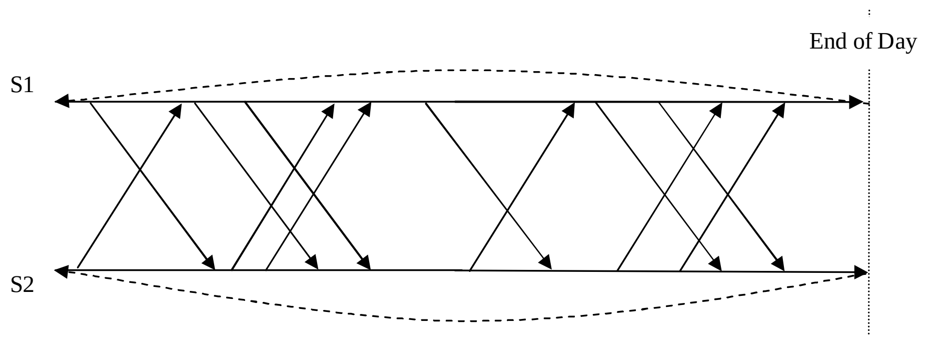

Figure 2.1: Time Space Network

After a fleet assignment problem is solved, the time space network for each fleet type is an *Eulerian* digraph, i.e. each node has its indegree equal to its outdegree and it is connected. Clarke et al. (1997) proposed to solve the aircraft maintenance routing problem by forming an *Euler tour* in this network, in which all the *service violation paths* are eliminated. An Euler tour is a cycle that includes all the arcs exactly once. A service violation path is a path with length in time longer than the specified service period. By excluding the service violation path in the Euler tour, the maintenance constraints are satisfied. The objective is to maximize the benefit derived from making specific connections, which referred as *through value*. The problem is solved using the Lagrangian relaxation and subgradient optimization.

Boland et al. (2000) and Mak and Boland (2000) proposed to solve the aircraft maintenance routing problem by modelling it as an *Asymmetric Travelling Salesman Problem with Replenishment arcs* (RATSP). A network is built for solving the aircraft maintenance routing problem by RATSP, where nodes represent flights, arcs represent pairs of flights that may be connected in sequence and replenishment arcs represent maintenance connections. In Boland et al. (2000), authors solved the RATSP using a branch-and-bound algorithm. Mak and Boland (2000) solved the problem using three different types of heuristics. One is a simulated annealing algorithm and the other two are based on the Lagrangian relaxation of different constraints (subtour constraints and weight limit violation constraints). The result shows that the algorithm based on simulated annealing performs well overall.

In Barnhart et al. (1998), authors proposed a string model to solve the aircraft maintenance routing problem. A *string* is a sequence of connected flights that begins and ends at maintenance stations, satisfies flow balance and is maintenance feasible. The string may begin and end at different maintenance stations. The model formulates the problem as a set partitioning problem with side constraints. Formally, the model is given as below.

$$(MR) \quad \min \sum_{r \in R} c_r z_r \tag{2.1}$$

subject to

$$\sum_{r \in R} \alpha_{fr} z_r = 1 \quad \forall\, f \in F \tag{2.2}$$

$$\sum_{\text{r ends at n}} z_r + g_n^- - \sum_{\text{r starts at n}} z_r - g_n^+ = 0 \quad \forall\, n \in N \tag{2.3}$$

$$\sum_{r \in R^M} \gamma_r z_r + \sum_{n \in Z^M} g_n^+ \leq K \tag{2.4}$$

$$d_r \in \{0, 1\} \quad \forall\, r \in R \tag{2.5}$$

$$g_n^+, g_n^- \geq 0 \quad \forall\, n \in N \tag{2.6}$$

where

F**:** the set of flights

R**:** the set of feasible route strings

c_r**:** the cost of route string r

α_{fr}**:** assuming value 1 if route string r covers flight f and 0 otherwise

z_r**:** decision variables, assuming value 1 if route string r is included in the solution and 0 otherwise

N**:** the set of nodes representing points in space and time at which route strings begin or end

g_n^- and g_n^+**:** the ground arc variables capturing the number of aircrafts on the ground at station s immediately prior to and immediately following time t, given a node n that represents time t at station s

R^M**:** the set of route strings spanning time M, which is an arbitrary time known as the *countline*

γ_r**:** the number of times string r crosses the countline M

Z^M**:** the set of nodes with corresponding ground arcs g_n^+ spanning the countline

K**:** the total number of available aircrafts

The objective function (2.1) minimizes the cost of the chosen route strings. The constraints set (2.2) are covering constraints, which state that each flight must be included in exactly one chosen route string. The constraints set (2.3) are balance constraints. Constraint (2.4) makes sure that the total number of aircrafts in use at time M does not exceed the number of aircrafts in the fleet. Consequently, the number of aircrafts does not exceed the fleet size at any time because of constraints (2.3). Constraints (2.5) ensure that the string decision variable are binary. Thus the integrality of the ground arc variables can be relaxed as denoted in (2.6).

This problem is solved using a branch-and-price algorithm. It is noticed that the time between any two sequential flights in the string can not be less than the *minimum turn time*. This time is used for aircrafts to change gate, clean up and so on.

2.2 Crew Pairing Problem

The airline crew pairing problem is well studied and is often modelled as a set partitioning problem (Anbil et al., 1992, Barnhart et al., 1999). The model is formulated as follows

$$(CP) \quad \min \sum_{p \in P} c_p y_p \tag{2.7}$$

subject to

$$\sum_{p \in P} \delta_{fp} y_p = 1 \quad \forall\ f \in F \tag{2.8}$$

$$y_p \in \{0,1\} \quad \forall\ p \in P \tag{2.9}$$

where

P: the set of feasible pairings

c_p: the cost of pairing p

δ_{fp}: assuming value 1 if flight f is included in pairing p and 0 otherwise

y_p: assuming value 1 if pairing p is included in the solution and 0 otherwise

The objective function (2.7) minimizes the cost of the chosen set of pairing. The constraints (2.8) are the covering constraints and (2.9) are the integrality requirement.

The payment structure and the working rules for the airline cockpit crew is complex. In US airlines, crews are mainly paid for the time they spend in flying; whereas in some other airlines, the crew pay structure may not be exactly the same. In this thesis, we follow closely with the US pay structure as it is readily available from the journal articles.

2.2.1 Duty

A pairing consists of several duties. A duty is a sequence of flights that can be flown by a single crew over the course of a work day. There are several constraints to be satisfied when a duty is constructed. The most obvious constraint is that flights must be sequential in space and time. The time between any two sequential flights can not be less than a constant time duration, which is known as *minimum sit time*. Also, there is a *maximum duty time* (also know as *maximum duty elapsed time*) limitation for a duty. The total elapsed time of a duty cannot be more than the maximum duty time. Another strict regulation governs the total number of flying hours (known as *block time*) that a crew can fly in a single duty period.

The cost of a duty is usually the maximum of three quantities. The first quantity is the block time. The second is a fraction of the duty elapsed time. The third one is a minimum guaranteed cost. Formally, the cost of a duty d, denoted as c_d, can be expressed as follow:

$$c_d = \max\{\tau_d \times elapse, fly, mg\} \tag{2.10}$$

where $\tau_d \times elapse$ is a fraction of the elapsed time, fly is the block time of the duty, and mg is the minimum guarantee cost. It should be noticed that the *start time* of a duty is usually one hour before the departure time of the first flight (this one hour duration is known as *briefing time*) and the *end time* of a duty is usually 15 minutes after the arrival time of the last flight (15 minutes is known as *debriefing time*). The duty elapsed time is the time duration between the duty start time and the duty end time.

2.2.2 Pairing

A pairing consists of a sequence of duties and it starts and ends at the same crewbase. In general, crews spend one to several days away from their homebase. A few constraints have to be considered when a pairing is built. First, a pairing's first duty must begin from one of the crewbases, hence it must end at the same crewbase. Also each duty must begin at the same airport where the previous duty ends. Similar with the duty, a pairing cannot last longer than a *maximum elapsed time*, also known as *time-away-from-base* (TAFB). Another very complicated rule is the *8-in-24* rule, which is imposed by the FAA (FAA, 2002). The basic idea of this rule is if a pairing contains more than 8 flying hours in any 24 hours period, then extra rest is required.

Similar with the duty cost, the cost of a pairing is the maximum of three values. The first value is a fraction of the total elapsed time of a pairing. The second value is the sum of the costs of the duties and the third one is called *minimum guaranteed cost* per pairing. The pairing cost is formally expressed as below:

$$c_p = \max\{\tau_p \times TAFB, \sum_{d \in p} c_d, ndp * mg\}$$

where ndp is the minimal number of duties per pairing, τ_p is the fraction constant, and $TAFB$ is the elapsed time of a pairing.

2.2.3 Selected Issues

In this section we discuss a few implemental issues.

Pairing Enumeration

Since the cost structure and the constraints are very complicated, pairings are often enumerated before a model is solved. Two different networks are commonly used in enumerating the crew pairings.

The first network is called *duty network* (Vance et al., 1997). Duties are first generated based on the daily schedule. Then a duty network is constructed. Within this network, nodes represent duties; arcs represent the possible connections between duties. Each duty is repeated as many times as the number of maximal calender days allowed in a pairing. Two duties can be connected only when the arrival airport of the first duty is the same as the departure airport of the second duty and the time in between is a legal overnight rest. It should be noted that no repeated flight can appear in a pairing in a daily problem. This can be partially done by ensuring no connects between any two duties if they contain a common flight. Also, a source and a sink node are included in the network. The source node is connected to duties that originate at a crewbase. The duties that end at a crewbase is connected with the sink node.

The second network is known as *flight network*. Each flight is duplicated as many as the maximal days allowed in a pairing. In this network, nodes represent the flights and arcs represent the possible connections between flights. Similar to the duty network, a source node and a sink node are included in the flight network and connected with flights which start and end at the same crewbase. Different from the duty network, no duty rules and over night rests information can be inherited in the flight network. However, the flight network is good for a large schedule when constructing pairings. This is because the number of duties increases exponentially with the number of flights in the schedule. Consequently, the duty network is much more complicated than the flight network for a large schedule.

The pairing is constructed by a depth-first search on the duty or flight network. The search is to extend partial pairings or backtrack. However, when using a flight network, more checks are needed to ensure that both duty rules and pairing rules are satisfied.

Branching Rules

The number of pairings grows exponentially with the number of flights. Therefore, for some large flight schedule, we cannot list out all the pairings and branch-and-price is often employed to get a good crew pairing solution. In the cases even though we can list down all pairings, it is still critical to engage good branching rule in the branch-and-bound solving procedure. When branching in solving the crew pairing problem, a useful technique referred as branching on *follow-ons* is often employed. This branching rule is motivated by a general rule for set partitioning problems developed by Ryan and Foster (1981). It is based on the observation that given a fractional solution to the LP relaxation of a set partitioning problem, there must exist two columns whose associated variables are fractional such that they both contain coefficients of one in a common row r and there exists another row s where one column has a coefficient of one and the other has a coefficient of zero. This fact leads to a general branching rule where a pair of rows r and s are required to be covered by the same column on one branch and by different columns on the other. Specifically, when solving the crew pairing problem, we can branch by requiring that two flights appear consecutively in pairings at one branch; in the other branch, these two flights cannot appear consecutively in any pairings. Here, we refer flight pair r, s as a follow-on.

2.3 Integrated Planning for Maintenance Routing and Crew Pairing Problems

Three papers were found to solve the integrated planning for the maintenance routing and crew pairing problems, which address the effect of short connects. They are discussed below in more details.

2.3.1 Klabjan et al. (2002)

Klabjan et al. (2002) are the first to address the impact of short connects on the crew pairing problem. They demonstrate that by considering the short connects in solving the crew pairing problem, the cost is significantly reduced from the traditional sequential model.

For example, in figures 2.2, there are 4 flights events in a station. If a minimal sit time is 45 minutes and not short connects are allowed, we can only have a crew connect $A \rightarrow D$. Therefore, we need another crew ready before 8:37 for flight C. However, if we allow short connects in the crew schedule, we could have crew connects $A \rightarrow C$ and $B \rightarrow D$. This could save the cost potentially.

In Klabjan et al. (2002), the planning problems are solved in reverse order. They first solve the crew pairing problem assuming all the short connects are valid. A set of constraints are added to the original crew pairing model to ensure that the number of the aircrafts used at any short connection period is not more than the fleet size. Then they solve a maintenance routing problem to incorporate the short connects selected in the crew pairing solution. This approach can lead to maintenance infeasibility. However, in practice they find feasible solutions for some hub-and-spoke flight networks using this approach. As long

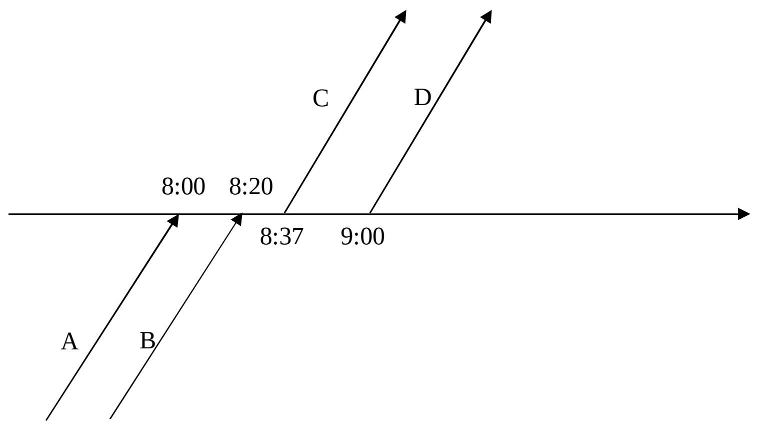

Figure 2.2: Short Connect Example

as the maintenance routing problem is feasible, the solution for the crew pairing problem is optimal for the integrated problem. This method requires no more computational effort than the traditional sequential method.

Here, the aircraft maintenance routing problem is considered to be a feasibility problem, since no costs are involved. This is reasonable because the running cost of aircraft is more or less determined at this planning stage, which takes place after the schedule design and fleet assignment stages.

2.3.2 Cordeau et al. (2001)

Cordeau et al. (2001) present a *basic integrated model* for the maintenance routing and crew pairing problems, which guarantees the maintenance feasibility. The maintenance routing cost is explicitly considered in this model. The authors assume a dated planning horizon in which the set of flights may vary from day to day. Here, we modify their model slightly to cater for daily problems. The basic integrated model is formulated as below.

$$(BIM) \quad \min \sum_{p \in P} c_p y_p + \sum_{r \in R} c_r z_r \tag{2.11}$$

subject to

$$\sum_{p \in P} \delta_{fp} y_p = 1 \quad \forall\ f \in F \tag{2.12}$$

$$\sum_{r \in R} \alpha_{fr} z_r = 1 \quad \forall\ f \in F \tag{2.13}$$

$$\sum_{\text{r ends at n}} z_r + g_n^- - \sum_{\text{r starts at n}} z_r - g_n^+ = 0 \quad \forall\ n \in N \tag{2.14}$$

$$\sum_{r \in R^M} \gamma_r z_r + \sum_{n \in Z^M} g_n^+ \leq K \tag{2.15}$$

$$\sum_{r \in R} \vartheta_{tr} z_r - \sum_{p \in P} \eta_{tp} y_p \geq 0 \quad \forall\ t \in T \tag{2.16}$$

$$d_r \in \{0, 1\} \quad \forall\ r \in R \tag{2.17}$$

$$g_n^+, g_n^- \geq 0 \quad \forall\ n \in N \tag{2.18}$$

$$y_p \in \{0, 1\} \quad \forall\ p \in P \tag{2.19}$$

where besides the notations defined in section 2.1 and 2.2,

T**:** the set of all possible short connects

ϑ_{tr}**:** assuming value 1 if string r contains short connect t and 0 otherwise

η_{tp}**:** assuming value 1 if pairing p contains short connect t and 0 otherwise

Objective function (2.11) minimizes the cost of chosen pairings and strings. Constraints sets (2.12) and (2.19) are the same as in CP model. Constraints sets (2.13)-(2.15), (2.17) and (2.18) are the same as in MR model. The two models are linked by constraints set (2.16), which ensure that a short connect is selected in a crew pairing only when it appears in the maintenance routing solution.

Here, crew pairings are constructed with all potential short connects allowed. This model results in a large-scale integer program. The model is solved using a Benders decomposition approach coupled with a heuristic branching strategy, in which the maintenance routing is considered as the master problem while the crew pairing as the subproblem.

2.3.3 Cohn and Barnhart (2003)

Cohn and Barnhart (2003) proposed an *extended crew pairing model* integrating crew pairing and maintenance routing decisions. As in Cordeau et al. (2001), to ensure the maintenance feasibility, the two decisions are linked by short connect constraints. The same constraint appears in Klabjan et al. (2002), but their model does not consider the maintenance routing cost explicitly. The decision on the maintenance routing is captured as a problem of choosing one out of all the feasible maintenance routing solutions. Formally, the model is formulated below.

$$(ECP) \quad \min \sum_{p \in P} c_p y_p \tag{2.20}$$

subject to

$$\sum_{p \in P} \delta_{fp} y_p = 1 \quad \forall\ f \in F \tag{2.21}$$

$$\sum_{s \in S} \beta_{ts} x_s - \sum_{p \in P} \eta_{tp} y_p \geq 0 \quad \forall\ t \in T \tag{2.22}$$

$$\sum_{s \in S} x_s = 1 \tag{2.23}$$

$$x_s \in \{0, 1\} \quad \forall\ s \in S \tag{2.24}$$

$$y_p \in \{0, 1\} \quad \forall\ p \in P \tag{2.25}$$

where the additional notations are

S**:** the set of feasible maintenance routing solutions

β_{ts}**:** assuming value 1 if short connect t is included in maintenance routing solution s and 0 otherwise

x_s**:** assuming value 1, if maintenance routing solution s is chosen and 0 otherwise

Without constraint sets (2.22)-(2.24), this problem is simply the crew pairing model. Constraints (2.23) and (2.24) ensure that exactly one maintenance routing solution is chosen. Constraints (2.22) ensure that a short connect is used in a pairing only when it is included in the selected maintenance routing solution. It is observed that the binary constraints (2.24) can be relaxed and replaced by a set of non-negative constraints. Hence the model becomes a mixed integer program and requires fewer integer variables than the basic integrated model used by Cordeau et al. (2001). In addition, its linear programming relaxation is tighter.

They further observe that the collection of all the maintenance routing solutions can be reduced to a much smaller set containing only those distinct maintenance routing solutions that represent the *unique maximal short connect sets* (UMs). Given a set of maintenance solutions which contain the same set of short connects, it is observed that these maintenance solutions have the same impact on the crew pairing decisions. Hence, all the maintenance solutions which contain the same short connect set can be represented by the associated short connect set. This is referred to as *uniqueness*. If a maintenance solution has a short connect set A and another maintenance solution has a short connect set B, where $B \subset A$, then the choice of short connects to be used in crew pairings provided by A is more than B. Thus, it suffices to include only the first maintenance solution A. In other words, it suffices to include maintenance solutions representing only *maximal short connect sets*.

Cohn and Barnhart (2003) proposed two approaches to solve the integrated problem. The first approach solves a modified string model for the aircraft maintenance routing problem in order to get a set of UMs (may not be the complete set of all the UMs). Then the extended crew pairing model is solved by including these UM sets as the maintenance solutions. This can be viewed as a restricted version of the complete problem, since only a subset of the maintenance solution variables are involved. In fact, the traditional way of airline schedule planning can be viewed as a special case, where only one maintenance solution is provided which may or may not represent a UM. Thus, a feasible solution by this approach is guaranteed to be at least as good as that found using the traditional sequential approach.

For the second approach, the problem is solved as a constrained crew pairing model. First, a crew pairing problem is solved with all potential short connects permitted. If the short connects used in the pairing solution are maintenance feasible, the optimal solution for the integrated model is obtained. Otherwise, a cut is added to eliminate the current infeasible crew pairing solution and the solution procedure continues. This cut is generated by identifying a *minimally infeasible short connect set* (MIS) of the current pairing solution. Formally, given a crew pairing solution $\bar{P}$ with the associated short connect set $\bar{T}$, for a set $\bar{T}'$ to be minimally infeasible, it must satisfy that $\bar{T}' \subseteq \bar{T}$ and any proper subset of $\bar{T}'$ is maintenance feasible while itself is not. The cut is written as below.

$$(MIS - CUT) \qquad \sum_{p \in P} \sum_{t \in \bar{T}'} \eta_{tp} y_p \leq |\bar{T}'| - 1 \tag{2.26}$$

This cut prohibits the crew pairing solution containing $\bar{T}'$. Here, Cohn and Barnhart want maintenance infeasible short connect sets of $\bar{T}$ to be as small as possible, so that a minimal number of cuts are needed in solving the constrained crew pairing problem.

Chapter 3

Solving the Integrated Model by Benders Decomposition

The ECP model can be viewed naturally as a two-stage decision problem to be solved by the Benders decomposition (Benders, 1962). In this chapter, we first review the Benders decomposition, then different cuts are proposed for the ECP model. We detail the methodology of solving the Benders subproblem and lastly a Benders algorithm is given.

3.1 Benders Decomposition Review

Consider the following mixed integer program.

$$z = \max cx + hy \tag{3.1}$$

subject to

$$Ax + Gy \leq b \tag{3.2}$$

$$x \in X \subseteq Z_+^n,\ y \in R_+^p \tag{3.3}$$

3.1.1 Benders Reformulation

Here, we view the integer variables x as complicating variables. Suppose x have been fixed, denoted as $\bar{x}$, the original problem becomes a linear program listed below.

$$LP(\bar{x}) \quad Z_{LP}(\bar{x}) = \max\{hy : Gy \leq b - A\bar{x}, y \in R_+^p\} \tag{3.4}$$

The dual problem is

$$\min\{u(b - A\bar{x}) : uG \geq h, u \in R_+^m\} \tag{3.5}$$

We can characterize whether $LP(\bar{x})$ is infeasible or has a bounded optimal value or has an unbounded optimal value by looking its dual polyhedron (Nemhauser and Wolsey, 1988). Define $\{u^k \in R_+^m : k \in K\}$ to be the set of extreme points of $Q = \{u \in R_+^m : uG \geq h\}$ and let $\{v^j \in R_+^m : j \in J\}$ be the set of extreme rays of $\{u \in R_+^m : uG \geq 0\}$. If $Q \neq \emptyset$, then $\{v^j \in R_+^m : j \in J\}$ is in fact also the set of extreme rays of Q. It is noted that the extreme points set $\{u^k\}$ and extreme rays set $\{v^j\}$ are independent of value $\bar{x}$.

- When $Q \neq \emptyset$, the dual problem (3.5) can have an optimal objective value or be unbounded.
  - If (3.5) has a finite optimal objective value, $v^j(b - A\bar{x}) \geq 0$ for all $j \in J$. Therefore, $Z_{LP}(\bar{x}) = \min_{k \in K} u^k(b - A\bar{x}) < \infty$, which means $LP(\bar{x})$ has and optimal solution;

it also can be written as

$$Z_{LP}(\bar{x}) \leq u^k(b - A\bar{x}) \text{ for all } k \in K$$

Furthermore, the above constraint is valid for any feasible x, therefore, we can have cuts

$$Z_{LP}(x) \leq u^k(b - Ax) \text{ for all } k \in K$$

- if (3.5) is unbounded, $v^j(b - Ax) < 0$ for some $j \in J$, therefore, $Z_{LP}(\bar{x}) = -\infty$. This means the original $LP(\bar{x})$ is infeasible. Hence, we need to have cuts as below to cut away this infeasibility.

$$v^j(b - Ax) \geq 0 \text{ for all } j \in J$$

- When $Q = \emptyset$, the dual problem (3.5) is infeasible. Consequently, $LP(\bar{x})$ can be either infeasible or unbounded.
  - When $v^j(b - A\bar{x}) \geq 0$ for all $j \in J$, $Z_{LP}(\bar{x}) = \infty$. $LP(\bar{x})$ is unbounded in this case. Therefore, the original problem is unbounded.
  - Otherwise, $Z_{LP}(\bar{x}) = -\infty$. $LP(\bar{x})$ is infeasible in this case, which is discussed before.

Formally, the original MIP can be stated as follows when $Q \neq \emptyset$.

$$z = \max_x (cx + \min_{k \in K} u^k(b - Ax)) \tag{3.6}$$

$$v^j(b - Ax) \geq 0 \quad \text{for all } j \in J \tag{3.7}$$

$$x \in X \tag{3.8}$$

The original MIP can be reformulated as following.

$$z = \max \eta \tag{3.9}$$

$$\eta \leq cx + u^k(b - Ax) \quad \text{for } k \in K \tag{3.10}$$

$$v^j(b - Ax) \geq 0 \quad \text{for } j \in J \tag{3.11}$$

$$x \in X, \eta \in R^1 \tag{3.12}$$

We call (3.9)-(3.12) the Benders master problem; fixed the value of x, (3.4) and (3.5) are the primal and dual form of the Benders subproblem respectively. Here (3.10) refers to optimality cuts and (3.11) refers to feasibility cuts. In general, there are enormous number of constraints, but only a small number of constraints are active in the optimal solution. Therefore, it is natural to generate constraints on an as needed basis.

3.1.2 Benders Decomposition Algorithm

The Benders relaxed master problem is defined by (3.9)-(3.12) with only a subset of the constraints included. The Benders decomposition algorithm is as below.

Step 1. Find an initial $x_0 \in X$. Set the lower bound $LB \leftarrow -\infty$ and upper bound $UB \leftarrow +\infty$

Step 2. For current x_i, solve the dual of the Benders subproblem.

2.1 If the dual subproblem has an optimal solution, we find a feasible solution to the primal subproblem. Update the lower bound with the current solution if needed. If $UB - LB < \epsilon$, stop. Otherwise, add a new constraint $\eta \leq cx + u^k(b - Ax)$ to the relaxed master problem. Go to Step 3.

2.2 If the dual subproblem is infeasible, the primal subproblem can be unbounded or infeasible. If the primal subproblem is unbounded, the original problem is unbounded, and the algorithm terminates. If the primal subproblem is infeasible, go to Step 2.3.

2.3 If the primal subproblem is infeasible for current x_i, add the feasibility cut $v^j(b - Ax) \leq 0$ to the relaxed master problem. Go to Step 3.

Step 3. Reoptimize the relaxed master problem. If there is no feasible solution, stop. The original problem is infeasible. If a new solution x_i is obtained, set the optimal value to be the new upper bound. If $UB - LB < \epsilon$, stop; otherwise, go to Step 2.

By using this algorithm, we add a constraint to the Benders relaxed master problem for each iteration. Hence the upper bound provided by the relaxed master problem (Step 3) decreased monotonically; whereas the value of the feasible solution at each iteration (lower bound procedure in Step 2.1) may not increase monotonically.

3.2 Benders Reformulation for Integrated Model

Now, let's consider the integrated model (2.20)-(2.25). We view the crew pairing problem as the first-stage problem, which involves only variables y_p. The selection of a maintenance routing solution is viewed as the second-stage problem and can be expressed as a linear program which involves only variables x_s. It is interesting to note that this approach reverses the traditional airline planning order.

Given a first-stage pairing solution $\bar{P}$ with the associated short connect set $\bar{T}$, the Benders subproblem (BS) becomes a feasibility problem as follows.

$$(BS) \quad \min 0 \tag{3.13}$$

subject to

$$\sum_{s\in S} \beta_{ts} x_s \geq 1 \qquad \forall\ t \in \bar{T} \tag{3.14}$$

$$\sum_{s\in S} \beta_{ts} x_s \geq 0 \quad \forall\ t \in T \setminus \bar{T} \tag{3.15}$$

$$\sum_{s\in S} x_s = 1 \tag{3.16}$$

$$x_s \geq 0 \qquad \forall\ s \in S \tag{3.17}$$

It is note that we can removed the redundant constraints (3.15), since they are always feasible. The dual of this subproblem (DBS) can be expressed as

$$(DBS) \quad \max \sum_{t\in \bar{T}} \mu_t - \omega \tag{3.18}$$

subject to

$$\sum_{t\in \bar{T}} \beta_{ts}\mu_t + \sum_{t\in T\setminus \bar{T}} \beta_{ts}\mu_t \leq \omega \qquad \forall\ s \in S \tag{3.19}$$

$$\mu_t \geq 0 \qquad \forall\ t \in T \tag{3.20}$$

$$\omega \quad \text{unrestricted} \tag{3.21}$$

Since $\mu_{t,t\in T\setminus \bar{T}}$ is the dual constraints for redundant constraints (3.15), $\mu_{t,t\in T\setminus \bar{T}} = 0$ is always feasible. It is not difficult to see that solution $(\mu_t = 0, \omega = 0)_{t\in \bar{T}}$ satisfies the dual constraints (3.19). Given a first-stage crew pairing solution $\bar{P}$, if its Benders subproblem is feasible, an optimal solution for the integrated problem is found. Failing this, an extreme ray for the dual subproblem can be generated to feed back to the first stage to add in a feasibility cut to cut away the current first-stage solution.

The Benders master problem for ECP can be described as follows.

$$\min \sum_{p \in P} c_p y_p \tag{3.22}$$

subject to

$$\sum_{p \in P} \delta_{fp} y_p = 1 \quad \forall\, f \in F \tag{3.23}$$

$$\sum_{t \in T} \mu_t^i \sum_{p \in P} \eta_{tp} y_p \le \omega^i \qquad \forall\, i \in I \tag{3.24}$$

$$y_p \in \{0,1\} \quad \forall\, p \in P \tag{3.25}$$

Here, $(\mu_t^i, \omega^i)_{t \in T}$ is an extreme ray of the polyhedron defined by (3.18)-(3.21) and I is the index set of the extreme rays.

3.3 A Feasibility Cut for the Integrated Model

If the BS is infeasible, then a solution defined by a minimally infeasible subset $\bar{T}'$ of $\bar{T}$ can induce an extreme ray of the DBS. As mentioned in Section 2.3.3, a minimally infeasible set is a minimal set of short connects such that removing any of its elements makes the set maintenance feasible. The induced extreme ray from $\bar{T}'$ is given by ($\mu_t = 1$, for $t \in \bar{T}'$, $\mu_t = 0$, for $t \in T \setminus \bar{T}'$, $\bar{\omega} = |\bar{T}'| - 1$). The feasibility cut generated from this extreme ray is

$$(MIS - CUT) \qquad \sum_{p \in P} \sum_{t \in \bar{T}'} \eta_{tp} y_p \le |\bar{T}'| - 1$$

This is exactly the cut (2.26) proposed in Cohn and Barnhart (2003).

3.4 Amended Benders Subproblem and a New Cut

With artificial variables $\{a_t\}_{t\in\bar{T}}$ added in constraints (3.14), the Benders subproblem can be reformulated as

$$(SP) \quad \min \sum_{t\in\bar{T}} a_t \tag{3.26}$$

subject to

$$\sum_{s\in S} \beta_{ts} x_s + a_t \geq 1 \quad \forall\, t \in \bar{T} \tag{3.27}$$

$$\sum_{s\in S} x_s = 1 \tag{3.28}$$

$$x_s \geq 0 \quad \forall\, s \in S \tag{3.29}$$

$$a_t \geq 0 \quad \forall\, t \in \bar{T} \tag{3.30}$$

Define T_s as the short connect set contained in maintenance routing solution s. The objective here is to minimize the $|\bar{T} \setminus T_s|$. The corresponding dual is

$$(DP) \quad \max \sum_{t\in\bar{T}} \mu_t - \omega \tag{3.31}$$

subject to

$$\sum_{t\in\bar{T}} \beta_{ts} \mu_t \leq \omega \qquad \forall\, s \in S \tag{3.32}$$

$$\mu_t \geq 0 \qquad \forall\, t \in \bar{T} \tag{3.33}$$

$$\mu_t \leq 1 \qquad \forall\, t \in \bar{T} \tag{3.34}$$

$$\omega \quad \text{unrestricted} \tag{3.35}$$

Finding the optimal value of SP is equivalent to finding a maintenance routing solution $\bar{s}$ such that it has the maximal number of short connects from $\bar{T}$. A cut generated from this solution can be captured as

$$(CUT1) \qquad \sum_{p\in P}\sum_{t\in \bar{T}} \eta_{tp} y_p \le |T_{\bar{s}} \cap \bar{T}| \tag{3.36}$$

Clearly when the BS is infeasible, $|T_{\bar{s}} \cap \bar{T}| \le |\bar{T}| - 1 < |\bar{T}|$. the above constraint gives a feasibility cut to cut away the current first-stage solution. Also, we know $CUT1$ is stronger than the cut

$$\sum_{p\in P}\sum_{t\in \bar{T}} \eta_{tp} y_p \le |\bar{T}| - 1$$

which is a weaker cut proposed in Cohn and Barnhart (2003).

3.5 Solving Benders Subproblem and Generating Cuts

In order to solve the integrated model, we need to solve two types of problems repeatedly. One problem is to ascertain whether a set of short connects is maintenance feasible. The other is to identify a feasibility cut in the form of $CUT1$ or $MIS - CUT$.

3.5.1 Checking Feasibility of a Short Connect Set

To ascertain whether a short connect set $\bar{T}$ is maintenance feasible, we can solve MR (defined by (2.1)-(2.6)) by assigning each route string r with a cost $-c_r$, where $c_r = |T_r \cap \bar{T}|$. Here, T_r is the short connect set contained in the routing string r. Then the short connect set is maintenance feasible if MR has an optimal value of $-|\bar{T}|$.

3.5.2 Generating $CUT1$

To generate a $CUT1$, we have to find a maintenance routing solution $\bar{s}$ so that $|T_{\bar{s}} \cap \bar{T}|$ is as big as possible. At the same time we would like $\bar{s}$ to represent a unique maximal short connect set so that it is non-redundant in S. This can be achieved by solving an MR with a modified route string cost defined by $-c1_r \times V - c2_r$, where $c1_r = |T_r \cap \bar{T}|$ and $c2_r = |T_r \setminus \bar{T}|$. V is a constant penalty value such that $V > |T|$. Let us call this modified problem as MMR.

It is interesting to note that checking the maintenance feasibility can be viewed as a special case of MMR. That is, a short connect set is maintenance feasible only when its associated MMR has a value less than or equal to $-|\bar{T}| \times V$. Hence we can enlarge the scope of checking the feasibility to obatin a $CUT1$ by solving an MMR only once.

3.5.3 Generating $MIS - CUT$

Assume that the short connect set $\bar{T}$ is not maintenance feasible. Denote by U the set of the maintenance routing solutions we have found up to the current iteration. Here, U is a subset of S and each maintenance routing solution in U represents a unique maximal short connect set. As $\bar{T}$ is maintenance infeasible, it has a subset which is minimally infeasible. To identify it, we perform the following steps:

Step 1. Find a subset of $\bar{T}$, denoted by $\bar{T}'$, that is not contained in any of the maintenance routing solutions in U.

Let λ_t be a vector representing $\bar{T}'$, i.e., $\lambda_t = 1$ when $t \in \bar{T}'$ and $\lambda_t = 0$ when $t \in \bar{T} \setminus \bar{T}'$.

Then $\bar{T}'$ can be found by solving the following program (Cohn and Barnhart 2003):

$$\min \sum_{t \in \bar{T}} \lambda_t \tag{3.37}$$

subject to:

$$\sum_{t \in (\bar{T} \setminus T_s)} \lambda_t \geq 1 \quad \forall\, s \in U \tag{3.38}$$

$$\lambda_t \in \{0, 1\} \quad \forall\, t \in \bar{T} \tag{3.39}$$

Step 2. Check whether $\bar{T}'$ is maintenance infeasible by solving an MMR. If $\bar{T}'$ is found to be maintenance infeasible, we have found a minimally infeasible set and the steps stop. Otherwise, follow the approach described in the following section to generate a new UM not in U and we can add this new solution to U and return to Step 1 to repeat the searching.

3.5.4 Generating More UM Sets

The quality of UM sets as well as the speed of the generation procedure are critical in solving ECP using MIS cuts. We are interested in finding those UM sets which can cover the $\bar{T}$ as much as possible, so that we can identify the MIS of $\bar{T}$ quickly. Thus, we want to generate UM sets such that short connects of it appearing in $\bar{T}$ as many as possible. Then we attend to generate UM containing short connects in $T \setminus \bar{T}$ as many as possible. In particular, we exploit two approaches to generate UM sets for identifying MIS. In the first approach we generate new UM sets by changing the string cost of MMR. In the second approach we add side constraints to MMR in order to produce new UM sets.

Generating New UM sets By Changing String Cost

We can generate new UM sets by changing the value of c_r, which is similar to what we do is Section 3.5.2. This problem can be viewed as a multiple objective problem. The objectives are listed below.

1. The first objective is to maximize the number of short connects in $\bar{T}$ but not in any UM sets yet;
2. The second objective is to maximize the number of short connects in $\bar{T}$;
3. The third objective is to maximize the number of short connects not appearing in any UM sets yet.
4. The last objective is to use as many short connects as possible, thus a UM set is generated.

The importance of the objectives decreases sequentially. Therefore, we set the value $c_r = -c1_r \times V^3 - c2_r \times V^2 - c3_r \times V - c4_r$, where $c1_r$, $c2_r$, $c3_r$ and $c4_r$ are the number of the short connects contained in r for the 4 categories listed above respectively. The cost coefficients of case 1 and 2 dominiating the cost coefficients of cases 3 and 4. Hence, this MMR tends to search UM of $\bar{T}$ intensively. The cost of 1 and 3 drive this MMR to use new short connects, in other words, it assures diversity of short connects used in the maintenance solution. The cost of case 4 makes sure the solution is UM.

Generating New UM sets By Adding Constraints

We can also generate new UM sets by solving an MMR with a constraint forcing a maintenance solution to contain at least one short connect t such that $t \in \bar{T} \setminus T_{\bar{s}}$. Formally, the constraint is listed below.

$$(MC1) \qquad \sum_{t \in \bar{T} \setminus T_{\bar{s}}} \sum_{r \in R} \theta_{tr} z_r \geq 1 \quad \forall s \in S$$

On the other hand, we want to search UM over T diversly so that each short connect has the potential to be included in S. This is done by solving MMR and forcing the maintenance solution to contain at least one short connect t such that $t \in T \setminus (\bar{T} \setminus T_{\bar{s}})$. Formally, the constraint is listed below.

$$(MC2) \qquad \sum_{t \in T \setminus (\bar{T} \cap T_{\bar{s}})} \sum_{r \in R} \theta_{tr} z_r \geq 1 \quad \forall s \in S$$

Since $|T \setminus (\bar{T} \setminus T_{\bar{s}})|$ can be large, the possible MC2 can be much more than MC1. We can eliminate the number of MC2 without losing diversity of the short connects by modifying it as below:

$$(MC2') \qquad \sum_{t \in T \setminus (\bigcup_{s \in S} \bar{T} \cap T_{\bar{s}})} \sum_{r \in R} \theta_{tr} z_r \geq 1 \quad \forall s \in S$$

Constraints set MC1 ensure that for every new solution s', $T_{s'} \cap \bar{T}$ is a different UM over $\bar{T}$. MC2 (MC2') ensures that $T_{s'}$ is a different UM of T. By adding these side constraints, it is possible to generate n UM sets in at most the amount of time it takes to solve n MMR. In fact, it may take significantly less time, because at each iteration we can use the previous iteration's maintenance routing solution as an advanced start.

In order to get a MIS from $\bar{T}$, we keep on adding constraints MC1 and MC2 (MC2') to MMR until we find one. If we get a new $\bar{T}'$ which is maintenance infeasible, we need to delete the previous constraints and then add a different set of constraints based on $\bar{T}'$ in order to get another MIS. This is more complicated comparing with changing the cost of strings, since in the later we do not change the problem matrix when $\bar{T}$ changes to $\bar{T}'$. Hence, in the real implementation, we use the later approache to generate more UM sets. In fact, we dramatically reduce the number of the UM sets required to produce the MIS comparing with the approach used in Cohn and Barnhart (2003). In an example, we reduced the number of the UM sets required for generating a MIS of size 3 from 9 to 3. In another example, the number of UM sets needed for a MIS of size 5 is reduced from over 100 to only 7.

3.6 Description of the Solution Procedure

The solution procedure for the integrated problem is summarized below.

Step 1. Initialize the relaxed master problem, defined by (3.21)-(3.23), with $I = \emptyset$. In other words, all short connects are assumed to be feasible crew connections.

Step 2. Solve the current version of the relaxed master problem. The optimal objective value provides an updated lower bound on the integrated problem. Let $\bar{T}$ be the set of short connects used in this solution.

Step 3. Test the maintenance feasibility of $\bar{T}$ by solving an MMR. If $\bar{T}$ is maintenance feasible, an optimal solution is found and the procedure stops. Otherwise, proceed to the next step.

Step 4. Use the result from MMR solved in Step 3 to generate a CUT1. Alternatively an MIS-CUT can be generated by following the steps described in Section 3.5.4.

Step 5. Add the newly found cut to the relaxed master problem and return to Step 2.

If needed, the algorithm can always provide a feasible solution to the ECP model at any iteration by solving a restricted ECP model using the current known UM sets as the maintenance routing variables. In order to get a good solution quickly, we can generate more UM sets by applying techniques introduced in Section 3.5.4 after we solve the relaxed master problem in the first iteration.

Chapter 4

Computational Issues

In this chapter we discuss several issues when we solve the ECP model.

4.1 The Test Problems

Experiences seem to suggest that it is laborious to identify a minimally infeasible short connect set (Cohn and Barnhart, 2003). At the other extremes, sometimes the initial crew pairing solution may turn out to be maintenance feasible (Klabjan et al., 2002). The solving time is generally quite long even for a medium size problem. Thus, we need an efficient way in order to test the performance of the different cuts.

We generate a few test cases. The design of these test cases highlights several factors that can affect the structure of the optimal solutions and the efficiency of the algorithms. These factors include: schedule network structure; pairing cost and the problem size.

These data are based on schedules from a commercial airline. Every set of the test cases is extracted from a real airline schedule and contains a single hub and a few spokes. However, since the number of the short connects in the real schedule is small, we manually create more short connects so that more MIS are contained in the schedule. We first aggregate the nodes on the schedule (Hane et al., 1995). The departure and arrival times at the aggregated nodes are then adjusted so that more short connects are produced. Therefore, the same set of test cases contain the same time-space network for both crew and aircraft. As an example, a small schedule for the testing purpose is listed in Appendix A.

For each flight schedule, we generate 20 different test cases by perturb every pairing cost randomly (2.5%) using different sequences of seed numbers. Therefore, the optimal solutions may be different for each of the test cases although they share the same network.

4.2 String and Pairing Generation

Since the schedule we are dealing with is small, we generate all the strings and pairings and store them in the memory during the computer solving procedure.

4.2.1 String Generation

Before the string generation, we need to build up the flight network. We first duplicate the flights as many times as the maximal number of calender days allowed for a feasible string. In this flight network, nodes represent the flights, arcs represents the possible connections between flights. Two flights are connected if the following constraints are satisfied.

- Previous flight's arrival station = Next flight's departure station
- Previous flight's arrival time + minimal turn time $\leq$ Next flight's departure time

- Previous flight's arrival time + one day > Next flight's depature time

Also, we introduce a source node and a sink node into the network. The source node connects with those flights departing from the maintenance stations. The sink node connects with those flights arrival at the maintenance stations.

In order to generate strings, we perform a depth first search on the flight network. A few constraints are considered when searching the network, e.g, a string must start and end at maintenance stations; the duration of a string must be no longer than the maximal duration between two maintenances; no flights are allowed to appear more than once in a string.

4.2.2 Pairing Generation

As we discussed in Section 2.2.3, there are two different approaches to generate pairings, in particular, by the flight network or by the duty network. The advantage of the duty network is that some rules can be inherited in duties. Consequently, during the depth first search on the duty network, less constraints need to be considered. Whereas more rules have to be satisfied when searching through the flight network, which requires more computer programming efforts. The advantage about using the flight network is that it can handle a much larger schedule. Since our schedule is small, we choose to use the duty network.

We first generate all the duties with duty rules considered, e.g. maximal duty duration regulation (FAA, 2002). Duties are stored in the memory. The cost of a duty is computed on the fly. After the duty generation, we duplicate all the duties as many as the maximal number of calender days allowed in a pairing. Before constructing pairings, we build a duty network. In this network, nodes represent duties, arcs represent the possible connection between duties. Two duties are connected in the duty network if the following constraints are satisfied.

- Previous duty end station = Next duty begin station
- Enough rest time is given between duties
- No common leg is included in both duties

Also, a source node and a sink node are introduced into the network. A set of arcs connect the source node with all the duties starting from crewbases. The other set of arcs connect the sink node with all the duties ending at crewbases.

A depth first search algorithm is used to generate pairings. A few constraints have to be considered during the search, which are listed below:

- The pairing must start and end at the same crewbase;
- 8-in-24 must be satisfied in the pairing (FAA, 2002);
- The maximal pairing elapsed time limitation is satisfied;
- No flight appears more than once in a pairing

The pairing cost is also computed on the fly when generating pairings. Table 4.1 lists the parameters which we used to generate duties and pairings as well as the duty and pairing cost.

4.3 Issues in Using CPLEX

We use the CPLEX callable library to solve the IP and LP in our research. There are a few issues in using CPLEX to solve the IP, which are discussed below.

Parameter	**Value**
mg	3 hrs
τ_d	$4 \setminus 7$
τ_p	$2 \setminus 7$
npd	3
Briefing Time	1 hrs
Debriefing Time	15 mins
Maximum Duty Period	12 hrs
Minimum Rest Period	8 hrs
Maximum Pairing Period	4 days
Maximum Maintenance Date	4 days
Minimum Turn Time	30 mins
Minimum Sit Time	45 mins

Table 4.1: Parameters for duty and pairing construction

4.3.1 Memory Problems in Using CPLEX

The ILOG CPLEX Mixed Integer Optimizer solves MIP models using a very general and robust branch-and-bound algorithm (ILOG, 2002b). It is quite often that the branch-and-bound tree becomes so large that insufficient memory is left for the remaining LP subproblems. There are a few ways to avoid this memory failure.

First, we can set the working memory parameter ($WorkMem$) to a size which is much smaller than the system available memory. Typically, a tree size is also specified to make sure that the CPLEX branch-and-bound tree operates in a safe memory limit. This is done by setting parameter $TreLim$. Because CPLEX needs to store all the nodes in the memory, it is quite often that the tree size grows over the system available memory. To overcome this difficulty, CPLEX provides some features which can store some parts of the branch-and-bound tree in files so that more nodes are explored and hopefully a desired solution is found. In particular, this is done by setting the $NodeFileInd$. If the $NodeFileInd$ is set to 1 (default value), then nodes are compressed to conserve the memory, but are still stored in the memory. If it is set to 2, node files are written to the disk without compression. If the parameter is set to 3, nodes are both compressed (as in option 1) and written to disk (as in option 2). In our research, we use option 2 since our problem is not too big and the node file size is small comparing with the available disk space.

4.3.2 Branch On Follow-ons

When solving a problem with covering constraints, we can use branch on follow-ons as a branching rule to find an IP solution. Given a fractional solution to the LP relaxation of the original problem (the crew pairing problem or the string model for the maintenance routing problem), there must exist two pairings/strings whose associated variables are fractional such that they both contain a common flight r and there exists another flight s where one pairing contains it and the other does not. Branch on follow-ons requires two flights (r and s) appearing consecutively in pairing/string in one branch; in the other branch, two flights are not allowed to appear consecutively in any pairings/strings.

In order to implement such a branching rule, we have to use the CPLEX MIP call-back routings (ILOG, 2002a). In particular, we use $CPXsetbranchcallbackfunc$ to set and modify the user-written callback routine to be called. This callback routine is called after CPLEX selects a branch, but before the branch is carried out during the MIP optimization. Hence, the branch that CPLEX selected is replaced by a user-selected branch. Then we use $CPXsetnodecallbackfunc$ to set and modify the user-written function to be called during the MIP optimization after CPLEX has selected a node to explore, but before this exploration is carried out. Thus, this callback routine changes the node to be explored from the CPLEX selection to a user preference.

Before writing the branching rules, we set $CPX\_PARAM\_MIPCBREDLP$ to 0 first. This ensures that the user can manipulate the original problem in the specified branching routine, rather than the *presolved* problem. Presolve is a process to reduce the user input problem by examining the logical reduction opportunities. During the presolve process, CPLEX builds an entirely new problem and stores enough information to translate a solution to this problem back to a solution to the original problem. However, during branching, we are trying to branching on the original variables (pairings or strings). There are two ways to deal with the mismatching between presolved problem variables and original problem variables. CPLEX provides Advanced Presolve Interface, which allows the user to map between original variables and presolved variables. Therefore, the user can translate pairing decision variables to presolved variables and manipulate it. However, more computer programming effort is required to implement this option. The other option is to set $CPX\_PARAM\_MIPCBREDLP$ to $CPX\_OFF(0)$, which tells CPLEX to translate automatically between the original data and presolved data. In this research, we implement the second option.

We use $CPXbranchcallbackbranchbds()$ to set the branching rule in the callback routine. By calling $CPXgetcallbacknodex$ function, we are able to get the solution for the subproblem at the current node of the branch-and-bound tree. Hence, we are able to find two flights which satisfy the branch on follow-ons condition when no feasible integer solution is available. For each of the pairing, if it contains both flights consecutively, we set the lower bound of the pairing decision variable to be 1 (thus $x_p = 1$) and put it to the first branch; otherwise, we set the upper bound to be 0 (therefore $x_p = 0$) and put it to the second branch. Lastly by setting parameter $useraction$ to $CPX\_CALLBACK\_SET$, CPLEX knows that it is a user specified branching rule and implements it.

By default, CPLEX explores the branch which is specified first. In our case, the branch which two flights (m, n) appear consecutively is explored first.

4.3.3 Comparison Between Different Data Types

When CPLEX returns a double type variable or parameter value, it is quite often that we need to compare this value, denoted as m, with an integer n. For example, when searching the two flights which satisfy the branch on follow-ons condition, we need to identify whether a double type variable is 0 or 1. However, since the ways to represent an integer or a double type are different in computer, we normally do not use a statement like $m == n$ to make the comparison between different data types. Instead, we can define a very small positive number ϵ, eg. $\epsilon = 0.0001$. If $|m - n| < \epsilon$, we consider these two values are equal.

Chapter 5

Computational Result

In order to study the effectiveness and efficiency of the two types of feasibility cuts, we carry out some computational experiments.

The computational results for solving the 20 integrated problems based on the schedule in Table A.1 are presented in Table 5.1. The leftmost column indicates the names of the test problems. The next three and the last three columns give the statistics for the procedures using CUT1 and MIS-CUT respectively. Columns CUT1-M and MIS-M give the numbers of master problems and their CPU times required. Columns CUT1-S and MIS-S list the numbers of MMRs and their CPU times required. The fourth and the last columns display the total CPU times required to solve the integrated problems. It is note that column 4 is greater than the sum of column 2 and 3, and column 7 is greater than the sum of column 5 and 6. This is because column 4 and column 7 contains the processing time for generating CUT1/MIS-CUT and initializing the LP and IP. All CPU times are counted in seconds. The last row of the table lists the test problems (8 out of 20) that require no feasibility cuts.

The algorithms are coded in C++. CPLEX 8.1 is used to solve LPs and IPs. All the experiments are performed on an Intel Pentium IV 1.9G PC with 512M memory and Windows XP installed. In solving the problem, all crew pairings and maintenance strings were enumerated first and stored in the memory. Altogether there are 103268 pairings and 645499 strings. To solve IPs with branch-and-bound, we use branchings on follow-ons which are executed from the CPLEX MIP callback interface.

The number of feasibility cuts generated for each problem is one less than the number of relaxed master problems solved. From Columns 2 and 5 it is evident that the cut generated by the minimally infeasible set (MIS-CUT) is stronger and more effective than the cut generated by the maximal short connect set (CUT1). However to obtain it, many MMRs need to be solved as indicated in Columns 3 and 6. As a result, this intensively computational requirement completely offsets its effective advantage and the approach by generating CUT1 becomes computationally more efficient for all the tested problems that require feasibility cuts. This can be seen very clearly from Columns 4 and 7. In general the approach using feasibility cuts generated by CUT1 requires merely a fraction of the computational time required by the approach using feasibility cuts generated by MIS-CUT.

Case	**CUT1-M**	**CUT1-S**	**Total**	**MIS-M**	**MIS-S**	**Total**
S1	2	2		2	8	
time	37	485	632	42	2351	3088
S2	4	4		2	7	
time	136	967	1407	43	7523	8758
S3	14	14		3	22	
time	1469	3212	6405	80	44646	47325
S7	13	13		3	20	
time	869	4128	6452	76	43259	45561
S10	5	5		4	30	
time	422	870	1706	107	8165	10441
S12	3	3		2	8	
time	142	604	965	64	1583	2314
S13	15	15		4	22	
time	853	3451	4816	121	50137	53960
S14	2	2		2	9	
time	60	542	727	39	3530	4272
S15	4	4		3	15	
time	208	819	1369	87	2684	3983
S17	2	2		2	9	
time	59	1112	1281	49	3337	4074
S18	9	9		3	13	
time	561	2065	3651	81	3964	5233
S20	2	2		2	7	
time	82	485	673	61	1477	2017
Total	75	75		32	170	
Time	4898	18740	30084	850	172656	191026
S4-S6, S8 S9, S11, S16, S19	1	1		1	1	
time	—	—	—	—	—	—

Table 5.1: Comparison between CUT1 and MIS-CUT

Appendix A

Small Flight Schedule For Testing

Given the schedule in Table A.1, we need some extra information to solve the integrated problem, which is listed below.

- **Crewbase:** DFW ATL BHM COS
- **Maintenance Station:** DAL DFW BJX

Flight	Origin	Departure	Destination	Arrival
1064	DFW	0625	ATL	0936
1093	ATL	0706	DFW	0724
0526	AMA	0620	DFW	0724
1219	BHM	0622	DFW	0724
1031	DAY	0640	DFW	0724
1422	DFW	0754	CVG	1231
2055	DFW	0754	COS	1014
0715	DFW	0754	AMA	1041
0563	DFW	0754	BJX	1207
0571	CVG	0919	DFW	1127
1930	COS	0902	DFW	1127
0542	BJX	0905	DFW	1127
0410	CLT	0945	DFW	1127
1905	ATL	1006	DFW	1127
1196	DFW	1157	BHM	1431
1100	DFW	1157	CLT	1620
1354	DFW	1157	BHM	1329
2052	AMA	1111	DFW	1334
1480	COS	1044	DFW	1334
1546	DFW	1404	ATL	1710
0376	DFW	1404	CVG	1742
1332	DFW	1404	DAY	1751
0692	DFW	1404	ATL	1710
1153	DFW	1404	COS	1525
1371	CVG	1301	DFW	1503
0440	BJX	1137	DFW	1503
1955	BHM	1259	DFW	1503
1956	DFW	1533	AMA	1719
0515	DFW	1533	COS	1707
1996	DFW	1533	CVG	1844
1131	BHM	1501	DFW	1816
1116	DFW	1846	CLT	2222
1067	DFW	1846	AMA	2028
0457	DFW	1846	BJX	2147
1099	CLT	1650	DFW	1935
1302	COS	1555	DFW	1935
1500	AMA	1749	DFW	1935
1189	ATL	1740	DFW	1935
1848	DFW	2005	BHM	2300
0517	ATL	1840	DFW	2100
1501	CVG	1812	DFW	2100
2051	DAY	1821	DFW	2100
1824	COS	1737	DFW	2100
2051	DFW	2130	COS	2313
1652	DFW	2130	DAY	0105
1730	DFW	2130	ATL	0124

Table A.1: Small Test Case Flight Schedule

Bibliography

Anbil, R. , Tanga, R. , and Johnson, E. . A global approach to crew-pairing optimization. *IBM Systems Journal, Vol 31, No 1*, pages 71–78, 1992.

Barnhart, C. , Boland, N. , Clarke, L. , Johnson, E. , Nemhauser, G. , and Shenoi, R. . Flight string models for aircraft fleeting and routing. *Transportation Science, 32, No.3*, pages 208–220, 1998.

Barnhart, C. , Johnson, E. , Nemhauser, G. , and Vance, P. . Crew scheduling. *Handbook of Transportation Science. R.W. Hall, ed. Kluwer Scientific Publishers*, pages 493–521, 1999.

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

Boland, N. , Clarke, L. , and Nemhauser, G. . The asymmetric traveling salesman problem with replenishment arcs. *European Journal of Operational Research, 123*, pages 408–427, 2000.

Clarke, L. , Johnson, E. , Nemhauser, G. , and Zhu, Z. . The aircraft rotation problem. *Annals of OR: Mathematics of Industrial Systems II. R.E. Burkard, T.Ibaraki, and M.Queyranne, eds.Baltzer Science Publishers*, pages 33–46, 1997.

Cohn, A. and Barnhart, C. . Improving crew scheduling by incorporating key maintenance routing decisions. *Operations Research, 51, No. 3*, pages 387–396, 2003.

Cordeau, J. , Stojkoviac, G. , Soumis, F. , and Desrosiers, J. . Benders decomposition for simultaneous aircraft routing and crew scheduling. *Transportation Science, 35, No.4*, pages 375–388, 2001.

FAA, . *Federal Aviation Regulations.* URL: http://www.faa.gov/avr/afs, 2002.

Gamache, M. and Soumis, F. . A method for optimally solving the rostering problem. *Operations Research in the Airline Industry. G.Yu, ed. Kluwer Academic Publishers*, pages 124–157, 1998.

Hane, C. , Barnhart, C. , Johnson, E. , Marsten, R. , Nemhauser, G. , and Sigismondi, G. . The fleet assignment problem: Solving a large-scale integer program. *Math. Programming, 70*, pages 211–232, 1995.

ILOG, . *ILOG CPLEX 8.1 Advanced Reference Manual.* ILOG, 2002a.

ILOG, . *ILOG CPLEX 8.1 User's Manual.* ILOG, 2002b.

Klabjan, D. , Johnson, E. , Nemhauser, G. , Gelman, E. , and Ramaswamy, S. . Airline crew scheduling with time windows and plane count constraints. *Transportation Science, 36, No.3*, pages 337–348, 2002.

Mak, V. and Boland, N. . Heuristic approaches to the asymmetric travelling salesman problem with replenishment arcs. *International Transactions in Operational Research, 7*, pages 431–447, 2000.

Nemhauser, G. and Wolsey, L. . *Integer and Combinatorial Optimization.* Wiley, 1988.

Phillips, R. L. , Boyd, D.W. , and Grossman, T.A. . An algorithm for calculating consistent itinerary flows. *Transportation Science, 25*, pages 225–239, 1991.

Ryan, D. and Foster, B. . An integer programming approach to scheduling. Wren, A. , editor, *Computer Schedule of Public Transport Urban Passenger Vehicle and Crew Scheduling*, pages 269–280. Elsevier Science B. V., 1981.

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