A Stochastic Programming Approach to the Airline Crew Scheduling Problem

Joyce W.Yen
Industrial Engineering
University of Washington
Seattle, Washington, USA
joyceyen@u.washington.edu

John R. Birge
Industrial Engineering and Management Sciences
Northwestern University
Evanston, Illinois, USA
jrbirge@northwestern.edu

A Stochastic Programming Approach to the Airline Crew Scheduling Problem

Abstract

Traditional methods model the billion-dollar airline crew scheduling problem as deterministic and do not explicitly include information on potential disruptions. Instead of modelling the crew scheduling problem as deterministic, we consider a stochastic crew scheduling model and devise a solution methodology for integrating disruptions in the evaluation of crew schedules. The goal is to use that information to find robust solutions that better withstand disruptions. Such an approach is important because we can proactively consider the effects of certain scheduling decisions. By identifying more robust schedules, cascading delay effects will be minimized. In this paper we describe our stochastic integer programming model for the airline crew scheduling problem and develop a branching algorithm to identify expensive flight connections and find alternative solutions. The branching algorithm uses the structure of the problem to branch simultaneously on multiple variables without invalidating the optimality of the algorithm. We present computational results demonstrating the effectiveness of our branching algorithm.

1 Introduction

To increase profits, airlines continually look for ways to better use their resources and to improve scheduling decisions. In 1991 American Airlines spent \$1.3 billion on crew costs (Anbil et al. 1991) while in 1993 United Airlines spent \$0.6 billion just on pilot costs (Graves et al. 1993). Combined crew costs involve billions of dollars of investment, second only to fuel costs among all airline costs, giving airlines incentives to efficiently use their crew resources (Graves et al. 1993). When schedules become disrupted, the potential for even more scheduling inefficiencies increases.

In day-to-day operations there are two components to the airline crew scheduling problem: short-range planning and long-range planning. This process has traditionally been seen as two separate problems. Short-range planning involves making crew assignments under short-term time constraints. (See Teodorivić and Stojković 1995 for a description.) Long-range planning, on the other hand, represents the traditional process where crew schedulers receive flight schedules several months in advance and assign crews their flights. (See Arabeyre et al. 1969, Etschmaier and Mathaisel 1985, and Gershkoff 1989 for general overviews.)

This paper presents a more realistic crew scheduling model that results in a departure from traditional methods. We approach the subject of long-range scheduling while providing for short-range interaction since long-range planning decisions inevitably are altered by short-range decisions. We integrate the two by introducing randomness in the form of a short-range variable in the long-range problem. We call this problem the stochastic crew scheduling problem.

By considering a stochastic model of the crew scheduling problem, we are able to demonstrate significant savings in the expected cost of a solution without compromising solution quality. Furthermore we show that these improved solutions can be obtained in just a few iterations of our algorithm. The solutions to our model are more robust because we reduce the perpetuation of delays via scheduling decisions. These results suggest important savings can be obtained by considering a stochastic model for crew scheduling.

We first present some background on the deterministic crew scheduling problem in Section 2. In Section 3 we present our stochastic formulation. Section 4 outlines our branching algorithm. Our computational results appear in Section 5. Finally we make concluding remarks in Section 6.

2 Background

The general long-range crew scheduling problem (CSP) is modelled as a deterministic integer program whose objective is to find a minimum cost assignment for a given set of crews for a given flight schedule. Typically the problem is modelled as a set partitioning or set covering problem (Nemhauser and Wolsey 1999).

Let a *pairing* be defined as a round trip itinerary that a crew member might fly. Let a *flight segment* be defined as a single flight from origin to destination. The deterministic crew scheduling problem can be formulated as follows: Suppose we have n feasible pairings and m different flight segments to cover. We want to

$$\text{minimize} \sum_{j=1}^{n} c_j x_j \tag{1}$$

subject to

$$\begin{aligned} \sum_{j=1}^{n} e_{ij} x_j = 1, \qquad & \forall i = 1, \ldots, m; \\ x_j = 0, 1, \quad & \forall j = 1, \ldots, n; \end{aligned} \tag{2}$$

where

$x_j = 1$ if pairing j is selected in the solution, 0 otherwise,

$e_{ij} = 1$ if flight segment i is covered by pairing j, and 0 otherwise,

c_j = cost of pairing j.

Rubin (1973) proposed the general technique of solving this massive set partitioning problem by decomposing the problem into more manageable subproblems. Much of the literature on crew scheduling takes this approach with most efforts focused on refining a particular subproblem solution methodology, generating better crew schedule candidates, or attempting to take advantage of problem structure. (See Anbil et al. 1992, Andersson et al. 1998, Barnhart et al. 1998, Chan and Yano 1992, Chu et al. 1997, Desrosiers et al. 1991, Gershkoff 1989, Graves et al. 1993, Hoffman and Padberg 1993, Klabjan et al. 1999, Marsten et al. 1979, Ryan and Foster 1981, Wark et al. 1997, and Wedelin 1995 for examples of such research.)

These long-range deterministic models have made progress in finding better crew schedules; however, none of the efforts has attempted to also address short-range problems by incorporating uncertainty into the problem formulation and solution methodology. Instead disruptions are considered in a separate problem of crew recovery.

This planning process is activated either (1) immediately after the disruption has occurred or (2) in response to the prediction that a disruption is imminent. Contrary to a proactive approach, such short term planning is essentially a reaction to a given event geared to recovering the system on an immediate basis. This short-range problem has been addressed in Argüello et al. 1998, Cao and Kanafani 1997a, Cao and

Kanafani 1997b, Jarrah et al. 1993, Lettovsky et al. 1999, Mathaisel 1996, Rakshit et al. 1996, Stojković et al. 1998 Teodorivić 1985, Teodorivić and Guberinic 1984, Wei et al. 1997, and Yan and Lin 1997.

While successful at a quick recovery, none of these published methods attempts to anticipate changes and disruptions to the schedule. This reactive strategy is reflected not only in published research but also at airline companies where companies spend millions of dollars in recovery costs and on computer systems to help them better react to disruptions (McDowell 1997). Disruptions are expensive and lead to loss of time, money, and customer goodwill.

To prevent some of these expenses, crew schedulers must be more aware of and responsive to potential changes in the schedule. More recently, Schaefer et al. (2000) proposed a stochastic extension to the deterministic CSP; however, their methodology does not capture interaction effects of planes, crews, and schedule recovery. They modify the objective function's coefficient vector to reflect the expected cost of each decision variable. They then solve the standard crew scheduling problem with this new coefficient vector. Their method does not account for disruption interactions between potential crew schedules.

We propose a stochastic programming model for the problem. Our model incorporates the cost of disruptions. In particular, we model this problem as a two-stage stochastic integer program with recourse where our proposed recourse model reflects the long-range and short-range interactions. Under this model we capture interactions between potential crew schedules and identify more robust solutions which can better withstand possible disruptions to the flight schedule.

Because crew scheduling requires assigning integers to the variables under consideration, we use stochastic integer programming (SIP). (Background information on stochastic integer programming can be found in Birge and Louveaux 1997, Haneveld and van der Vlerk 1999, Schultz 1996, and Stougie and van der Vlerk 1997) Most SIP methodologies manipulate non-integer stochastic programming techniques, such as the L-shaped methods and cutting plane methods, in such a way as to satisfy the integer constraints. Current SIP methodologies are not suitable for handling the nonlinear relationship we propose between long-range planning first stage variables and short-range planning second stage variables; thus, we develop a new algorithm for handling our problem.

3 Problem Formulation

Since plane usage and crew assignment turn-around times are often short, long delays can have a cascading effect on future plane and crew assignments. As a result crew schedules which are cost effective in the deterministic case may no longer be so when random disruptions are considered. For example, in 1997 almost 55% of the delays on Air New Zealand's domestic flights, totaling over 90,000 minutes of delay, were directly attributed to a flight crew delay or an upstream delay (a delay occurring earlier in the day) affecting downstream flights (flights occurring later in the day). Since the deterministic crew scheduling model fails to address this problem, we turn to stochastic models.

3.1 Stochastic Programming Formulation

Our method adjusts crew assignments during the planning phase so as to continue to minimize crew costs while creating more robust solutions through identifying pairings that are less sensitive to schedule disturbances. We add the expected delay costs to the deterministic objective function and consider how delays affect flight segments constraints.

We suppose that we will observe the disruptions and obtain a recovery cost. We model the disruption as increases in flight operation times, such as ground holds and actual flight times. These times form a random vector, $\xi(\omega)$, where ω is a random element in some space Ω. We let each ω represent a disruption scenario. For each disruption scenario ω, we have a different recourse. For our purposes here, we assume Ω has finite cardinality and each ω occurs with probability p_ω.

Therefore, we reformulate the problem as:

$$\text{minimize } \mathbf{c}^T\mathbf{x} + \mathcal{Q}(\mathbf{x}) \tag{3}$$

$$\text{subject to } \mathbf{A}\mathbf{x} = \mathbf{b},$$

$$0 \leq \mathbf{x} \leq 1, \mathbf{x} \text{ integer.}$$

In this formulation the vector $\mathbf{c}$ represents the expected cost of flying a given pairing independent of all other pairings, that is, the cost of the pairing without considering delays due to crews switching planes. The constraints $\mathbf{Ax} = \mathbf{b}$ consist of the coverage equation (2) and other crew constraints and $\mathcal{Q}(\mathbf{x}) = \int_\omega Q(\mathbf{x}, \omega) P(d\omega)$ is the expected value of future actions due to disruptions in the original schedule. This formulation is a standard two-stage stochastic (integer) program with recourse (Birge and Louveaux 1997).

3.2 A Nonlinear Recourse Model for $Q(\mathbf{x}, \omega)$

We use a nonlinear recourse program to find $Q(\mathbf{x}, \omega)$ and compute the cost of the delay. The recourse formulation (NLR) with a disruption scenario ω is

$$\textbf{NLR}\text{: } Q(\mathbf{x}, \omega) = \min{}_{d^a, a^a} \sum_j \rho_j (\Delta_{j\omega} - \Delta'_{j\omega}) \tag{4a}$$

subject to

$$a^a_{j\omega} - d^a_{j\omega} \geq t_{j\omega}, \qquad \forall j; \tag{4b}$$

$$a^a_{j\omega} - \Delta j\omega \leq a^s_j, \qquad \forall j; \tag{4c}$$

$$d^a_{j\omega} - \Delta_{j\omega} \geq d^s_j, \qquad \forall j; \tag{4d}$$

$$d^a_{j\omega} - a^a_{p^p_j \omega} \geq g^p_{j\omega}, \qquad \forall j; \tag{4e}$$

$$\Delta'_{j\omega} \geq d^a_{j\omega} - (a^a_{p^p_j \omega} + g^p_{j\omega}), \qquad \forall j; \tag{4f}$$

$$\Delta_{j\omega} \geq 0, \qquad \forall j; \tag{4g}$$

$$\Delta'_{j\omega} \geq 0, \qquad \forall j; \tag{4h}$$

$$d^a_{j\omega} - [a^a_{p^c_{jk} \omega} + g^c_{jk}] x_k \geq 0, \qquad \forall j \in \text{pairing } k; \tag{4i}$$

where j refers to flight segments, $t_{j\omega}$ is flight j's flight time under scenario ω, $d^a_{j\omega}$ ($a^a_{j\omega}$) is *actual* time of departure (arrival) for flight j under scenario ω, d^s_j (a^s_j) is *scheduled* time of departure (arrival), p^p_j is plane predecessor flight for flight j, p^c_{jk} is crew predecessor flight of flight j under pairing k, $g^p_{j\omega}$ (g^c_{jk}) is plane ground time for flight j under scenario ω (crew ground time under pairing k), and ρ_j is the penalty cost for delaying flight j. $\Delta_{j\omega}$ is the total delay to flight j including plane-induced delays and crew-induced delays, while $\Delta'_{j\omega}$ is the non crew-induced delay. Therefore, $\Delta_{j\omega} - \Delta'_{j\omega}$ is the delay to flight j due to crews switching planes. Note that p^c_{jk} and p^p_j are just flight indices. Also, d^a and a^a are vectors of actual departure and arrival times, respectively.

The objective (equation 4a) represents the cost of delays due to crews switching planes. The first three inequalities, equations (4b) - (4d), represent consistency constraints for flight arrival and departure times. Inequality (4e) represents plane precedence constraints and inequality (4i) represents crew precedence constraints. Finally equations (4g) and (4h) indicate nonnegative delay times.

The random vector is $\xi(\omega) = (t_{j\omega}, j = 1, ..., ; g^p_{j\omega}, j = 1, ...)$. Our decision variables are $d^a_{j\omega}, a^a_{j\omega}$, and x_k.

Note that we have nonlinear constraints in equation (4i). If x_k is given, this formulation reduces to a linear program because we can determine the crew predecessor flight for each of the flights. The general formulation of the recourse problem becomes

$$\begin{aligned} Q(\mathbf{x}, \omega) = \quad & \text{minimize } \mathbf{q}^T\mathbf{y} \qquad (RP) \\ & \text{subject to } (\mathbf{W} + \mathbf{G}\mathbf{x}^\mathbf{T})\mathbf{y} = \mathbf{h}(\omega), \\ & \mathbf{y} \geq 0. \end{aligned}$$

We next describe our **Delay Branching Algorithm** for solving this stochastic integer programming problem.

4 Delay Branching Algorithm

If a flight is delayed, then crew scheduling decisions introduce delays only when crews are assigned to switch planes. Crew assignments which follow plane assignments do not introduce delays to the system. We want solutions where crew plane changes are minimized. We capture this concept through a specific type of delay, *switch_delay*.

4.1 Defining Switch Delay

A *switch_delay* is a delay due to a plane change. Suppose a crew is assigned to service two flights, flight i followed by flight j, and flight i experiences a delay. If flight i and flight j are assigned to different planes, then flight j would incur a *switch_delay* because of this crew assignment.

Definition 1 *For flight pair (i, j) the switch_delay due to crew connection in pairing k over all scenarios $\omega \in \Omega$ can be written as:*

$$switch\_delay(x_k, i, j) = \sum_{\omega \in \Omega} p_\omega[(d_{j\omega} - (a^a_{p^p_j \omega} + gnd\_time - d^s_j))\delta_k],$$

where

$$\delta_k = \begin{cases} 1 & if \begin{cases} d_{j\omega} & > 0, \\ x_k & = 1, \\ a_{ik} & = 1, \\ a_{jk} & = 1, \\ d_{i\omega} & > 0, \\ p^c_{jk} & = i, \\ plane(j) & \neq plane(i), \\ d^s_j & < a^a_{i\omega}, \end{cases} \\ 0 & otherwise. \end{cases}$$

In the above definition, *gnd_time* is the required time on ground (equal to the maximum of the crew ground time constraint and the plane ground time constraint) and a_{jk} is the j^{th} row and k^{th} column of constraint matrix $\mathbf{A}$. (Recall if $a_{jk} = 1$, then pairing k services flight j.)

4.2 Algorithm Description

First we describe the algorithm in words and then follow with a formal description of the algorithm. A summary of the algorithm notation can be found in Table 1.

Our algorithm essentially augments existing crew scheduling methods with the evaluation of a recourse problem. In our algorithm we either allow or disallow key flight pairs where crews switch planes in the solution. After solving the set partitioning problem using state-of-the-art deterministic crew scheduling knowledge, we evaluate the expected cost of disruptions via the recourse problem. The recourse problem evaluation determines upon which flight pair to branch. We devise a hierarchy for the flight pairs based on the delay costs (specifically, *switch_delay* costs), placing those pairs with larger delays higher in the hierarchy.

The branching methodology employed uses a variation of constraint branching (Ryan and Foster 1981). This method allows us to constrain a collection of variables. Along one branch, we forcibly include the identified flight pair in a pairing selected in the optimal solution at that node. The other branch forcibly excludes said flight pair from any pairing selected in the solution for that node. By focusing on expensive flight pairs, we eliminate subsets of pairings containing the expensive pair. We evaluate a number of pairings at the same time.

If n is the number of flights in the schedule, then the maximum number of flight pairs is ${}_nC_2$. Therefore the maximum number of nodes in our tree is $O(2^{n^2})$. This number is actually an upperbound on the number of flight pairs since not all flights can be coupled with all other flights. In fact,we only include flight pairs occurring over two planes. In traditional branch and bound, the number of pairings is at most 2^n; therefore, the number of nodes is $O(2^{2^n})$ In practice $m << k$, where m is the true number of feasible flight pairs and k is the true number of feasible pairings, and our branching method produces a much smaller tree than that given by traditional branch and bound. (An example is given in Section 5.2.)

With the exception of solving the set partitioning problem, all other steps in the algorithm are relatively easy to evaluate. Using the SPP solution, the recourse problem becomes a simple, albeit large, linear program. We can solve such linear programs efficiently. Identifying costly flight pairs is also simple. Evaluating each flight's *switch_delay* value is a simple addition/subtraction operation.

At any point, upper and lower bounds can be used to select the next node from which to branch until the algorithm terminates. The user may choose, however, to prematurely terminate the algorithm if the current solution's disruption cost is satisfactorily limited. Furthermore, at any iteration of the algorithm we have a feasible solution which is more robust than the initial solution in that the expected cost of disruptions is less.

Table 1: Notation used in Delay Branching Algorithm

Symbol	**Definition**
C	set of nodes which are available for branching/evaluation
$\mathcal{D}^l$	set of (i,j) flight pairs with a positive *switch_delay* at current node l
D_{ij}	total accumulated *switch_delay* for flight pair (i,j) at current node
$D_{i_l j_l}$	maximum total accumulated *switch_delay* at current node l (as accumulated by flight pair (i_l, j_l))
i,j	flight indices
(i_l, j_l)	flight pair which accrues the maximum *switch_delay* at node l
(i,j)	flight i followed by flight j in a crew schedule
k, κ	pairing index
K	index for the number of nodes already evaluated
$\mathcal{K}_{ij}$	set of pairings selected in current solution which cover both flights in a designated flight pair (i,j)
l	current node, node index
L	set of all nodes in the tree
N	total number of pairings
$\mathcal{P}^l$	set of excluded flight pairs at node l
$\mathcal{R}^l$	set of included flight pairs at node l
z^l	overall stochastic program objective value at node l
$(\mathbf{x}^l)^*$	optimal pairing vector at node l

continued on next page

continued from previous page	
Symbol	**Definition**
$(x^l)^*_k$	k^{th} element of the vector $(\mathbf{x}^l)^*$
$(w^l)^*$	optimal deterministic objective value at node l

Now we are ready to give the formal description of the **Delay Branching Algorithm**.

Step 0: Initialize $C = \{\emptyset\}$, $UB = \infty$, K $= 0$, $l = 0$, $\mathcal{P}^0 = \{\emptyset\}$, $\mathcal{R}^0 = \{\emptyset\}$. Solve

$$(w^0)^* = \text{minimize } \mathbf{c}^T\mathbf{x} \tag{5}$$

$$\text{subject to } \mathbf{Ax} = \mathbf{b};$$

$$0 \leq \mathbf{x} \leq 1, \mathbf{x} \text{ integer};$$

to obtain $(\mathbf{x}^0)^*$.

Step 1: for node l :

Set $C = C \cup \{l\}$. Let

$$z^l = (w^l)^* + \mathcal{Q}((\mathbf{x}^l)^*).$$

Note z^l is a function of $\mathbf{x}^l$, $z^l = z(\mathbf{x}^l) = \mathbf{c}^T(\mathbf{x}^l)^* + \mathcal{Q}((\mathbf{x}^l)^*)$. Set upperbound

$$UB = min\{UB, z^l\}.$$

Set $\mathcal{D}^l = \{(i,j)|switch\_delay((x^l)^*_k, i, j) > 0\}$. (See Definition 1 for a definition of *switch_delay*.) If $\mathcal{D}^l = \{\emptyset\}$, let $C = C \setminus \{l\}$ and go to **Step 2**. If $\mathcal{D}^l \neq \{\emptyset\}$, go to **Step 3**. If $C = \emptyset$, terminate. The optimal solution vector is $\mathbf{x}^*$ such that $z(\mathbf{x}^*) = UB$.

Step 2: Find a node $l+1$ such that

$$w^{l+1} = min_{l' \in C}\{w^{l'}\}.$$

Set $l = l + 1$ and go to **Step 1**.

Step 3: Initialize $D_{i_l j_l} = 0$.

For each $(i,j) \in \mathcal{D}^l$**, if**

$$\begin{aligned} (i,j) &\notin \mathcal{P}^l, \\ (i,j) &\notin \mathcal{R}^l, \\ a_{ik}(x^l)^*_k + a_{jk}(x^l)^*_k &\geq 2, \end{aligned}$$

then set $\mathcal{K}_{ij} = \{k | (x^l)^*_k = 1, a_{ik} = a_{jk} = 1\}$. (Note that if we use the set partitioning formulation, then $|\mathcal{K}| = 1$.)

Let

$$D_{ij} = \sum_{k \in \mathcal{K}_{ij}} switch\_delay((x^l)^*_\kappa, i, j).$$

If $D_{ij} > D_{i_l j_l}$, then $D_{i_l j_l} = D_{ij}$ and set $i_l = i$ and $j_l = j$.

Branch from node l. Let

$$E_{i_l j_l}(k) = \begin{cases} 1 & \text{if } a_{i_l k} = 1 \text{ and } a_{j_l k} = 1, \qquad \text{for } k = 1, \ldots, N, \\ 0 & \text{otherwise} \end{cases}$$

Go to **Step 4**.

Step 4: Branch such that pair (i_l, j_l) is excluded from all solutions along that branch. Add the constraint

$$(\mathbf{E^{K+1}})^T \mathbf{x} = \mathbf{E_{i_l j_l}}^T \mathbf{x} = 0.$$

Label this node K+1. Set $\mathcal{P}^{K+1} = \mathcal{P}^l \cup \{(i_l, j_l)\}$. Set $\mathcal{R}^{K+1} = \mathcal{R}^l$. Set $C = C \cup \{K+1\}$.

Find the optimal solution to

$$w^{K+1} = \text{minimize } \mathbf{c}^T \mathbf{x}$$

subject to sub (K+1)

$$\begin{aligned}
\mathbf{Ax} &= \mathbf{b}; \\
\mathbf{E^T_{i_l j_l} x} &= 0; \\
\mathbf{E^T_{ij} x} &= 0, && \forall (i,j) \in \mathcal{P}^l; \\
\mathbf{E^T_{ij} x} &\geq 1, && \forall (i,j) \in \mathcal{R}^l; \\
0 \leq \mathbf{x} &\leq 1; \\
\mathbf{x} &\ \text{integer};
\end{aligned}$$

with optimal value $(w^{K+1})^*$, and solution vector $(\mathbf{x}^{K+1})^*$.

Go to **Step 5**.

Step 5: Branch such that pair (i_l, j_l) is included in all solutions along that branch by adding the constraint,

$$(\mathbf{E^{K+2}})^T \mathbf{x} = \mathbf{E_{i_l j_l}}^T \mathbf{x} \geq 1,$$

to the subproblem for w^l. Label this node K+2. Set $\mathcal{R}^{K+2} = \mathcal{R}^l \cup \{(i_l, j_l)\}$. Set $\mathcal{P}^{K+2} = \mathcal{P}^l$. Set $C = C \cup \{K+2\}$.

Find the optimal solution to

$$w^{K+2} = \text{minimize } \mathbf{c}^T \mathbf{x}$$

subject to sub (K+2)

$$\begin{aligned}
\mathbf{Ax} &= \mathbf{b}; \\
\mathbf{E^T_{i_l j_l} x} &\geq 1; \\
\mathbf{E^T_{ij} x} &= 0, && \forall (i,j) \in \mathcal{P}^l; \\
\mathbf{E^T_{ij} x} &\geq 1, && \forall (i,j) \in \mathcal{R}^l; \\
0 \leq \mathbf{x} &\leq 1; \\
\mathbf{x} &\ \text{integer};
\end{aligned}$$

with value $(w^{K+2})^*$ and solution vector $(\mathbf{x}^{K+2})^*$. Note $(w^{K+2})^* = (w^l)^*$.

Go to **Step 6**.

Step 6: If $(w^{K+1})^* > UB$, fathom node K+1, and set $C = C \setminus \{K+1\}$.

Let $K = K + 2$ and find a node $l + 1$ such that

$$w^{l+1} = min_{l' \in C}\{w^{l'}\}.$$

Set $l = l + 1$ and go to **Step 1**.

4.3 Proof of Convergence to Optimality

We show this algorithm terminates in a finite number of iterations having found all optimal solutions. Lemma 1 shows the termination with no more branches to explore. Theorem 2 demonstrates termination with all optimal solutions being found.

Lemma 1 *The Delay Branching Algorithm terminates in a finite number of iterations and upon termination, there are no more branches to explore.*

Proof:

Since we have a finite number of flights, we have a finite number of flight pairs and, consequently, a finite number of branches. When branching from a node we either have no more flight pairs to constrain or else we exclude a new flight pair. If a flight pair is identified, we branch and add constraint

$$\mathbf{E_{ij}}^T\mathbf{x} = 0,$$

or

$$\mathbf{E_{ij}}^T\mathbf{x} \geq 1.$$

Given the inequalities when excluding a flight pair, we find a new solution along the branch where the flight pair is excluded and, thus, we cannot repeat a solution. Since there is only a finite number of branches, the algorithm must terminate in a finite number of iterations.

Next, suppose the algorithm has terminated, then $C = \{\emptyset\}$, implying no more nodes to explore in the algorithm.

Consider some node K. Either K has a pair (i_K, j_K) such that $D_{i_K j_K} > 0$ or else no such pair exists at node K. If no such pair exists, we are done. If $D_{i_K j_K} > 0$, there are two possibilities: (a) $(w^K)^* > UB$ or (b)$(w^K)^* \leq UB$. Under case (a), we fathom node K leaving no more branches to explore from node K; otherwise, under case (b), we identify at node K a branching pairing $(\hat{i}, \hat{j})$ which satisfies the conditions of **Steps 3-5** and set $C = C \cup \{K+1, K+2\}$. This is a contradiction to the termination of the algorithm.

It must be either there does not exists a pair (i_K, j_K) such that $D_{i_K j_K} > 0$ or $(w^K)^* > UB$ for all nodes K in the tree. Hence when the algorithm terminates, there are no more branches to be explored. ■

Theorem 2 *The Delay Branching Algorithm terminates with an optimal solution.*

Proof:

Suppose the algorithm terminates without finding all optimal solutions. Consider an optimal solution $\hat{\mathbf{x}}$ which has a lower objective function value than the solution found by the algorithm and which was not found by the algorithm. We will show this leads to a contradiction to $\hat{\mathbf{x}}$'s optimality.

Let $I^* = \{(i_1^*, j_1^*), \ldots, (i_n^*, j_n^*)\}$ be the set of flight segments with plane changes in solution $\hat{\mathbf{x}}$. Begin at the root of the tree and trace through the tree so as not to violate any of the pairs in I^*. We will eventually arrive at an end node M such that

$$\left|\mathcal{R}^M \cap \overline{I^*}\right| + \left|\mathcal{P}^M \cap I^*\right| = 0$$

Note $\hat{\mathbf{x}}$ is feasible for the subproblem at node M and $\mathbf{c}^T(\mathbf{x}^M)^* \leq \mathbf{c}^T\hat{\mathbf{x}}$.

We stop branching at node M if either (a) $\mathcal{D}^M = \{\emptyset\}$ or (b) $(w^M)^* > UB$.

Case a: $\mathcal{D}^M = \{\emptyset\}$**.** Therefore,

$$\begin{aligned} \mathbf{c}^T(\mathbf{x}^M)^* + \mathcal{Q}(\mathbf{x}^M)^* &= \mathbf{c}^T(\mathbf{x}^M)^* \\ &\leq \mathbf{c}^T\hat{\mathbf{x}} \\ &< \mathbf{c}^T\hat{\mathbf{x}} + \mathcal{Q}(\hat{\mathbf{x}}), \end{aligned}$$

a contradiction of $\hat{\mathbf{x}}$ being optimal. So it must be that case (b) has occurred.

Case b: $(w^M)^* > UB$**.** Note, $\hat{\mathbf{x}}$ is not the optimal solution at node M since the algorithm did not find $\hat{\mathbf{x}}$ at node M. Hence $\mathbf{c}^T\hat{\mathbf{x}} > (w^M)^*$. Since $D_{ij} > 0$, $\mathbf{c}^T\hat{\mathbf{x}} + \mathcal{Q}(\hat{\mathbf{x}}) > \mathbf{c}^T\hat{\mathbf{x}}$. Thus

$$\mathbf{c}^T\hat{\mathbf{x}} + \mathcal{Q}(\hat{\mathbf{x}}) > UB.$$

But since the solution vector which provides the value of UB is a feasible solution, we contradict the optimality of $\hat{\mathbf{x}}$.

Hence the algorithm must terminate with all optimal solutions identified. ■

We now have the following corollary:

Corollary 1 *The Delay Branching Algorithm terminates in a finite number of iterations with an optimal solution to the stochastic crew scheduling problem given by Problem 3 and recourse problem NLR.*

Proof: The corollary follows directly from Lemma 1 and Theorem 2. ■

5 Computational Results

Next we report on implementation results for three test problems based on Air New Zealand's 1997 domestic operations.

5.1 Assumptions and Data Preprocessing

We make the following simplifying assumptions when implementing the algorithm:

- Access to a state-of-the-art set partitioning solver that effectively and efficiently solves the deterministic problem. We use the solver described in Johnson et al. 1999.
- All crews fly their schedules as planned regardless of the delay circumstances. Legality considerations and other constraints make this assumption unrealistic in practice; however, the solution obtained is a lower bound for the actual expected cost of delays.
- Plane assignments are static. Flights are assigned to the same planes every day.

- All planes have the same ground time requirements.
- All crew have the same ground time requirements.
- Maximum delay times are bounded.

Based on these assumptions, we enumerate our set of possible crew pairings and generate the random disruption scenarios ω. For each test problem we sample 100 scenarios using a truncated gamma or lognormal distribution for the length of delays. A truncated distribution is used because the data indicate delay times are bounded above. Presumably flights experiencing extraordinary delay times may be cancelled and, therefore, do not appear in our delay data. We assume that each disruption scenario is equally likely.

5.2 Test Problem Results

The three test problems represent a simplification of the Air New Zealand domestic schedule: 9 planes covering 61 daily flights, 10 planes covering 66 flights, and 11 planes covering 79 flights. Each of these three problems represents a possible domestic schedule for Air New Zealand's 737 fleet. The schedule services seven cities (Auckland, Christchurch, Dunedin, Hamilton, Rotorua, Queenstown, and Wellington) with the same schedule being flown each day of the week.

Using the set partitioning solver described in Johnson et al. 1999, we enumerate 1911 feasible pairings for the 9-plane problem, 2737 pairings for the 10-plane problem, and 3296 for the 11-plane problem. We implement the algorithm using AMPL (1993) to formulate the model and CPLEX (1998) to solve the resulting linear program. A master C program, which uses a general purpose priority queue (described in Kelly 2000) to maintain the branching decisions, is used to manage the branching process.

The branching tree becomes very large and, due to computational limitations, we do not explore the entire tree. Nevertheless, our results indicate that we quickly find good solutions and that subsequent improvements may be quite minimal.

Although our tree is large, it is much smaller than that produced by traditional branch and bound. Consider, for example, the 9-plane problem with 1911 feasible pairings. In this problem there are 20 flights departing or arriving at Auckland, 18 at Christchurch, 3 and Dunedin, and 20 at Wellington. Due to location restrictions, the maximum number of flight pairs is ${}_{20}C_2 +_{18} C_2 +_3 C_2 +_{20} C_2 = 536$. This value is an upperbound for the number of flight pairs because time limitations would actually preclude some flights from being paired together and because some flight pairs follow plane assignment and, therefore, are not considered for branching. Consequently, our branching tree produces at most $2^{537} - 1$ nodes, while regular branch and bound produces $2^{1912} - 1$ nodes. Note both numbers are ridiculously large but the observation is that generally our process quickly identifies good candidate solutions.

In order to calculate the objective function value, we must determine the value for the penalty parameter. The cost of a delay depends on a number of issues that are difficult to measure. For example, the cost of losing a passenger can vary. Perhaps the passenger is lost for a single flight. In this case, the cost is relatively low; however, a passenger may choose never to fly that airline again. This passenger loss is quite costly because there is repeated lost revenue. Since, in practice, we are not sure how to measure the cost of delays, we want to examine a set of possible delay values.

We test four variations of each test problem examining how different penalty parameter values affect the solution. We show we obtain large savings in the recourse objective without significant sacrifice in the cost of the first stage objective. We also calculate the value of the stochastic solution (VSS). Calculating the VSS further demonstrates the merits of considering the stochastic model since VSS is the possible gain from solving the stochastic model. In particular $VSS = E_\xi(z(\bar{x}(\bar{\xi}), \xi)) - min_x E_\xi(z(x, \xi))$ (Birge and Louveaux 1997).

In expectation planes always arrive on time. In practice, it is assumed planes meet their scheduled arrival and departure times. That is, we assume that the schedule is designed such that every flight adheres to its expected flight times and there are no delays to the schedule. Thus the solution to the problem ignoring delay is the same as the initial solution to the problem with expected delay since we assume the schedule is constructed such that no delays occur. Consequently the initial solution is the same as the expected value solution. Given these assumptions we calculate $VSS = z(\mathbf{x}^0) - z(\mathbf{x}^*)$, where $\mathbf{x}^0$ is the initial solution and $\mathbf{x}^*$ is the best solution vector.

5.2.1 Comparing Deterministic Solutions and Stochastic Solutions

From the penalty parameter analysis (discussed in more detail Section 5.2.2), we believe it is useful to examine several different versions of a test problem, where each version has a different penalty parameter value. We create four variations per test problem, setting the penalty parameter at 1, 10, 100, and 1000. See Table 2 for the results. The first column identifies the test problem. The second column lists the penalty parameter value. Columns 3 and 4 list the initial solution's value for the first and second stage, respectively. Note this initial solution, $\mathbf{x^0}$, is the solution to the deterministic problem from equations (1) and (2). Columns 5 and 6 show our best solution's first stage and second stage values, respectively. The seventh column gives the value of the stochastic solution, while the eighth column gives the best solution's objective value, $\mathbf{c}^T\mathbf{x}^* + \mathcal{Q}(\mathbf{x}^*)$, relative to the initial solution's objective, $\mathbf{c}^T\mathbf{x}^0 + \mathcal{Q}(\mathbf{x}^0)$. For example, row 2 shows the best solution costs for the 9-plane problem with a penalty parameter of ten is 82.5218 units less than the initial solution and is approximately 98.619% of the initial solution.

While we do not explore the entire branching tree, we do explore enough of the tree such that we believe our solutions are near optimal or optimal since the curve of new optimal solution values flattens. (See Figures 1 and 2); however, we do not have proof of optimality for these results. Table 3 gives a summary of the number of iterations executed until finding the best solutions listed in Table 2, as compared to the total number of iterations calculated for each problem.

Even though we may execute many iterations before finding the best solution, we are very close to the best solution after just a few iterations. For example, consider the 10-plane problem with penalty of 100. We are within five percent of the best solution after only five iterations, within two percent after thirteen iterations, and within one percent after 55 iterations. We get quick savings early in the implementation of the algorithm (see Figures 1 and 2). In this case, we see small changes to $\mathbf{c^T x}$ while simultaneously seeing relatively large changes to the overall objective and the recourse objective. Similar results hold for the 9-plane and 11-plane problems.

Test Problem	**Penalty**	$\mathbf{c^T x^0}$	$\mathcal{Q}(\mathbf{x^0})$	$\mathbf{c^T x^*}$	$\mathcal{Q}(\mathbf{x^*})$	**VSS**	$\frac{z(\mathbf{x^*})}{z(\mathbf{x^0})}$
9-plane problem	1	3889	208.803	3889	208.803	0	1.00
	10	3889	2088.035	3937	1957.513	82.5218	.98619
	100	3889	20880.35	4297	18139.53	2332.81	.90582
	1000	3889	208803.5	4819	179628	28245.5	.8672
10-plane problem	1	4219	247.5961	4220	234.1843	12.4118	.99722
	10	4219	2475.96	4392	2136.543	166.418	.97514
	100	4219	24759.61	5498	19664.14	3816.47	.8683
	1000	4219	247596.1	5498	196641.4	49675.7	.8027
11-plane problem	1	5058	175.87	5059	174.55	.322	.99994
	10	5058	1758.76	5175	1627.15	14.62	.99079
	100	5058	17587.62	5408	15563.02	1674.59	.9260
	1000	5058	175876.2	5408	155630.2	19895.9	.8900

Table 2: Test Problem Results

5.2.2 Penalty Parameter Analysis

An important part of the modelling process is determining the value of the penalty parameter. When the penalty parameter equals one, the recourse cost is exactly the sum of the delay minutes over all flights averaged over 100 scenarios. For these problems, the first stage objective is order 10^3 while the recourse objective is order 10^2. These orders of magnitude are consistent with the 1997 data. The simulated total delay time minutes are the same order of magnitude as the average number of delay times from the 1997 data. In 1997 the total delay minutes per day averages 392.65 minutes.

We examine the effects of different penalty parameter values. The penalty parameter is a reflection of how much disruptions are worth, representing a trade-off between disruption costs and crew costs. As we increase the penalty parameter, we can track this trade-off. Quantifying this trade-off is useful, for example, to conduct a cost analysis comparing schedules with different levels of slack time.

Number of Iterations						
	9 Planes		10 Planes		11 Planes	
Penalty	opt	total	opt	total	opt	total
1	1	1287	289	468	2	998
10	7	1127	10	162	13	197
100	25	995	107	184	76	580
1000	140	386	107	256	76	173

Table 3: Number of Iterations Before Finding Current $\mathbf{x}^*$

As the penalty parameter increase, we reduce our disruption costs in the final solution but pay more crew costs. By implementing the algorithm using several different values for the penalty parameter, we can generate a trade-off curve. (See Table 4 and Figure 3. Table 4 demonstrates the percentage of change between the initial solution's value and the final solution's value. A negative value indicates a decrease in value from the initial solution. For example, in the 9-plane, penalty = 10 problem $z(\mathbf{x}^*)$ is 1.38% smaller than $z(\mathbf{x^0})$.)

Percent Change in Value from Initial Solution									
	9 Planes			10 Planes			11 Planes		
Penalty	z^*	$\mathbf{c^T x}^*$	$\mathcal{Q}(\mathbf{x}^*)$	z^*	$\mathbf{c^T x}^*$	$\mathcal{Q}((\mathbf{x}^*)$	z^*	$\mathbf{c^T x}^*$	$\mathcal{Q}(\mathbf{x}^*)$
1	0	0	0	-.28	.023	-5.42	-.006	.02	-.75
10	-1.38	1.23	-6.25	-2.49	4.10	-13.71	-.21	2.31	-7.5
100	-9.42	10.49	-13.13	-13.17	30.32	-20.58	-7.40	6.92	-11.51
1000	-13.28	23.91	-13.97	-19.73	30.32	-20.58	-10.99	6.92	-11.51

Table 4: Effects of the Penalty Parameter

From these results we conclude there exists some threshold for the penalty parameter after which no additional information or savings will result from a higher penalty. Also, there exists a region where the rate

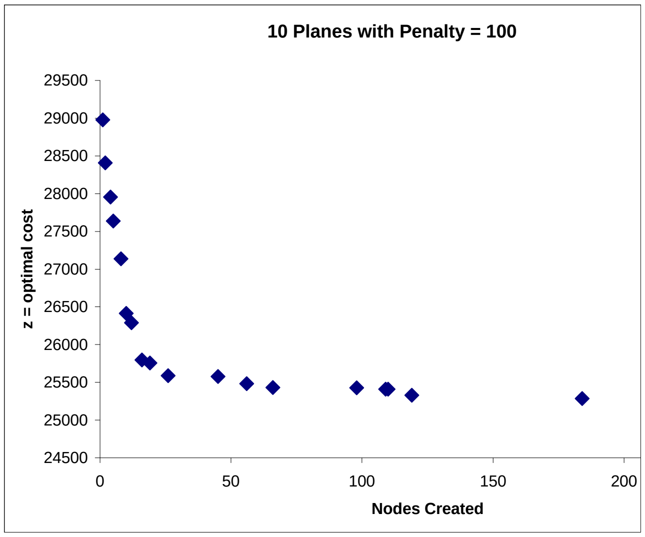

Figure 1: 10-Plane, Penalty = 100, Optimal Solution Cost

of decreasing disruption costs is faster than rate of increasing crew costs. It is useful to include information about disruption costs when evaluating crew schedules.

Furthermore, we are concerned with the final solution's sensitivity to the penalty parameter value. As previously stated, it is difficult to determine the exact value for the penalty parameter. Therefore, it is important that small changes in the parameter value do not lead to widely varying solutions. In our experiments we vary the penalty parameter by $\pm 20\%$. For example, we tested the 10-plane problem with penalty values of 80, 90, 100, 110, 120. In these experiments, our algorithm returns the same solution vector. Moreover, the algorithm generally finds the same sequence of solution vectors regardless of the penalty value. However, higher penalty values lead to more vectors in the sequence of improving solutions. We conclude the algorithm is robust in that similar penalty parameter values yield the same or similar solution vectors.

The penalty parameter value should be determined by the users as it is a measure of how much disruptions costs are valued with respect to crew costs. Some possible ways to determine the penalty value include:

- penalty parameter = number of passengers per flight;
- penalty parameter = number of crew members per flight;

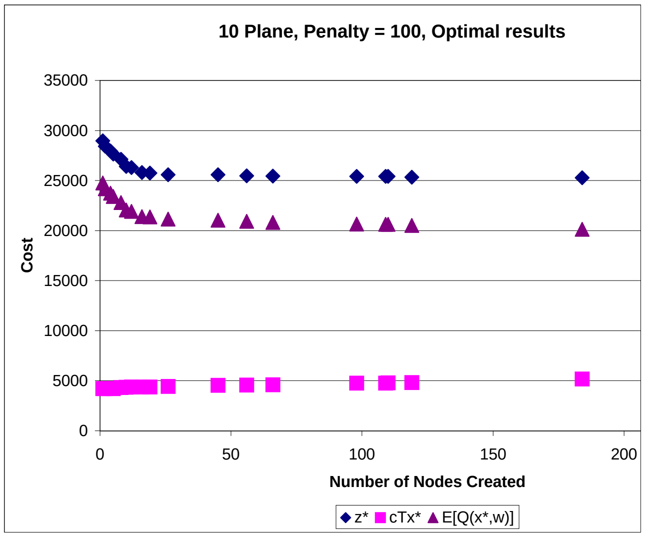

Figure 2: 10-Plane Problem with Penalty = 100: z, $\mathbf{c}^T\mathbf{x}$ and $\mathcal{Q}(\mathbf{x})$

- penalty parameter = number of passengers / number of crew;
- penalty parameter = price per minute of delay (dollar value);
- penalty parameter relative to arrival airport size.

Varying the penalty produces a range of solutions, as evidenced by the examples described above. The penalty choice may depend on other factors that a scheduler may consider. Having flexibility with the penalty parameter, and consequently the objective function, allows schedulers to evaluate different scenarios and find alternative solutions. The penalty parameter allows schedulers to include overall corporate goals regarding passenger disruptions into crew schedule.

5.3 Computational Results Conclusions

In general as the algorithm finds better solutions, these solutions contain pairings with fewer plane changes and longer connection times between plane changes. (See Table 5 for sample numerical results.) The algorithm adds buffer time in the solution and it become less sensitive to disruptions. Our results

i^{th} soln.	avg. cnx. time	plane changes
1	41.17	18
2	41.86	18
3	42.06	17
4	43.63	18
5	47.84	17
6	47.94	12
7	49.41	19
8	49.30	11
9	49.30	10
10	50.50	9
11	54.18	9
12	55.82	13
13	57.20	13
14	60.32	11
15	58.72	12
16	63.65	12
17	61.25	7
18	74.67	15
19	86.82	17

Table 5: Average Connection Time and Number of Plane Changes for 10-plane Problem, Penalty = 100

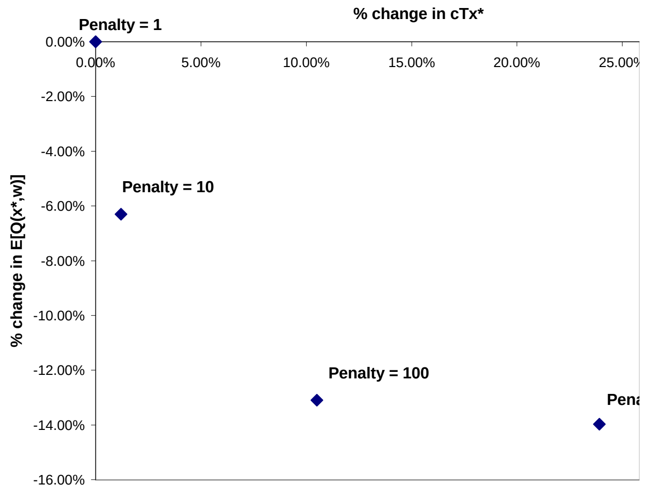

Figure 3: 9-Plane Problem Penalty Effects on Changes to $\mathbf{c}^T\mathbf{x}$ and $\mathcal{Q}(\mathbf{x})$

indicate reducing the number of plane changes and increasing the connection times when crews change planes will lead to less costly schedule disruptions.

6 Conclusion

We demonstrate the value of considering a stochastic formulation of the crew scheduling problem. Significant savings can be gained if delay effects on crew schedules, and consequently effects on the entire system, are considered during the planning phase.

Our Delay Branching Algorithm has promising results. Not only do we have a smaller branch and bound tree to explore as compared to the tree produced by traditional branch and bound methods, but we also show that our methodology results in significant savings in the expected cost of a crew schedule when disruptions are considered. It takes advantage of other developments in crew scheduling, such as improved pairing generation or more effective set covering algorithms, while also considering uncertainty. At any iteration of the algorithm we have already found a feasible solution which is more robust than the initial solution. Moreover, the algorithm is flexible and allows users to customize it to their needs and preferences.

7 Acknowledgements

This work was supported in part by the National Science Foundation under Grant DMI-9523275. The authors would like to thank Stephen Miller and Air New Zealand for their generosity in providing delay data.

References

ANBIL, R., E. GELMAN, B. PATTY, AND R. TANGA 1991. Recent Advances in Crew-Pairing Optimization at American Airlines. *Interfaces 21*, 62–74.

ANBIL, R., R. TANGA, AND E. JOHNSON 1992. A Global Approach to Crew-pairing Optimization. *IBM Systems Journal 31*, 71–78.

ANDERSSON, E., E. HOUSOS, N. KOHL, AND D. WEDELIN 1998. Crew Pairing Optimization. In *Operations Research in the Airline Industry.* Kluwer Academic Publishers.

ARABEYRE, J., J. FEARNLEY, F. C. STEIGER, AND W. TEATHER 1969. The Airline Crew Scheduling Problem: A Survey. *Transportation Science 3*, 140–163.

ARGÜELLO, M. F., J. F. BARD, AND G. YU 1998. Models and Methods for Managing Airline Irregular Operations. In *Operations Research in the Airline Industry.* Kluwer Academic Publishers.

BARNHART, C., E. L. JOHNSON, G. L. NEMHAUSER, M. W. P. SAVELSBERGH, AND P. H. VANCE 1998. Branch-and-Price: Column Generation for Solving Huge Integer Programs. *Operations Research 46*, 316–329.

BIRGE, J. AND F. LOUVEAUX 1997. *Introduction to Stochastic Programming.* Springer.

CAO, J.-M. AND A. KANAFANI 1997a. Real-Time Decision Support for Integration of Airline Flight Cancellations and Delays Part I: Mathematical Formulation. *Transportation Planning and Technology 20*, 183–199.

Cao, J.-M. and A. Kanafani 1997b. Real-Time Decision Support for Integration of Airline Flight Cancellations and Delays Part II: Algorithm and Computational Experiments. *Transportation Planning and Technology 20*, 201–217.

Chan, T. J. and C. A. Yano 1992. A Multiplier Adjustment Approach for the Set Partitioning Problem. *Operations Research 40*, S40–S47.

Chu, H. D., E. Gelman, and E. L. Johnson 1997. Solving Large Scale Crew Scheduling Problems. *European Journal of Operational Research 97*, 260–268.

Desrosiers, J., Y. Dumas, M. Desrochers, F. Soumis, B. Sanso, and P. Trudeau 1991. A Breakthrough in Airline Crew Scheduling. Technical Report G-91-11, Cahiers du GERAD.

Etschmaier, M. M. and D. F. X. Mathaisel 1985. Airline Scheduling: An Overview. *Transportation Science 19*, 127–138.

Fourer, R., D. M. Gay, and B. W. Kernighan 1993. *AMPL: A Modeling Language for Mathematical Programming*. The Scientific Press.

Gershkoff, I. 1989. Optimizing Flight Crew Schedules. *Interfaces 19*, 29–43.

Graves, G. W., R. D. McBride, I. Gershkoff, D. Anderson, and D. Mahidhara 1993. Flight Crew Scheduling. *Management Science 39*, 736–745.

Haneveld, W. K. K. and M. H. van der Vlerk 1999. Stochastic Integer Programming: General Models and Algorithms. *Annals of Operations Research 85*, 39–57.

Hoffman, K. L. and M. Padberg 1993. Solving Airline Crew Scheduling Problems by Branch-and-Cut. *Management Science 39*, 657–682.

ILOG, Inc. 1998. *CPLEX* (version 6.0 ed.). Incline Village, NV: ILOG, Inc.

Jarrah, A. Z., G. Yu, N. Krishnamurthy, and A. Rakshit 1993. A Decision Support Framework for Airline Flight Cancellations and Delays. *Transportation Science 27*, 266–280.

Johnson, E., T. Shaw, and R. Ho 1999. Modeling Tools for Airline Crew-Scheduling and Fleet-Assignment Problems. In *Operational Research in Industry*, pp. 1–24. MacMillan Press.

Kelly, T. P. 18 February 2000. Heap-based Priority Queue Implementation in ANSI C. URL: http://www-personal.engin.umich.edu/ tpkelly/heap/heap.tar.gz. Description: A simple and thoroughly tested general-purpose priority queue implemented with a binary heap in ANSI C.

Klabjan, D., E. L. Johnson, and G. L. Nemhauser 1999. Solving Large Airline Crew Scheduling Problems: Random Pairing Generation and Strong Branching. Technical report, School of Industrial and Systems Engineering, Georgia Institute of Technology.

Lettovsky, L., E. L. Johnson, and G. L. Nemhauser 1999. Airline Crew Recovery. *submitted to Transportation Science*.

Marsten, R. E., M. R. Muller, and C. L. Killion 1979. Crew Planning at Flying Tiger: A Successful Application of Integer Programming. *Management Science 25*, 1175–1183.

Mathaisel, D. F. X. 1996. Decision Support for Airline System Operations Control and Irregular Operations. *Computers and Operations Research 23*, 1083–1098.

McDowell, E. 1997. When Weather is the Enemy. *The New York Times*, D1, D20.

Nemhauser, G. L. and L. A. Wolsey 1999. *Integer and Combinatorial Optimziation.* John Wiley & Sons, Inc.

Rakshit, A., N. Krishnamurthy, and G. Yu 1996. System Operations Advisor: A Real Time Decision Support System for Managing Airline Operations at United Airlines. *Interfaces 26*, 50–58.

Rubin, J. 1973. A Technique for the Solution of Massive Set Covering Problems, with Applications to Airline Crew Scheduling. *Transportation Science 7*, 34–48.

Ryan, D. and B. Foster 1981. An Integer Programming Approach to Scheduling. In A. Wren (Ed.), *Computer Scheduling of Public Transport*, pp. 269–280. North-Holland Publishing Company.

Schaefer, A., E. Johnson, A. Kleywegt, and G. Nemhauser 2000. Robust Airline Crew Scheduling. Institute for Operations Research and Management Science Conference.

Schultz, R. 1996. Two-Stage Stochastic Integer Programming: a Survey. *Statistica Neorlandica 50*, 404–416.

Stojković, M., F. Soumis, and J. Desrosiers 1998. The Operational Airline Crew Scheduling Problem. *Transportation Science 32*, 232–245.

Stougie, L. and M. H. van der Vlerk 1997. Stochastic Integer Programming. In *Annotated Bibliographies in Combinatorial Optimization*, pp. 127–141. Wiley.

Teodorivić, D. 1985. A Model for Designing the Meteorologically Most Reliable Schedule. *European Journal of Operational Research 21*, 156–164.

Teodorivić, D. and S. Guberinic 1984. Optimal Dispatching Strategy on an Airline Network after a Schedule Perturbation. *European Journal of Operational Research 15*, 178–182.

Teodorivić, D. and G. Stojković 1995. Model to Reduce Airline Schedule Disturbances. *Journal of Transportation Engineering*, 324–331.

Wark, P., J. Holt, M. Rönnqvist, and D. Ryan 1997. Aircrew Schedule Generation Using Repeated Matching. *European Journal of Operational Research 102*, 21–35.

Wedelin, D. 1995. An Algorithm for Large Scale 0-1 Integer Programming with Application to Airline Crew Scheduling. *Annals of Operations Research 57*, 283–301.

Wei, G., G. Yu, and M. Song 1997. Optimization Model and Algorithm for Crew Management during airline irregular operations. *Journal of Combinatorial Optimization 1*, 305–321.

Yan, S. and C.-G. Lin 1997. Airline Scheduling for the Temporary Closure of Airports. *Transportation Science 31*, 72–82.