TRANSPORTATION SCIENCE
Vol. 40, No. 1, February 2006, pp. 3–14
ISSN 0041-1655 | EISSN 1526-5447 | 06 | 4001 | 0003

DOI 10.1287/trsc.1050.0138
© 2006 INFORMS

A Stochastic Programming Approach to the Airline Crew Scheduling Problem

Joyce W. Yen
College of Engineering, University of Washington, Seattle, Washington 98195-2180, joyceyen@u.washington.edu

John R. Birge
Graduate School of Business, University of Chicago, Chicago, Illinois 60637, john.birge@chicagogsb.edu

Traditional methods model the billion-dollar airline crew scheduling problem as deterministic and do not explicitly include information on potential disruptions. Instead of modeling 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 are 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.

Key words: airline operations; crew scheduling; stochastic programming
History: Received: November 2003; revision received: December 2004; accepted: January 2005.

Introduction

To increase profits, airlines continually look for ways to better use their resources and to improve scheduling decisions. Billion-dollar crew costs are second only to fuel costs among all airline costs; thus, airlines seek to efficiently use their crew resources (Graves et al. 1993; Anbil, Tanga, and Johnson 1992). Disruptions can lead to increased crew costs and additional scheduling inefficiencies.

Day-to-day operations consist of two components of airline crew scheduling: short-range planning and long-range planning, traditionally seen as two separate problems. Short-range planning yields crew assignments under short-term time constraints, generally done at the operational level. (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 far in advance and select a set of optimal crew itineraries to be later assigned to specific crews. (See Arabeyre et al. 1969; Etschmaier and Mathaisel 1985; Gershkoff 1989 for general overviews.)

This paper departs from traditional methods by considering the effect of random disruptions explicitly in the construction of initial schedules. Because long-range planning decisions are inevitably altered by short-range decisions responding to schedule disruptions and changes, 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 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 few iterations of our algorithm. The solutions to our model are more robust than those of traditional methods because we reduce the perpetuation of delays via scheduling decisions. These results suggest that important savings can be obtained by considering a stochastic model for crew scheduling.

Our approach is practical for airline crew scheduling because it requires a modest number of iterations of existing crew scheduling methods to produce improved solutions. Because crew scheduling decisions occur offline over an extended period before their required implementation, the computation times for the procedures here remain practical. These results provide an example for other large integer optimization planning or design problems that do not currently include explicit consideration of uncertain parameters.

We first present some background on the deterministic crew scheduling problem in §1. In §2 we present our stochastic formulation. Section 3 outlines our branching algorithm. Our computational results appear in §4. Finally, we make concluding remarks in §5.

1. Background

The general long-range crew scheduling problem (CSP) is modeled as a deterministic integer program whose objective is to find a minimum cost assignment of flights to itineraries. An itinerary is a round-trip sequence of flights. A cockpit and cabin crew will later be assigned to each itinerary. Typically, the problem is modeled 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 CSP can be formulated as follows: Suppose we have N feasible pairings and m different flight segments to cover. We have the following set partitioning problem:

$$\text{minimize } \sum_{n=1}^{N} c_n x_n \tag{1}$$

$$\text{subject to } \sum_{n=1}^{N} a_{in} x_n = 1, \quad \forall i = 1, \ldots, m;$$

$$x_n \in \{0, 1\}, \quad \forall n = 1, \ldots, N; \tag{2}$$

where

$x_n = 1$ if pairing n is selected in the solution, 0 otherwise,

$a_{in} = 1$ if flight segment i is covered by pairing n, and 0 otherwise,

$c_n =$ 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, Tanga, and Johnson 1992; Andersson et al. 1998; Barnhart et al. 1998; Chan and Yano 1992; Chu, Gelman, and Johnson 1997; Desrosiers et al. 1991; Gershkoff 1989; Graves et al. 1993; Hoffman and Padberg 1993; Klabjan, Johnson, and Nemhauser 2001; Marsten, Muller, and Killion 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 crew recovery planning process is activated either (1) immediately after a 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, Bard, and Yu (1998); Cao and Kanafani (1997a, b); Jarrah et al. (1993); Lettovsky, Johnson, and Nemhauser (2000); Mathaisel (1996); Rakshit, Krishnamurthy, and Yu (1996); Stojković, Soumis, and Desrosiers (1998); Teodorivić (1985); Teodorivić and Guberinic (1984); Wei, Yu, and Song (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, which 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. 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. This expected cost is calculated using a Monte Carlo simulation called SimAir. This simulation evaluates a crew schedule under actual operations. They then solve the standard crew scheduling problem (given by (1) and (2)) with this revised cost coefficient vector.

Our model can be considered an extension to the analysis in Schaefer et al. We can include these expected costs as depending on single crew assignments, but we focus on disruption interactions between potential crew schedules, that is, those disruptions that are perpetuated to other planes and other crews due to crew scheduling decisions. We propose a stochastic programming model for this problem.

Our model focuses on identifying disruptions perpetuated by assignment decisions and incorporates the cost of such disruptions into an optimization procedure. Our goal is to identify disruption costs that occur not just because of a single crew's delays, but also because of the interactions among crew pairings. For our examples, we approximate the actual costs by allowing flights to accrue delay but operate as planned. (In practice, more detailed recovery operations and their costs could be considered.) For our examples, we also do not accumulate additional delays due to crew regulations, although again such detailed analyses, as in Schaefer et al. (2000), may be used. For the Air New Zealand domestic network that we consider, we believe our results represent a good approximation of reality due to that network's limited daily flight times, single time zone, and short flight legs. Generally, our results provide some basic intuition regarding disruptions effects and offer motivation for stochastic crew scheduling and other stochastic integer optimization models.

We represent this problem as a two-stage stochastic integer program with recourse, where our proposed recourse model reflects the interaction between long-range planning decisions and short-range operational results. This model includes interaction across flights, crew resources, and plane resources. By capturing such interactions, the model is able to identify robust solutions that can better withstand schedule disruptions than the original schedules can.

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

2. Problem Formulation

Because plane usage and crew assignment turnaround times are often short, long delays can have a cascading effect on future plane and crew assignments. As a result, crew schedules that 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). Because the deterministic crew scheduling model fails to address this problem, we turn to stochastic models.

2.1. Stochastic Programming Formulation

Our method adjusts assignment decisions 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 expected delay costs to the deterministic objective function and observe which flight pairs contribute to the delay.

We suppose that we 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, $\boldsymbol{\xi}(\omega)$, 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.

We then reformulate the problem as:

$$\begin{aligned} \text{minimize} \quad & \mathbf{c}^T\mathbf{x} + \mathcal{Q}(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} = \mathbf{b}, \\ & \mathbf{x} \in \{0, 1\}. \end{aligned} \tag{3}$$

In this formulation the vector $\mathbf{c}$ represents the expected cost of flying a given pairing independent of all other pairings. (The method in Schaefer et al. 2000 may be used to calculate these values.) The constraints $\mathbf{A}\mathbf{x} = \mathbf{b}$ consist of the coverage equation (2) and other crew constraints. The recourse function, $\mathcal{Q}(\mathbf{x}) = \sum_\omega Q(\mathbf{x}, \omega)P(d\omega)$, is the expected value of future actions due to disruptions in the original schedule that cause a delay to a crew other than the crew for an original delay. This formulation is a standard two-stage stochastic (integer) program with recourse (Birge and Louveaux 1997).

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

The key contribution of our formulation is our inclusion of the relationship between pairings. We propose that crew schedules (pairings) should not be considered in isolation. Effects to one pairing can impact other crews and planes because these resources are connected through scheduling decisions. For example, a crew that is scheduled to switch planes and arrives late may transfer this delay to their new flight, thus impacting a new plane. We propose that there is a cost associated with these interactions between crews, planes, and pairings. Furthermore, it is impossible to determine the effects of these connected relationships without considering the crew schedule as a whole. Hence, we propose evaluating an entire schedule assignment so as to capture the interconnected resource effects. This evaluation will be done in the recourse model of our stochastic program.

In our formulation, $Q(\mathbf{x}, \omega)$ represents the cost of delays due to crew assignment decisions. The first-stage variables x are the pairing decision variables. Once these have been determined, we assume some mechanism to measure additional costs due to actual arrival and departure times. In our version of this problem, we only consider delay costs from following the original schedule. Our model is quite general and can include other forms of recourse such as rerouting due to legalities and other specific flight network information. However, for our example (Air New Zealand), overtime crew costs within a day, which are represented by delay costs, were sufficient. Other more complicated forms of recovery from disruptions could be included, but would not change the essential features of the method we propose. Even with only these relatively minor costs, the consideration of crew assignment interactions with random interruptions can lead to substantial saving relative to assignment methods that ignore interactions and stochastic delays.

In the formulation, second-stage decision variables, particularly actual arrival and departure times for each flight, depend on the first-stage decisions. For the basic recourse problem without crew or plane reassignment, the formulation (NLR) with a disruption scenario ω is

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

subject to

$$r^a_{j\omega}-d^a_{j\omega}\ge t_{j\omega}, \quad \forall j; \tag{4b}$$

$$r^a_{j\omega}-\Delta_{j\omega}\le r^s_j, \quad \forall j; \tag{4c}$$

$$d^a_{j\omega}-r^a_{p^p_j\omega}\ge g^p_{j\omega}, \quad \forall j; \tag{4d}$$

$$d^a_{j\omega}-\left[r^a_{p^c_{jn}\omega}+g^c_{jn}\right]x_n\ge 0, \quad \forall j\in \text{pairing } n,\ n=1,\ldots,N; \tag{4e}$$

$$(d^a_{j\omega})'-(r^a_{p^p_j\omega})'=g^p_{j\omega}, \quad \forall j; \tag{4f}$$

$$(r^a_{j\omega})'=r^s_j+\Delta'_{j\omega}, \quad \forall j; \tag{4g}$$

$$(r^a_{j\omega})'=(d^a_{j\omega})'+t_{j\omega}, \quad \forall j; \tag{4h}$$

$$\Delta_{j\omega}\ge 0, \quad \forall j; \tag{4i}$$

$$\Delta'_{j\omega}\ge 0, \quad \forall j; \tag{4j}$$

where j refers to flight segments and $t_{j\omega}$ is flight j's flight time under scenario ω. The prime notation indicates a variable associated exclusively with late-arriving aircraft equipment. Hence, $d^a_{j\omega}$ ($r^a_{j\omega}$) is *actual* departure (arrival) time for flight j under scenario ω, while $(d^a_{j\omega})'$ ($(r^a_{j\omega})'$) is actual departure (arrival) time for flight j under scenario ω when only connecting plane equipment constraints are considered. Also, $\Delta_{j\omega}$ is the total arrival delay to flight j, including plane-induced delays and crew-induced delays, while $\Delta'_{j\omega}$ is the arrival delay without considering crew interactions with other crews and planes. Therefore, $\Delta_{j\omega}-\Delta'_{j\omega}$ is the arrival delay to flight j due to crew assignment interactions. We have d^s_j (r^s_j) represent the *scheduled* departure (arrival) time, p^p_j is plane predecessor flight for flight j, p^c_{jn} is crew predecessor flight of flight j under pairing n, $g^p_{j\omega}$ (g^c_{jn}) is plane ground time for flight j under scenario ω (crew ground time under pairing n), and ρ_j is the penalty cost for delaying flight j. Note that p^c_{jn} and p^p_j are just flight indices. Finally, $\mathbf{d}^a$ and $\mathbf{r}^a$ are vectors of actual departure and arrival times, respectively.

In our implementation, crew assignment interactions occur only when crews switch planes. In other cases, crew duty-period legalities (such as minimum rest times) may cause interactions by creating additional plane delays without switching planes. These cases were not prevalent in the network we considered, but could be included for other implementations, such as networks with longer flight times, when they become critical.

Thus, in our implementation, the objective (Equation (4a)) represents the cost of delays due to crews switching planes. The first two inequalities, (4b) and (4c), represent consistency constraints for flight arrival and departure times. Inequality (4d) represents plane precedence constraints, and inequality (4e) represents crew precedence constraints. Equations (4f)–(4h) address delay due to late aircraft equipment. Finally, Equations (4i) and (4j) indicate nonnegative delay times.

The random vector is $\boldsymbol{\xi}(\omega)=(t_{j\omega}, j=1,\ldots,; g^p_{j\omega}, j=1,\ldots)$. Our decision variables for the recourse problem NLR are $d^a_{j\omega}$ and $r^a_{j\omega}$. Note that the decision variables for the full stochastic program (3) are $d^a_{j\omega}$, $r^a_{j\omega}$, and x_n.

Equation (4e) allows us to evaluate the schedule as a whole by considering effects of crews changing planes. To measure these effects, we introduce a nonlinear relationship between actual flight arrival and departure times and crew-scheduling decisions. Note, if x_n 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

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

Note that the second-stage variables (y) measure how infeasible the first-stage decisions make the scheduled departure and arrival times. With this observation, the recourse problem (RP) can be solved quickly by following the flight schedule in chronological sequence.

We next describe our *flight-pair branching algorithm* for solving this stochastic integer programming problem.

3. Flight-Pair Branching Algorithm

As mentioned, this model provides an estimate of the cost of implementing crew scheduling decisions under schedule disruptions. For our examples, we do not include crew labor restrictions, flight cancellations, standby crews, or other events that may occur after a disruption. In general, we could model any form of recovery from disruption. The key aspect of this approach is to identify the delays that occur from crew assignment interactions.

For our examples, crew assignment interactions introduce delays only when crews are assigned to switch planes; thus, crews scheduled to follow plane assignments do not introduce new delay to the system. We therefore wish to identify solutions minimizing crew plane changes and capture this concept through a specific type of delay, *switch_delay*.

These *switch_delay*s determine the branching order in our flight-pair branching algorithm. Because *switch_delay* only occurs when two flights on different planes are included in a single pairing, our branching either includes or excludes all pairings with that pair of flights. Other causes of crew assignment-interaction delay, such as labor restrictions, would also involve some pair of flights (possibly on the same plane) that would form part of the branching. In any case, we would seek to eliminate those flight pairs causing the most costly disruptions to the schedule. For our examples, we only need to compute *switch_delay*, as defined in the following section, to order the disruption costs.

3.1. 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 the plane assigned to flight i next flies flight k, and flight i is delayed, then flight k may be delayed as well. Thus if $j \neq k$, then a delay of flight i may delay both flights j and k. We call such delays to flight j *switch_delay*. It is a delay due to a crew scheduling decision and may lead to other delays for flights with the same plane as j. Let the set of pairings that induce a *switch_delay* for flights i and j be S_{ij}.

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

$$\mathit{switch\_delay}(x_n, i, j) = \sum_{\omega\in\Omega} p_\omega\big[\big(\Delta_{j\omega} - \big(r^a_{p^p_j\omega} + \mathit{gnd\_time} - d^s_j\big)\big)\delta_{ijn}\big],$$

where

$$\delta_{ijn} = \begin{cases} 1 & \text{if } n \in S_{ij} \\ 0 & \text{otherwise.} \end{cases}$$

and S_{ij} is the set of all pairings n satisfying the following inequalities:

$$\Delta_{j\omega} > 0, \quad x_n = 1, \quad a_{in} = 1, \quad a_{jn} = 1, \quad \Delta_{i\omega} > 0,$$
$$p^c_{jn} = i, \quad \mathit{plane}(j) \neq \mathit{plane}(i), \quad d^s_j < r^a_{i\omega}.$$

In the above definition, δ_{ijn} is an indicator variable denoting which pairings allow flight i to introduce a delay to flight j due to crews changing planes. Also, *gnd_time* is the required time on ground (equal to the maximum of the crew ground-time constraint and the plane ground-time constraint). For this work, we presume *gnd_time* is independent of the flights and is constant. This assumption can be removed later. Finally, a_{jn} is the value in the the jth row and nth column of constraint matrix **A**, given by (2).

3.2. Algorithm Description

First we describe the algorithm in words, and then follow with a formal description. 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. We either allow or disallow key flight pairs where crews switch planes. 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

Table 1 **Notation Used in Flight-Pair Branching Algorithm**

Symbol	Definition
C	set of nodes that are available for branching/evaluation
c	node index
$\mathcal{D}^l$	set of flight pairs (i, j) with a positive *switch_delay* at current node l
E_{ij}	binary vector indicating which pairings includes both flights i and j
$\mathbf{E}^K$	matrix whose rows are vectors E_{ij}, where i and j are determined in subproblem K
i, j	flight indices
(i, j)	flight pair with flight i followed by flight j
(i_l, j_l)	flight pair that accrues the maximum *switch_delay* at node l
K	subproblem index, also indexes the number of nodes already evaluated
l	current node
l'	node index
L	set of all nodes in the tree
n	pairing index
N	total number of pairings
$\mathcal{N}^l$	$\mathcal{D}^l \cap \bar{\mathcal{P}}^l \cap \bar{\mathcal{R}}^l$
$\mathcal{P}^l$	set of excluded flight pairs at node l
$\mathcal{R}^l$	set of explicitly included flight pairs at node l
$(\mathbf{x}^l)^*$	optimal pairing vector at node l
$(x^l)^*_n$	nth element (pairing) of the vector $(\mathbf{x}^l)^*$
UB	upper bound value
$(w^l)^*$	optimal deterministic objective value at node l
z^l	overall stochastic program objective value at node l

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, hence evaluating a number of pairings at the same time. Along one branch, we allow the identified flight pair to appear 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. We eliminate subsets of pairings containing the expensive pair.

We call our method *flight-pair branching,* because we partition the set of all crew pairings by included flight pairs. In traditional binary integer programming branch and bound, branching occurs on the value of a single variable, in our case, a crew pairing. We call this traditional approach, *crew-pairing branching*. Comparing these two approaches, if we have N pairings and F flights, then crew-pairing branching could include 2^N nodes, while flight-pair branching includes at most $2^{F(F-1)}$ nodes (and, in fact, many fewer because not all flights can be coupled with all other flights). In general, $N \gg F(F-1)$ (as in the example given in §4.2), so that flight-pair branching produces smaller branching trees than crew-pairing branching.

With the exception of solving the set partitioning problem (SPP) given in (2), 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. A user may choose, however, to terminate the algorithm whenever these bounds are sufficiently close or the current solution's disruption cost is satisfactorily limited. Furthermore, the algorithm produces feasible and progressively robust solutions on each iteration as different flight-pairs are eliminated from the set of pairings.

Now we are ready to give the formal description of the *flight-pair branching algorithm*.

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

$$\begin{aligned}(w^0)^* = \text{minimize}\quad & \mathbf{c}^T\mathbf{x} \\ \text{subject to}\quad & \mathbf{A}\mathbf{x} = \mathbf{b}; \\ & 0 \le \mathbf{x} \le 1, \quad \mathbf{x} \text{ integer};\end{aligned} \tag{5}$$

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 upper bound

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

Identify flight pairs (i, j) and pairing(s) n such that $switch\_delay((x^l)^*_n, i, j) > 0$ and set $\mathcal{D}^l = \{(i, j) \mid switch\_delay((x^l)^*_n, i, j) > 0\}$. If $\mathcal{D}^l \neq \{\emptyset\}$, go to Step 3. Otherwise, let $C = C \backslash \{l\}$ and go to Step 2.

Step 2. If $C = \emptyset$, terminate with optimal solution vector $\mathbf{x}^*$ such that $z(\mathbf{x}^*) = UB$. For an ϵ-optimal solution, terminate if $UB - \min_{c\in C}\{(w^c)^*\} < \epsilon$ because $\min_{c\in C}\{(w^c)^*\}$ is a lower bound on the optimal solution. Otherwise, find a node $l' \in \arg\min_{c\in C}\{(w^c)^*\}$. Set $l = l'$ and go to Step 1.

Step 3. Find

$$(i_l, j_l) \in \arg\max_{(i,j)\in\mathcal{N}^l} \sum_{n=1}^{N} switch\_delay((x^l)^*_n, i, j).$$

Branch from node l. Let

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

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

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

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 an optimal solution to

$$\begin{aligned}(w^{K+1})^* = \text{minimize}\quad & \mathbf{c}^T\mathbf{x} \\ \text{subject to}\quad & \text{subproblem } (K+1) \\ & \mathbf{A}\mathbf{x} = \mathbf{b}; \\ & \mathbf{E}^T_{ij}\mathbf{x} = 0, \quad \forall (i, j) \in \mathcal{P}^{K+1}; \\ & \mathbf{E}^T_{ij}\mathbf{x} \ge 1, \quad \forall (i, j) \in \mathcal{R}^{K+1}; \\ & \mathbf{x} \in \{0, 1\};\end{aligned}$$

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

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

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

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

$$\begin{aligned}(w^{K+2})^* = \text{minimize} \quad & \mathbf{c}^T\mathbf{x}\\ \text{subject to} \quad & \text{subproblem } (K+2)\\ & \mathbf{A}\mathbf{x} = \mathbf{b};\\ & \mathbf{E}_{ij}^T\mathbf{x} = 0, \quad \forall (i,j) \in \mathcal{P}^{K+2};\\ & \mathbf{E}_{ij}^T\mathbf{x} \geq 1, \quad \forall (i,j) \in \mathcal{R}^{K+2};\\ & \mathbf{x} \in \{0,1\};\end{aligned}$$

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

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

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

$$w^{l'} = \min_{c \in C}\{(w^c)^*\}.$$

Set $l = l'$ and go to Step 1.

3.3. Proof of Convergence to Optimality

We show that 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 an optimal solution.

LEMMA 1. *The flight-pair branching algorithm terminates in a finite number of iterations, and upon termination there are no more branches to explore.*

PROOF. Because 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

$$\mathbf{E}_{ij}^T\mathbf{x} = 0, \quad \text{or} \quad \mathbf{E}_{ij}^T\mathbf{x} \geq 1.$$

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. Because 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, $C = \{\varnothing\}$, implying no more nodes to explore in the algorithm.

Consider some node K. Either K has a pair $(i_K, j_K) \in \mathcal{N}^K$ such that

$$\sum_{n=1}^{N} switch\_delay\big((x^K)_n^*, i_K, j_K\big) > 0$$

or else no such pair exists at node K. If no such pair exists, we are done; otherwise, 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; under Case (b), we identify at node K a branching pairing $(\hat{i}, \hat{j})$ that 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.

We have shown that either there does not exists a pair (i_K, j_K) such that

$$\sum_{n=1}^{N} switch\_delay((x^K)_n^*, i, j) > 0$$

or $(w^K)^* > UB$ for all nodes K in the tree; hence, when the algorithm terminates, no more branches remain to be explored. □

THEOREM 2. *The delay branching algorithm terminates with an optimal solution.*

PROOF. Suppose the algorithm terminates without finding an optimal solution. 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

$$|\mathcal{R}^M \cap \bar{I}^*| + |\mathcal{P}^M \cap I^*| = 0.$$

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) $\mathcal{D}^M = \{\varnothing\}$ or (b) $(w^M)^* > UB$.

Case a. $\mathcal{D}^M = \{\varnothing\}$. 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 $\hat{\mathbf{x}}$'s being optimal.

Case b. $(w^M)^* > UB$. As observed earlier, $\mathbf{c}^T\hat{\mathbf{x}} \geq \mathbf{c}^T(\mathbf{x}^M)^* = (w^M)^*$; thus

$$\mathbf{c}^T\hat{\mathbf{x}} + \mathcal{Q}(\hat{\mathbf{x}}) \geq \mathbf{c}^T\hat{\mathbf{x}} \geq (w^M)^* > UB,$$

which contradicts 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 the recourse problem* NLR.

PROOF. The corollary follows directly from Lemma 1 and Theorem 2. □

4. Computational Results

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

4.1. 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, Shaw, and Ho (1999).
- All crews fly their schedules as planned regardless of the delay circumstances. This assumption may not hold because of either crew work restrictions or other recovery actions, but it provides an estimate that approximates the results in the network we considered.
- We solve a static daily problem; that is, the same schedule is flown every day. We presume plane assignments are fixed with the same flight-plane assignments each 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 the set of possible crew pairings and generate the random disruption scenarios ω. For each test problem we sampled 100 scenarios using a truncated gamma or lognormal distribution for the length of delays to match disruption data from Air New Zealand for mean, second moment, and range. We assume that each disruption scenario is equally likely.

4.2. 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, serving 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, Shaw, and Ho (1999), we enumerate 1,911 feasible pairings for the 9-plane problem, 2,737 pairings for the 10-plane problem, and 3,296 for the 11-plane problem. We implement the algorithm using AMPL (1993) to formulate the model and ILOG (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.

Because the branching tree becomes very large, 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 the flight-pair branching tree is large, it is naturally much smaller than the crew-pairing branching tree. Consider, for example, the nine-plane problem with 1,911 feasible pairings. In this problem, 20 flights are departing or arriving at Auckland, 18 at Christchurch, 3 at Dunedin, and 20 at Wellington. Due to location restrictions, the maximum number of flight pairs is at most ${}_{20}C_2 + {}_{18}C_2 + {}_{3}C_2 + {}_{20}C_2 = 536$. This value is an upper bound 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, the flight-pair branching tree produces at most $2^{537} - 1$ nodes, while crew-pairing branch and bound produces $2^{1,912} - 1$ nodes. While both numbers are ridiculously large, we observe that generally our process quickly identifies good candidate solutions.

To calculate the objective function value, we must choose a value for the penalty parameter. The cost of a delay depends on a number of issues that are difficult to measure. The cost of losing a passenger, for example, can vary. A passenger may be minimally affected by the delay, with little or no cost to the airline; may be lost for a single flight for the cost of one ticket's lost revenue; or may choose never to fly that airline again, resulting in significant losses of ongoing revenue. Because in practice we may not know the exact cost of delays, we examine a set of possible delay penalty values.

We test four variations of each test problem, examining how different penalty parameter values affect the solution. We show that 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), which measures the expected gain from using a solution from a stochastic programming model over a solution from a deterministic model (Birge 1982). Calculating the VSS further demonstrates the merits of considering the stochastic model.

To describe how to calculate VSS, we must first describe two problem formulations. First, we have the value of the optimal solution to the recourse problem, given by (3) and NLR, $z(\mathbf{x}^*)$. Second, although departure and arrival times are random, we presume that the scheduled departure and arrival times represent the expected departure and arrival times. Therefore, we can consider the solution to (5), $\mathbf{x}^0$, the solution to the expected value problem. We evaluated the expected results of using x^0 and compare the two

Table 2 Test Problem Results

Test problem	Penalty	$\mathbf{c}^T\mathbf{x}^0$	$\mathcal{Q}(\mathbf{x}^0)$	$\mathbf{c}^T\mathbf{x}^*$	$\mathcal{Q}(\mathbf{x}^*)$	VSS	$z(\mathbf{x}^*)/z(\mathbf{x}^0)$
9-plane problem	1	3,889	208.803	3,889	208.803	0	1.00
	10	3,889	2,088.035	3,937	1,957.513	82.5218	0.98619
	100	3,889	20,880.35	4,297	18,139.53	2,332.81	0.90582
	1,000	3,889	208,803.5	4,819	179,628	28,245.5	0.8672
10-plane problem	1	4,219	247.5961	4,220	234.1843	12.4118	0.99722
	10	4,219	2,475.96	4,392	2,136.543	166.418	0.97514
	100	4,219	24,759.61	5,498	19,664.14	3,816.47	0.8683
	1,000	4,219	247,596.1	5,498	196,641.4	49,675.7	0.8027
11-plane problem	1	5,058	175.87	5,059	174.55	0.322	0.99994
	10	5,058	1,758.76	5,175	1,627.15	14.62	0.99079
	100	5,058	17,587.62	5,408	15,563.02	1,674.59	0.9260
	1,000	5,058	175,876.2	5,408	155,630.2	19,895.9	0.8900

results. This difference is precisely the VSS, VSS $= z(\mathbf{x}^0) - z(\mathbf{x}^*)$.

4.2.1. Comparing Deterministic Solutions and Stochastic Solutions. From the penalty parameter analysis (discussed in more detail in §4.2.2), 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 1,000. 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 that 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 nine-plane problem with a penalty parameter of 10 is 82.5218 units less than the initial solution and is approximately 98.619% of the initial solution. We see a 10%–20% reduction in recourse delay costs between the x^0 and x^* solutions. Depending on the cost of a delay, such a reduction could result in significant savings.

The values in Table 2 also give some indication of the consistency of the model with the 1997 data from Air New Zealand. The delay figures $\mathcal{Q}(\mathbf{x}^0)$ approximately match the actual Air New Zealand delays due to crew assignments. For example, in 1997, the total delay minutes per day averaged 392.65 with approximately half caused by plane and crew connections, closely matching our model's average delays of 176 to 248 minutes. The results indicate that our delay model (of following the original schedule) closely captures the actual disruption recovery effects in this small network.

While we do not explore the entire branching tree, we do explore enough of the tree that we believe our solutions are near optimal or optimal, because the curve of new optimal solution values flattens (see Figures 1 and 2). However, we do not have proof of optimality for these results. Figure 1 illustrates the objective function value progression of improving solutions for the 10-plane problem. This information is repeated in Figure 2 (diamonds, in the figure) in conjunction with the recourse function value (triangles) and the lower bound (squares). Figure 2 shows both the upper and lower bounds (indicated by the diamonds and squares, respectively) on the optimal solution value. 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 5% of the best solution after only five iterations, within 2% after 13 iterations, and within 1% 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

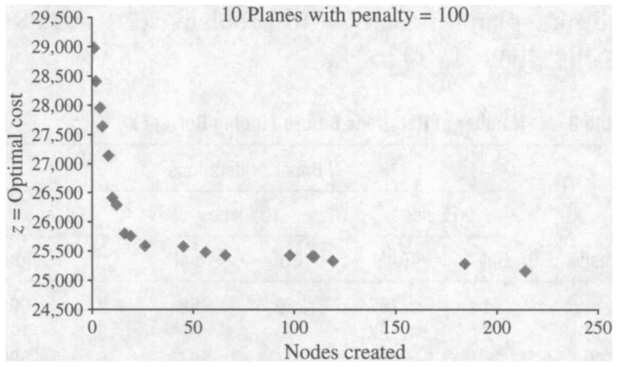

Figure 1 10-Plane, Penalty = 100, Optimal Solution Cost

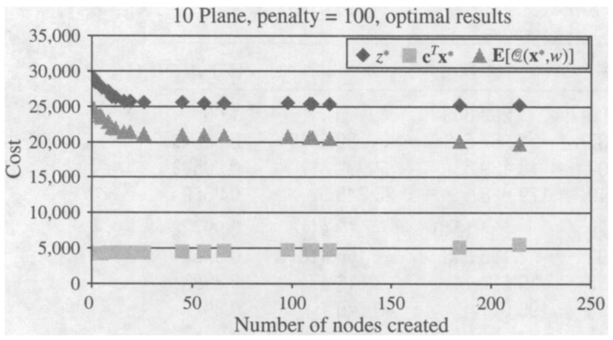

Figure 2 **10-Plane Problem with Penalty = 100: z, $\mathbf{c}^T\mathbf{x}$, and $\mathcal{Q}(\mathbf{x})$**

to $\mathbf{c}^T\mathbf{x}$ 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.

4.2.2. Penalty Parameter Analysis. An important part of the modeling process is determining the trade-off between known crew costs and uncertain future delay costs. The value of the penalty parameter controls this trade-off. When the penalty parameter equals 1, 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 of order 10^3, while the recourse objective is of order 10^2.

We examine the effects of different penalty parameter values. As we increase the penalty parameter, we track the trade-off between known crew costs and disruption costs. Quantifying this trade-off can be useful both for determining crew assignments and for schedule generation to compare schedules with different levels of slack time.

As the penalty parameter increases, 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 nine-plane, penalty = 10 problem $z(\mathbf{x}^*)$ is 1.38% smaller than $z(\mathbf{x}^0)$.)

Table 3 **Number of Iterations Before Finding Current x***

	Number of iterations					
	9 Planes		10 Planes		11 Planes	
Penalty	Opt	Total	Opt	Total	Opt	Total
1	1	1,287	289	468	2	998
10	7	1,127	10	162	13	197
100	25	995	107	184	76	580
1,000	140	386	107	256	76	173

From these results we conclude that, beyond some threshold for the penalty parameter, no additional delay savings result from a higher penalty. For example, in the 10-plane and 11-plane results we find the same solution is returned for a penalty value of 100 and 1,000. These results also suggest that, at some point, delays can no longer be eliminated regardless of the penalty value. One additional conclusion is there exists a region where the rate of decreasing disruption costs is faster than the rate of increasing crew costs.

We are, furthermore, 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, we would prefer that small changes in the parameter value do not lead to widely varying solutions. In our experiments we vary the penalty parameter by ±20%. For example, we tested the 10-plane problem with penalty values of 80, 90, 100, 110, and 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. Higher penalty values, however, lead to more vectors in the sequence of improving solutions. We conclude the algorithm is robust in these cases with similar penalty parameter values yielding the same or similar solution vectors.

The penalty parameter value should be determined by the users to measure how disruptions costs are valued relative to crew costs. Some possible inputs to determine the penalty value include:

- penalty parameter = number of passengers per flight;
- penalty parameter = number of crew members per flight;
- 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 shown in the examples 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 also allows schedulers to include overall corporate goals regarding passenger disruptions into crew schedule.

4.3. Computational Results Conclusions

In general, as the flight-pair branching algorithm finds better solutions, these solutions contain pairings with fewer plane changes that result in fewer overall crew plane changes. Examples of these reductions

Table 4 Effects of the Penalty Parameter

	Percent change in value from initial solution								
	9 Planes			10 Planes			11 Planes		
Penalty	z^*	$\mathbf{c}^T\mathbf{x}^*$	$\mathcal{Q}(\mathbf{x}^*)$	z^*	$\mathbf{c}^T\mathbf{x}^*$	$\mathcal{Q}(\mathbf{x}^*)$	z^*	$\mathbf{c}^T\mathbf{x}^*$	$\mathcal{Q}(\mathbf{x}^*)$
1	0	0	0	−0.28	0.023	−5.42	−0.006	0.02	−0.75
10	−1.38	1.23	−6.25	−2.49	4.10	−13.71	−0.21	2.31	−7.5
100	−9.42	10.49	−13.13	−13.17	30.32	−20.58	−7.40	6.92	−11.51
1,000	−13.28	23.91	−13.97	−19.73	30.32	−20.58	−10.99	6.92	−11.51

appear in Table 5. These results also indicate that the algorithm adds buffer time in the initial schedule, making it less sensitive to disruptions. Connection times are a major factor for crews changing planes. If connection times are too tight and delays occur, crews may not be able to make their connections. Longer connection times provide greater flexibility when faced with delay. Hence, we observe increasing average connection times as the algorithms identify improving solutions. We also see generally higher percentages of crew connections following planes as better solutions are identified. A user could refine the algorithm by including this kind of information.

Our results, not surprisingly, indicate that reducing the number of plane changes and increasing the connection times when crews change planes leads to less-costly schedule disruptions. In contrast to ad hoc procedures that might inflate the cost of pairings with shorter connection times or plane changes, the method here provides a procedure for considering the actual costs of these pairing features.

5. 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.

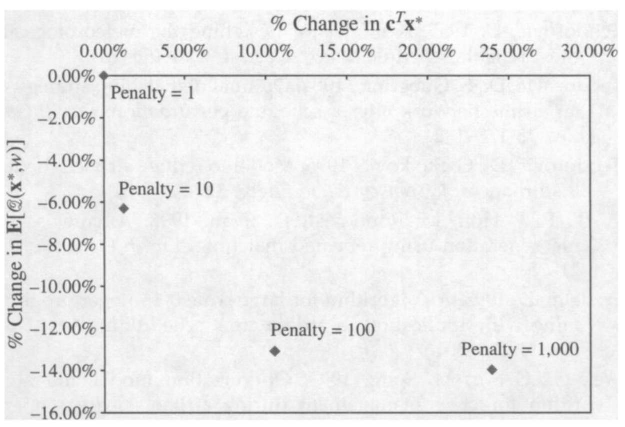

Figure 3 Nine-Plane Problem Penalty Effects on Changes to $\mathbf{c}^T\mathbf{x}$ and $\mathcal{Q}(\mathbf{x})$

Our flight-pair branching algorithm has promising results. It takes advantage of other developments in crew scheduling, such as improved pairing generation and more effective set partitioning and/or set covering algorithms, while also considering uncertainty. At any iteration of the algorithm, we have a feasible solution that is more robust than the initial solution; moreover, the algorithm is flexible and allows users to customize it to their needs and preferences, resulting in overall savings in the expected cost of a crew schedule when disruptions are considered.

Beyond implications for crew scheduling, our results demonstrate that efficient branching techniques based on specific problem structure can yield solutions to practical stochastic integer programs that improve significantly upon the results of deterministic models that ignore uncertainty or replace random parameters with their expected values. In our examples, improved solutions resulted from relatively few iterations of the algorithm despite the potential of extremely large branching trees. We believe that similar forms of branching based on key stochastic program cost generators (such as flight delays in our case) may yield similar results in a wide variety of integer

Table 5 Average Connection Time and Percentage of Crew Connections Following Planes for 10-Plane Problem, Penalty = 100

ith best soln.	Avg. cnx. time (min.)	Percentage cnx. follow planes (%)
1	41.17	64.71
2	41.86	64.71
3	42.06	66.67
4	43.63	64.71
5	47.84	66.67
6	47.94	76.47
7	49.41	62.75
8	49.30	78.00
9	49.30	80.00
10	50.50	78.00
11	54.18	81.63
12	55.82	73.47
13	57.20	74.00
14	60.32	76.60
15	58.72	74.47
16	63.65	75.00
17	61.25	85.42
18	74.67	66.67
19	86.82	61.36

optimization problems generally thought to be too formidable for the explicit modeling of uncertainty.

Acknowledgments

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. Additional thanks are due to Tina Shaw and Ellis Johnson for allowing the authors to use their pairing generator and set partitioning solver, and to the referees and associate editor for their valuable comments, which greatly improved this paper.

References

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

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

Arabeyre, J., J. Fearnley, F. C. Steiger, W. Teather. 1969. The airline crew scheduling problem: A survey. *Transportation Sci.* **3**(2) 140–163.

Argüello, M. F., J. F. Bard, G. Yu. 1998. Models and methods for managing airline irregular operations. G. Yu, ed. *Operations Research in the Airline Industry*. Kluwer Academic Publishers, Boston, MA, 1–45.

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

Birge, J. 1982. The value of the stochastic solution in stochastic linear-programs with fixed recourse. *Math. Programming* **24**(3) 314–325.

Birge, J., F. Louveaux. 1997. *Introduction to Stochastic Programming*. Springer, New York.

Cao, J.-M., A. Kanafani. 1997a. Real-time decision support for integration of airline flight cancellations and delays Part I: Mathematical formulation. *Transportation Planning Tech.* **20** 183–199.

Cao, J.-M., A. Kanafani. 1997b. Real-time decision support for integration of airline flight cancellations and delays Part II: Algorithm and computational experiments. *Transportation Planning Tech.* **20** 201–217.

Chan, T. J., C. A. Yano. 1992. A multiplier adjustment approach for the set partitioning problem. *Oper. Res.* **40**(1) S40–S47.

Chu, H. D., E. Gelman, E. L. Johnson. 1997. Solving large scale crew scheduling problems. *Eur. J. Oper. Res.* **97** 260–268.

Desrosiers, J., Y. Dumas, M. Desrochers, F. Soumis, B. Sanso, P. Trudeau. 1991. A breakthrough in airline crew scheduling. Technical Report G-91-11, Cahiers du GERAD, Montreal.

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

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

Gershkoff, I. 1989. Optimizing flight crew schedules. *Interfaces* **19**(4) 29–43.

Graves, G. W., R. D. McBride, I. Gershkoff, D. Anderson, D. Mahidhara. 1993. Flight crew scheduling. *Management Sci.* **39**(9) 736–745.

Hoffman, K. L., M. Padberg. 1993. Solving airline crew scheduling problems by branch-and-cut. *Management Sci.* **39**(6) 657–682.

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

Jarrah, A. Z., G. Yu, N. Krishnamurthy, A. Rakshit. 1993. A decision support framework for airline flight cancellations and delays. *Transportation Sci.* **27**(3) 266–280.

Johnson, E., T. Shaw, R. Ho. 1999. Modeling tools for airline crew-scheduling and fleet-assignment problems. T. A. Ciriani, ed. *Operational Research in Industry*. MacMillan Press, Basingstoke, U.K., 1–24.

Kelly, T. P. 2000. Heap-based priority queue implementation in ANSI C. 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, G. L. Nemhauser. 2001. Solving large airline crew scheduling problems: Random pairing generation and strong branching. *Comput. Optim. Appl.* **20**(1) 73–91.

Klein Haneveld, W. K., M. H. van der Vlerk. 1999. Stochastic integer programming: General models and algorithms. *Ann. Oper. Res.* **85** 39–57.

Lettovsky, L., E. L. Johnson, G. L. Nemhauser. 2000. Airline crew recovery. *Transportation Sci.* **34**(4) 337–348.

Marsten, R. E., M. R. Muller, C. L. Killion. 1979. Crew planning at flying tiger: A successful application of integer programming. *Management Sci.* **25**(12) 1175–1183.

Mathaisel, D. F. X. 1996. Decision support for airline system operations control and irregular operations. *Comput. Oper. Res.* **23**(11) 1083–1098.

McDowell, E. 1997. When weather is the enemy. *The New York Times* (1/27/97) D1, D20.

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

Rakshit, A., N. Krishnamurthy, G. Yu. 1996. System operations advisor: A real time decision support system for managing airline operations at United Airlines. *Interfaces* **26**(2) 50–58.

Rubin, J. 1973. A technique for the solution of massive set covering problems, with applications to airline crew scheduling. *Transportation Sci.* **7**(1) 34–48.

Ryan, D., B. Foster. 1981. An integer programming approach to scheduling. A. Wren, ed. *Computer Scheduling of Public Transport*. North-Holland Publishing Company, Amsterdam, 269–280.

Schaefer, A., E. Johnson, A. Kleywegt, G. Nemhauser. 2000. Robust airline crew scheduling. Institute for Operations Research and Management Science Conference, San Antonio, TX.

Schultz, R., L. Stougie, M. H. van der Vlerk. 1996. Two-stage stochastic integer programming: A survey. *Statistica Neorlandica* **50**(3) 404–416.

Stojković, M., F. Soumis, J. Desrosiers. 1998. The operational airline crew scheduling problem. *Transportation Sci.* **32**(3) 232–245.

Stougie, L., M. H. van der Vlerk. 1997. Stochastic integer programming. F. M. M. Dell'Amico, S. Martello, eds. *Annotated Bibliographies in Combinatorial Optimization*. Wiley, Chichester, U.K., 127–141.

Teodorović, D. 1985. A model for designing the meteorologically most reliable schedule. *Eur. J. Oper. Res.* **21** 156–164.

Teodorović, D., S. Guberinić. 1984. Optimal dispatching strategy on an airline network after a schedule perturbation. *Eur. J. Oper. Res.* **15** 178–182.

Teodorović, D., G. Stojković. 1995. Model to reduce airline schedule disturbances. *J. Transportation Engrg.* **121** 324–331.

Wark, P., J. Holt, M. Rönnqvist, D. Ryan. 1997. Aircrew schedule generation using repeated matching. *Eur. J. Oper. Res.* **102** 21–35.

Wedelin, D. 1995. An algorithm for large scale 0-1 integer programming with application to airline crew scheduling. *Ann. Oper. Res.* **57** 283–301.

Wei, G., G. Yu, M. Song. 1997. Optimization model and algorithm for crew management during airline irregular operations. *J. Combin. Optim.* **1** 305–321.

Yan, S., C.-G. Lin. 1997. Airline scheduling for the temporary closure of airports. *Transportation Sci.* **31**(1) 72–82.