A Branch-and-Price Approach for Operational Aircraft Maintenance Routing

by

Abdulkadir Sarac*, Rajan Batta** and Christopher M. Rump**

* Curbell Inc.
7 Cobham Drive
Orchard Park, NY 14127, USA

**Department of Industrial Engineering
342 Lawrence D. Bell Hall
University at Buffalo (SUNY)
Buffalo, NY 14260-2050, USA

Revised: April 2003

Abstract

In recent years, considerable effort in the field of operations research has been paid to optimizing airline operations, including the logistics of an airline’s fleet of aircraft. We focus on the problem of aircraft routing, which involves generating and selecting a particular route for each aircraft of a sub-fleet that is already assigned to a set of feasible sequences of flight legs. Similar studies typically focus on long-term route planning. However, stochastic events such as severe weather changes, equipment failures, variable maintenance times, or even new regulations mandated by the Federal Aviation Administration (FAA) play havoc on these long-term plans. In addition, these long-term plans ignore detailed maintenance requirements by considering only one or two of the primary maintenance checks that must be performed on a regular, long-term basis. As a result, these plans are often ignored by personnel in airline operations who are forced on a daily basis to develop quick, ad hoc methods to address these maintenance requirements and other irregular events. To address this problem, we develop an operational aircraft maintenance routing problem formulation that includes maintenance resource availability constraints. We propose a branch-and-price algorithm for solving this problem, which, due to the resource constraints, entails a modification of the branch-on, follow-on branching rule typically used for solving similar problems. Through computational testing, we explore the efficiency of this solution approach under a combination of heuristic choices for column (route) generation and selection.
1. Introduction

Operations Research has been a vital tool for tackling the many logistical problems faced by the airline industry. Airplane logistics are focused in four main areas: schedule preparation, fleet assignment, aircraft routing, and disruption recovery. In schedule preparation, the airlines prepare a list of flight legs along with departure and arrival times. This is the main product sold by airlines. This product does not specify which aircraft types are to fly which flight legs. This question is answered by the fleet assignment problem. After the fleet assignment problem has been solved, airlines face the problem of aircraft routing. Aircraft (tail number) routing assigns aircraft (tail numbers) to pre-generated routes. While these assignments are being performed, maintenance constraints mandated by Federal Aviation Administration (FAA) are considered.

Currently, these first three problems are solved well in advance, based on forecasts of customer demand and competitor supply for the various flight legs. This results in long-term plans that typically prescribe that the airplanes follow a cyclic path that revisits a maintenance station on a regular basis. However, the airline industry operates in a dynamic environment, where many unplanned events force airlines to modify their existing long-term plans. Such unexpected events or disruptions include severe weather changes, unplanned equipment failures, and emergency maintenance requests from the FAA and aircraft manufacturers. Disruption recovery is the process of reacting to these operational disruptions. The decision-making in disruption recovery is the process of rerouting in order to recover aircraft, crew, and passengers.

In this dynamic environment, the airlines try to stay competitive by responding to these changes quickly while keeping their daily operations running as smoothly as possible. Doing so is quite a challenge, particularly because the complex logistical routing plans are developed long-term, and unexpected events that upset these plans occur daily. This leaves airline personnel
with the daily challenge of adjusting long-term plans to fit immediate operational constraints posed by these unexpected events.

Although there is much literature that deals with aircraft routing on a long-term planning basis, there is little work on the more pressing operational routing problem faced daily by the airlines. We address this weakness by considering the problem of operational aircraft maintenance routing. This means that instead of developing optimal long-term routing techniques, we intend to develop methods to enable airlines to perform short-term aircraft routing in the face of irregular operations.

This paper is organized in the following fashion. Section 2 presents a brief literature review of aircraft fleeting, routing and disruption recovery. Section 3 discusses the daily operational aircraft maintenance routing problem and its characteristics. A mathematical formulation of the problem is given in Section 4. Section 5 explains the proposed branch-and-price solution methodology, including a modification of the branch-on, follow-on branching rule typically used for solving similar problems. In Section 6 we report the results of a computational study to test the effects of alternative heuristic methods used in the proposed solution approach. Finally, in Section 7 we provide some concluding remarks.

2. Literature Review

A rich body of literature addresses airline problems (fleet assignment, aircraft routing, yield management, etc.) and the use of operations research tools for these problems. The studies related to the problem at hand may be classified into three groups, fleeting-related studies, routing-related studies, and disruption recovery studies. The following paragraphs briefly describe some of these major studies.
The earlier research primarily focused on the classical fleet assignment problems. Abara (1989) explained the classical fleet assignment problem, which included constraints on flight coverage, equipment continuity, aircraft count, and some operational constraints such as gate availability, noise restriction, etc. Subramanian et al. (1994) introduced a large-scale mixed integer linear programming (MILP) formulation, COLDSTART, for the classical fleet assignment model including some maintenance and crew pairing constraints along with the concept of one-stop flights and passenger spill costs.

Hane et al. (1995) formulated and solved a daily fleet assignment problem as an MILP. Clarke et al. (1996) extended the study in Hane et al. (1995) using some maintenance and crew considerations. The maintenance requirements were classified into two groups, short-term maintenance (A-check) and long-term maintenance (B-check), based upon the time requirements of the maintenance. Using the same reduction methods in Hane et al. (1995), the authors solved this problem within two to five hours. The solutions obtained in both studies in Clarke et al. (1996) and Hane et al. (1995) were aggregate maintenance feasible fleet assignments, not necessarily a maintenance feasible aircraft tail number routing. Moreover, both studies have no evidence of any daily operational resource constraints.

While some researchers focused on the fleet assignment problem as listed above, other researchers studied aircraft routing problems, which assume that the fleet assignment problem is already solved. Daskin and Panayotopoulos (1989) formulated a route selection problem to maximize profits in a single hub-and-spoke network as an MILP. Feo and Bard (1989) presented a model used by American Airlines (AA) both to locate maintenance stations and to develop flight schedules that better met cyclical demand for A-checks. Kabani and Patty (1993) presented an A-check feasible route selection model, which was formulated as a set-partitioning problem.
Clarke et al. (1997) presented a mathematical formulation for the aircraft rotation problem in which a specific route for each aircraft was determined under the maintenance feasibility constraints. This formulation bears many similarities to the asymmetric traveling-salesman problem. Gopalan and Talluri (1998) developed a polynomial-time algorithm for long-term aircraft maintenance routing that employed a route-modification scheme of swaps and interchanges among aircraft to form a feasible solution.

More recently, two studies have combined fleet assignment and aircraft routing. These two problems were traditionally solved separately due to their large size. Ignoring maintenance constraints allowed Desaulniers et al. (1997) to reduce the size of the problems and develop two models for a combined fleeting and routing problem called the Daily Aircraft Routing and Scheduling Problem (DARSP). Barnhart et al. (1998) also developed a model to solve both fleeting and routing problems simultaneously. The authors attempted to solve these two problems together to overcome the traditional drawback of a classical fleet assignment problem: the lack of a well-spaced maintenance opportunity for each aircraft of the sub-fleet.

In the disruption recovery area, there are some related studies in the literature. Some of these earlier studies include Teodorovic and Guberinic (1984) and (1990), Jarrah et al. (1993), Cao and Kanafani (2000). These studies focused on certain aspects of disruption recovery such as minimizing the customer delay, minimizing the number of aircraft needed to recover from a disruption, minimizing the number of cancelled flight legs, minimizing the schedule perturbations, and maximizing the profit with consideration for cancellation and delay cost penalties. These studies are primarily considered to be “aircraft recovery” studies. There are other studies, Lettovsky et al (2000) and Stojkovic (1998), for example, that focus on “crew recovery” instead. The current trend in disruption recovery is on building integrated models in
which aircraft and crew recovery are considered together. Letovsky (1997) presented a model for integrated recovery in which crew, aircraft routing, and passenger recovery was considered all together. Recently, Rosenberger (2002) provides a comprehensive review of the related literature for integrated (hybrid) modeling.

Despite the similarity of these many studies to our problem, none have addressed our problem. Most of these studies are for (long-term) planning purposes rather than (short-term or daily) operational purposes except for the disruption recovery studies. The few studies that are operational in nature fail to consider the legal remaining flying hours of the aircraft and the resource constraints at the maintenance stations. It is with this in mind we address the daily operational aircraft maintenance routing problem in the next section.

3. Problem Definition and Characteristics

The components of an aircraft may be grouped into two main classes, resident and non-resident components (Bandla, 1994). Resident components consist of the aircraft and its vital systems and are grouped for maintenance purpose according to their functionalities. These groups are maintained on a regular basis via letter checks. For example, the A-check (performed after 65 hours of flying) involves visual inspection of the aircraft and its vital systems (Clarke et al., 1997). On the other hand, non-resident components (call-outs) such as an engine master chip detector need to be maintained individually. Thus, each non-resident component has its own maximum legal remaining flying hour limit.

For safety considerations, the Federal Aviation Administration (FAA) mandates these legal remaining flying hour limits. Violations result in severe financial penalties. Most airlines employ automated routing systems to keep track of the accumulated flying hours for both letter
checks and for non-resident components of aircraft. When the accumulated flying hours of an aircraft reach a predefined level, the aircraft is labeled as a “high-time” aircraft and is listed in the daily list of high-time aircraft. An aircraft may appear more than once in this list due to multiple maintenance requirements. The aircraft on the daily high-time list need to be routed to feasible maintenance stations where there exist enough man-hours and maintenance slots to perform the required maintenance.

Figure 1. Time limit definitions for an aircraft.

\[ T_1, T_4 \]: Maximum legal remaining flying hours.

\[ T_2 \]: Time at which the latest maintenance was performed.

\[ T_3 \]: Time at which aircraft becomes labeled “high-time”.

\[ T_4 \]: Time at which next maintenance is performed.

\[ T_1 \]: Time after which aircraft cannot fly without maintenance.

In practice, the task of routing high-time aircraft for maintenance involves two major steps. In the first step, multiple maintenance requirements of a high-time aircraft are batched into a package. The legal remaining flying hours of this package is defined as the minimum of the legal remaining flying hours of the maintenance requirements in the package. In the second step, all aircraft of a particular sub-fleet are routed in such a way that all flight legs are covered and high-time aircraft end up at a feasible maintenance station. Airlines perform these two main steps using some heuristic methods to optimize various performance measures.

These heuristic methods do not always find a feasible solution to the problem at hand, however. The infeasibility occurs because a high-time aircraft is forced to stay overnight at a station where there does not exist either enough man-hours or necessary equipment to perform
the maintenance. To continue the daily operations and avoid severe financial penalties, airlines are forced to move their maintenance personnel and equipment from station to station, at substantial additional cost.

The daily aircraft maintenance routing is a problem in which the objective is to minimize the total daily maintenance costs without violating legal remaining flying hours of each aircraft, subject to the resource constraints (available man-hours and maintenance slots) of the maintenance stations. The optimal solution of this problem will not only minimize total daily maintenance costs, but also reduce the movement of maintenance personnel and equipment from one station to another. Because it is difficult to define and calculate a daily maintenance cost for an aircraft, we utilize a surrogate objective to accomplish our primary goal. As a surrogate objective, we minimize the total cushion time (total unutilized legal remaining flying hours) of a sub-fleet. In other words, we intend to maximize the utilization of the total green time (total legal remaining flying hours) for a sub-fleet. (See Figure 1.)

A feasible solution is one in which all high-time aircraft are routed to a feasible maintenance station for the given day. The maintenance for all high-time aircraft is assumed to happen at the end of the day during the overnight stay. If an aircraft requires a longer maintenance than can be achieved overnight, then enough maintenance time can be ensured by either selecting its ending flight leg to finish early in the day or its starting flight leg for the following day to begin late in that day.

The time horizon of the problem is assumed to be one day. (The horizon may in fact be shorter than a day since a disruption might occur in the middle of the day, necessitating a recovery for the remainder of the flight leg network with the given start locations for the aircraft.) A time horizon longer than one day, say one week, becomes more of a planning horizon
that is too optimistic for the frequent disruptions that plague the airline industry. The solution to a problem with a longer time horizon may indeed better utilize the remaining flying hours of the aircraft by delaying the maintenance of some high-time aircraft beyond the current day but still within the legal limits. These opportunities are relatively rare, however, since the airlines impose fairly tight maintenance time limits, even more strict than what the FAA requires.

With a longer time horizon, one could also argue for including “non-high-time” aircraft, i.e., those aircraft with remaining legal flying hours of two days or more. However, maintaining these aircraft earlier than necessary is counterproductive to the goal of maximizing the utilization of aircraft green time and consumes scarce maintenance slot and man-hour resources needed for the high-time aircraft.

4. Problem Formulation

We start by explaining the network structure that will form the basis for the mathematical formulation of our problem.

4.1. Network Structure

Airline flight schedules are traditionally depicted as flight-leg networks, where nodes represent cities and arcs between nodes represent flight legs connecting these cities. A difficulty with this representation involves keeping track of the departure and arrival times of each arc in the network. To remove this difficulty, we use a connection network structure in which nodes represent flight legs and arcs represent feasible connections among the flight legs. In this so-called connection network, the presence of an arc \((i,j)\) between node (flight leg) \(i\) and node (flight leg) \(j\) means that the departure city of node \(j\) is the same as the arrival city of node \(i\), and the
arrival time of flight leg $i$ plus the turn time (the time needed to prepare the aircraft for the next flight) is less than or equal to the departure time of flight leg $j$. In other words, if the same aircraft can successively fly flight legs $i$ and $j$ then arc $(i,j)$ must exist. It is trivial to show that the connection network is always acyclic (Sarac, 2000).

The connection network includes dummy source and sink nodes that are connected to appropriate starting and ending flight legs, respectively. The arcs from the dummy source to the other nodes in the network have a flight duration of zero, while the arcs to the dummy sink node will have a duration of the incoming flight leg (node). Furthermore, the underlying flight schedule of the connection network must be balanced, which means the number of outbound flights of a particular city/airport must equal the number of inbound flights of the same city (Clarke et al., 1997).

4.2. Mathematical Formulation

We use a set-partitioning based formulation in which decision variables represent feasible routes for aircraft. We choose this approach for several reasons: it can easily incorporate route-based constraints, it focuses on the feasible assignment of routes to aircraft, and such formulations have proven to be very promising for general vehicle routing problems (Barnhart et al., 1998; Desrosiers et al., 1984; Desrochers et al., 1989; Desrochers et al., 1992; Dumas et al., 1991; Krishnamurthy, 1990; Ryan and Foster, 1981; Savelsbergh and Sol, 1998; Vance et al., 1997). Table 1 summarizes the notation for our formulation.
Table 1. Notation Summary

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i$</td>
<td>Index for flight legs.</td>
</tr>
<tr>
<td>$j$</td>
<td>Index for routes.</td>
</tr>
<tr>
<td>$m$</td>
<td>Index for maintenance types.</td>
</tr>
<tr>
<td>$k$</td>
<td>Index for aircraft.</td>
</tr>
<tr>
<td>$s$</td>
<td>Index for overnight stations.</td>
</tr>
<tr>
<td>$o$</td>
<td>Dummy source node used in sub-problem formulation.</td>
</tr>
<tr>
<td>$t$</td>
<td>Dummy sink node used in sub-problem formulation.</td>
</tr>
<tr>
<td>$A$</td>
<td>Set of connection arcs in sub-problem formulation.</td>
</tr>
<tr>
<td>$R_k$</td>
<td>Set of feasible routes generated for aircraft $k$.</td>
</tr>
<tr>
<td>$N$</td>
<td>Set of flight legs.</td>
</tr>
<tr>
<td>$M$</td>
<td>Set of maintenance types.</td>
</tr>
<tr>
<td>$S_m$</td>
<td>Set of overnight station where maintenance $m$ can be performed.</td>
</tr>
<tr>
<td>$K$</td>
<td>Set of aircraft.</td>
</tr>
<tr>
<td>$t_i$</td>
<td>Flight duration for node $i$.</td>
</tr>
<tr>
<td>$c_{jk}$</td>
<td>Cost of selecting the $j$th route of aircraft $k$.</td>
</tr>
<tr>
<td>$\tau_k$</td>
<td>Legal remaining flying hours of aircraft $k$.</td>
</tr>
<tr>
<td>$a_{mk}$</td>
<td>Maintenance man-hours are needed to perform maintenance $m$ for aircraft $k$.</td>
</tr>
<tr>
<td>$b_{mk}$</td>
<td>$\begin{cases} 1 &amp; \text{if aircraft } k \text{ needs maintenance type } m \ 0 &amp; \text{otherwise} \end{cases}$</td>
</tr>
<tr>
<td>$d_{js}$</td>
<td>$\begin{cases} 1 &amp; \text{if the } j\text{th route of aircraft } k \text{ ends at overnight station } s \ 0 &amp; \text{otherwise} \end{cases}$</td>
</tr>
<tr>
<td>$\lambda_{is}$</td>
<td>$\begin{cases} 1 &amp; \text{if arrival city of flight leg } i \text{ is overnight station } s. \ 0 &amp; \text{otherwise} \end{cases}$</td>
</tr>
<tr>
<td>$\gamma_{ji}$</td>
<td>$\begin{cases} 1 &amp; \text{if the } j\text{th route of aircraft } k \text{ contains flight leg } i \ 0 &amp; \text{otherwise} \end{cases}$</td>
</tr>
<tr>
<td>$L_{ms}$</td>
<td>Available man-hours for maintenance $m$ at overnight station $s$.</td>
</tr>
<tr>
<td>$Z_{ms}$</td>
<td>Number of available slots for maintenance type $m$ at overnight station $s$.</td>
</tr>
<tr>
<td>$w_{in}$</td>
<td>$\begin{cases} 1 &amp; \text{if the connection arc } (i,n) \text{ is selected,} \ 0 &amp; \text{otherwise} \end{cases}$</td>
</tr>
<tr>
<td>$y_{jk}$</td>
<td>$\begin{cases} 1 &amp; \text{if the } j\text{th route of aircraft } k \text{ is selected,} \ 0 &amp; \text{otherwise} \end{cases}$</td>
</tr>
</tbody>
</table>

The cost coefficients, $c_{jk}$, associated with decision variables (routes) represent the cushion time (Figure 1) for the aircraft. If aircraft $k$ is not a high-time aircraft, then the corresponding
cost coefficients of all its routes take zero value. On the other hand, if aircraft $k$ is a high-time aircraft, then $c_{j}^{k}$ is equal to the legal remaining flying hours of the aircraft $k$ minus the duration of route $j$. The decision variable, $y_{j}^{k}$, represents a feasible route $j$ (string of flight legs) for the aircraft $k$ of a particular sub-fleet. Moreover, there are four sets of constraints: aircraft coverage (1), flight leg coverage (2), man-hour availability (3), and slot availability (4). The general mathematical formulation becomes:

\[
(0) \quad \min \sum_{k \in K} \sum_{j \in R_{k}} c_{j}^{k} y_{j}^{k} \quad \text{(dual variables)}
\]

subject to

\[
(1) \quad \sum_{j \in R_{k}} y_{j}^{k} = 1 \quad \forall k \in K \quad (\pi_{k})
\]

\[
(2) \quad \sum_{k \in K} \sum_{j \in R_{k}} y_{j}^{k} = 1 \quad \forall i \in N \quad (\sigma_{i})
\]

\[
(3) \quad \sum_{k \in K} \sum_{j \in R_{k}} a_{ij}^{k} d_{js}^{k} y_{j}^{k} \leq L_{ms} \quad \forall m \in M \text{ and } s \in S_{m} \quad (\beta_{ms})
\]

\[
(4) \quad \sum_{k \in K} \sum_{j \in R_{k}} b_{ij}^{k} d_{js}^{k} y_{j}^{k} \leq Z_{ms} \quad \forall m \in M \text{ and } s \in S_{m} \quad (\alpha_{ms})
\]

\[
(5) \quad y_{j}^{k} \in \{0,1\} \quad \forall k \in K \text{ and } j \in R_{k}
\]

The above formulation aims to minimize the number of unused legal flying hours while ensuring aircraft count (1) and flight leg coverage (2) by maintaining feasibility for available maintenance man-hours (3) and the number of maintenance slots (4). The necessity of using two separate constraints for the maintenance hangar capacity is due to the possibility of having some man-hours but no slots at some maintenance station or vice versa. This situation forces airlines to make undesirable costly maintenance personnel moves from station to station.

The problem (P) has an exponential number of feasible routes and cannot be solved directly in an efficient fashion. In fact, this problem is NP-hard since it contains the NP-complete partition problem as a special case (Sarac, 2000). Hence, in the next section we apply a column generation scheme that uses a small subset of feasible routes during the solution process instead of explicitly enumerating all feasible routes.
5. Branch-and-Price Algorithm

In our column-generation scheme, the LP relaxation of the problem, called the *restricted master problem* (RMP), is optimized over a subset of promising routes. These promising feasible routes, which are generated on an as-needed basis, are determined by another optimization problem called the *pricing problem*. However, the solution obtained from the master problem is not necessarily integer. Hence, we need to use a branch-and-bound search tree. At each node of the search tree we employ column generation in what has been recently coined *branch-and-price* (Barnhart et al., 1998; Savelsbergh and Sol, 1998). Although the general problem formulation and solution strategy for branch-and-price is well defined (see Figure 2), its implementation details show great variety from problem to problem. In this section we explain these details for our problem.

Figure 2. General outline of a branch-and-price algorithm.
5.1. Preprocessing

Preprocessing is used to eliminate unnecessary connection arcs in order to simplify the structure of the connection network. The preprocessing focuses on four special types of nodes in the connection network: nodes with their indegree equal to zero, nodes with their outdegree equal to zero, nodes with their indegree equal to one, and nodes with their outdegree equal to one.

If a node-$i$ has its indegree equal to zero, then it is the designated starting flight leg for a route. Due to the flight leg coverage constraint, this implies that there exists at least one aircraft on the ground at the departure city of node-$i$. Similarly, if a node-$i$ has its outdegree equal to zero, then it is a designated ending flight leg for a route. This implies that the arrival city of node-$i$ is an overnight station where an aircraft may stay overnight at the end of the day. If the arrival city of node-$i$ is not an overnight station, then this indicates the existence of a discontinuity in the connection network under the assumed one-day time horizon. This occurs due to an ill-designed flight schedule which we assume not to be the case.

If a node-$i$ has its indegree equal to one, then node-$i$ has only one incoming connection arc, call it $(j,i)$, and node-$i$ must be flown right after node-$j$ in order to be covered. Therefore, we can eliminate all outgoing arcs of node-$j$ except for arc $(j,i)$. We may even aggregate node-$i$ and node-$j$ into a super node-$ji$. Similarly, if a node-$i$ has its outdegree equal to one, then node-$i$ has only one outgoing connection arc, call it $(i,j)$, and node-$i$ and node-$j$ must be flown successively. Therefore, we can eliminate all incoming arcs of node-$j$ except for the arc coming from node-$i$. Once again, we may aggregate node-$i$ and node-$j$ into a super node-$ij$.

This preprocessing reduces the problem size without changing the characteristics of the connection network, i.e., it remains connected and acyclic (Sarac, 2000).
5.2. Generation of an Initial Starting Feasible Solution

To start the column generation at each node of branch-and-bound tree, an initial starting feasible solution to the restricted master problem (RMP) can be provided to pass proper dual information to the pricing problems. However, finding an initial starting feasible solution is difficult due to the existence of resource availability constraints. We use the following route generation scheme:

- Sort the aircraft in ascending order of their legal remaining flying hours. (Note that non-high-time aircraft are assumed to have infinite legal remaining flying hours.)
- Starting with the aircraft with the least legal remaining flying hours, generate a route for each aircraft, using FIFO when selecting consecutive connections. Once a flight leg is selected for a route, it is eliminated from the connection network so as to be used at most once.

This route generation scheme does not necessarily find a feasible solution for (RMP), although it will generate feasible routes for some of the aircraft. The intent of this scheme is to generate an initial set of feasible routes in which each flight leg is covered at most once.

If the above scheme fails to identify an initial feasible solution, we employ a two-phase method that is similar to the one used by the simplex algorithm to identify an initial basic feasible solution (Savelsbergh and Sol, 1998). In the first phase, we introduce an artificial variable, with an associated large penalty cost, into each flight leg coverage constraint. We use the length of the longest path in the connection network as the penalty on the artificial variables to make these variables unattractive in an optimal solution.
By adding these artificial variables, we can even solve (RMP) with an empty initial set of feasible routes. Upon the detection of a feasible solution, we begin the second phase by eliminating all of the artificial variables in order to reduce the number of variables in the problem. However, we may reintroduce these artificial variables back into the problem after branching to a newly generated node of the branch-and-bound tree, since the branching scheme might destroy feasibility.

5.3. Solving the Restricted Master Problem

In the column generation method, the optimality of (RMP) is confirmed by the solutions of pricing problems, one for each aircraft. Suppose that we explicitly know a subset $\overline{R}_k$ of all feasible routes for each aircraft $k \in K$. Moreover, assume that we solve (RMP) using these subsets $\overline{R}_k$’s and obtain a solution $y$ and corresponding dual variables $(\pi_k, \sigma_i, \beta_{ms}, \alpha_{ms})$, where the $\pi_k$’s are associated with aircraft coverage constraints, $\sigma_i$’s are associated with flight leg coverage constraints, $\beta_{ms}$’s are associated with maintenance man-hours availability constraints, and $\alpha_{ms}$’s are associated with maintenance slot availability constraints. From duality theory, we know that solution $y$ is optimal if and only if no promising columns (routes) are identified by the pricing problems. In other words, the reduced cost $\overline{c}_j^k$ for each $k \in K$ and each $j \in R_k$ are greater than or equal to zero:

$$\overline{c}_j^k = c_j^k - \pi_k - \sum_{m} \sum_{s} (a_{ms}^m \beta_{ms}^m + b_{ms}^m \alpha_{ms}^m) d_{js}^k - \sum_{i} \gamma_i^k \sigma_i \geq 0 \quad \forall k \in K \text{ and } j \in R_k$$  \hspace{1cm} (6)

where $$c_j^k = \tau_j^k - \sum_{i \in N} \gamma_i^k t_i.$$

The confirmation of the optimality for (RMP) is obtained by solving $|K|$ pricing problems that are in the following general form:

$$\omega = \min \{ \overline{c}_j^k \mid j \in R_k \} \quad \forall k \in K$$  \hspace{1cm} (7)
If any of the above pricing problems generates a solution with \( \omega < 0 \), then solution \( y \) is not optimal. Otherwise, it is optimal. If the solution \( y \) is not optimal, then the new columns (routes) with negative reduced costs are added into (RMP) and (RMP) is re-optimized using the dual simplex method. This process continues until either the optimal solution is found or infeasibility is confirmed.

### 5.4. Solving Pricing Problems

The pricing problems presented below are constrained shortest path (CSP) problems with a single resource constraint on the legal remaining flying hours. The objective (8) is to minimize the reduced cost. In other words, we intend to find the route with minimum reduced cost while satisfying the legal remaining flying hours of the aircraft. The objective function is derived from equation (6) by rearranging the terms. The constraint in (9) is the classic shortest path flow conservation constraint and the constraint in (10) is the legal remaining flying hours constraint defined for each high-time aircraft.

\[
(8) \quad \min \left\{ \tau_k - \pi_k - \sum_{[i,j] \in A} \left( \sum_{m \in M} \sum_{s \in S_m} (a_m^k \beta_m^s + b_m^k \alpha_m^s) \lambda_{is} + (t_i + \sigma_i)\right)w_{in} \right\}
\]

subject to

\[
(9) \quad \sum_{\{n ; (i,n) \in A\}} w_{in} - \sum_{\{n ; (n,i) \in A\}} w_{ni} = \begin{cases} 1 & \text{if } i = o \\ 0 & \text{otherwise} \\ -1 & \text{if } i = t \end{cases} \quad \text{(CSP)}
\]

\[
(10) \quad \sum_{[i,j] \in A} t_i w_{in} \leq \tau_k
\]

\[
(11) \quad w_{in} \in \{0,1\} \quad \forall (i,n) \in A
\]

CSP problems are proven to be at least as hard to solve as NP-complete problems (Handler and Zang, 1980). The complexity of the problem increases since the addition of a
knapsack-type constraint destroys the unimodularity property of the classical shortest path problem. To solve pricing problems (CSP), we employ the generalized permanent labeling algorithm developed by Desrochers and Soumis (1988). The proposed method is simply a dynamic programming shortest path algorithm with a single resource constraint similar to the ones used in Desauliniers et al. (1997) and Desrochers, Desrosiers and Solomon (1992). The algorithm is based on the definition of the concept of a generalized bucket and on a specific order of handling the labels.

The generalized bucket concept eliminates the need to search for the minimum existing label at each step of the algorithm. Instead, the algorithm picks any available label within the specific range defined by the current bucket. In a dynamic programming sense, any available state within the defined range of the bucket is selected to evaluate. The specific order for treating the labels in the algorithm not only defines a dominance rule, which eliminates unpromising labels (or states in dynamic programming) from the treatment, but also ensures that it is impossible to improve a label treated previously. These two concepts make the algorithm work very efficiently.

5.5. Branching Scheme

The operational aircraft maintenance routing problem is an integer-programming (IP) problem whose solution must be integral. However, the optimal solution to (RMP) is not necessarily integer. In order to obtain an integral solution, we need to use a branch-and-bound method for which a valid branching scheme must be defined. A valid branching scheme must cut out current fractional solutions from the solution space of the problem and validly partition the solution space of the problem. The scheme should also be easy to incorporate into the
pricing problems without losing tractability. Since the standard binary branching scheme almost always complicates the pricing problems (Savelsbergh, 1997; Savelsbergh and Sol, 1998; Vance et al., 1993; Vance et al., 1997), we need to apply some other branching scheme by exploiting the structure of the problem at hand.

In the literature, various versions of a branching rule that was originally developed by Ryan and Foster (1981) have been successfully used for set-partitioning problems (Desrosiers et al., 1984; Desrochers et al., 1989; Savelsbergh, 1997; Savelsbergh and Sol, 1998; Vance et al., 1993; Vance et al., 1997). This branching rule is based on the simple observation that there exist no identical columns in the simplex tableau of the set-partitioning problem. In other words, if the solution to (RMP) is fractional, there exist two fractional variables (columns) such that they both have coefficients of one in one row, and there is another row in which only one of the these fractional columns has a coefficient of one.

In airline problems, particularly crew pairing problems, this branching rule is utilized in such a way that two flight legs are flown consecutively on one branch, whereas these two flight legs are never flown consecutively on the other branch. This rule is named *branch-on follow-on*. It is valid and easy to incorporate into the pricing problems without adding any hard constraints (Vance et al., 1997).

However, branch-on follow-on is not sufficient for our operational aircraft maintenance routing problem since the problem is not a set-partitioning problem. Instead it is a set-partitioning problem with resource availability side constraints. In this problem, two fractional columns may share an identical route (the same string of flight legs) and differ only in the resource availability rows. Thus, two different aircraft may together (each fractional) cover a single route. In other words, we may reach a point at which the solution is fractional and no
branching pair of flight legs might exist to branch on. Therefore, we need to define a new branching scheme, since the naïve use of branch-on follow-on may result in premature pruning.

We modified branch-on follow-on to define two valid branching schemes for our problem. In the proposed branching schemes, there are two phases. In the first phase, the original branch-on follow-on rule is used to branch until the point at which no branching pair of flight legs may be identified and the current optimal solution is fractional. At this point the second phase starts. In the second phase, we used two different approaches.

In the first approach, we continue to branch, now considering both aircraft and flight legs. In other words, the proposed branching scheme will identify *aircraft-specific* pairs of flight legs to branch on. This search is performed in prioritized fashion, in which the aircraft with the least legal remaining flying hours will have the highest priority. If the proposed branching scheme fails to identify a pair of flight legs for the first aircraft, then there exists a column of the aircraft that is set to one (Sarac, 2000). In this case, the search for an aircraft-specific pair of flight legs continues with the aircraft with the second least legal remaining flying hours and so on.

As in the original branch-on follow-on, the proposed branching scheme searches for a pair of flight legs \((n,f)\) for aircraft \(k\) by calculating a value of \(0 < \sum_{j \in R_k^{(n,f)}} y_{kj}^k < 1\), where \(R_k^{(n,f)}\) is the set of routes for aircraft \(k\) containing flight legs \((n,f)\) consecutively. The existence of such a pair \((n,f)\) simply indicates that there exist some other fractional columns under the other aircraft containing the same pair \((n,f)\). It is also possible that there exist more than one \((n,f)\) pair for a particular aircraft \(k\). In that case, we will select the pair \((n,f)\) with the highest \(\sum_{j \in R_k^{(n,f)}} y_{kj}^k < 1\), as in the original branch-on follow-on rule.
After detecting a set $A_{nf}$ of aircraft having nonzero columns sharing the pair $(n,f)$, the proposed branching scheme checks to see if the pair $(n,f)$ has been already used by the original branch-on follow-on during the first phase. If it has been, this means that the current node is on the side of the branch-and-bound search tree where $n$ and $f$ are flown consecutively. In this case, the following aircraft-specific branching is performed to generate $|A_{nf}| + 1$ new nodes in the search tree:

1. $(n,f) - \text{branch 1} \Rightarrow$ only aircraft $k_1$ is allowed to fly $(n,f)$ where $k_1 \in A_{nf}$,
2. $(n,f) - \text{branch 2} \Rightarrow$ only aircraft $k_2$ is allowed to fly $(n,f)$ where $k_2 \in A_{nf}$,
3. \[\cdots\]
4. $(n,f) - \text{branch } |A_{nf}| \Rightarrow$ only aircraft $k_{|A_{nf}|}$ is allowed to fly $(n,f)$ where $k_{|A_{nf}|} \in A_{nf}$,
5. $(n,f) - \text{branch } |A_{nf}| + 1 \Rightarrow$ none of the aircraft in $A_{nf}$ is allowed to fly $(n,f)$.

If the pair $(n,f)$ that is identified for aircraft $k$ has not been used by the original branch-on follow-on during the first phase, this simply indicates that these two flight legs $(n,f)$ have always been flown consecutively so far. This also means that these pairs will be more likely flown consecutively at optimality. However, this does not necessarily guarantee that the optimal solution will have $(n,f)$ flown consecutively. Therefore, the branching scheme generates another new node in addition to the ones above. This additional node is not aircraft-specific; instead, it is general as in the original branch-on follow-on:

1. $(n,f) - \text{branch } |A_{nf}| + 2 \Rightarrow (n,f)$ must be flown non-consecutively.

The validity proof and the other implementation details of the proposed branching scheme can be found in Sarac (2000).
In the second approach to the second phase of branching, we continue to branch, now considering both aircraft and ending flight legs. In other words, this branching scheme will identify only aircraft-specific ending flight legs to branch on. As in the first approach, the second approach starts at the onset of the second phase of branching, when no branch-on follow-on pairs can be found. At that point, a set $EF$ of the ending flight legs that correspond to fractional nonzero columns in the current optimal is identified. Then, for each ending flight leg $n$ in $EF$, a set $A_n$ of aircraft that have that particular ending flight leg in their nonzero fractional column is formed. The ending flight leg $n^*$ with the maximum number of sharing aircraft is selected as branching ending flight leg. This aircraft specific ending flight leg branching generates $|A_{n^*}| + 1$ new nodes at each branching as follows:

$$(n^*) - \text{branch } 1 \Rightarrow \text{only aircraft } k_1 \text{ is allowed to fly (} n^* \text{) where } k_1 \in A_{n^*},$$

$$\cdots$$

$$(n^*) - \text{branch } |A_{n^*}| \Rightarrow \text{only aircraft } k_{|A_{n^*}|} \text{ is allowed to fly (} n^* \text{) where } k_{|A_{n^*}|} \in A_{n^*},$$

$$(n^*) - \text{branch } |A_{n^*}| + 1 \Rightarrow \text{none of the aircraft in } A_{n^*} \text{ is allowed to fly (} n^* \text{).}$$

The proof of validity and implementation details of the second branching scheme is the same as in the first scheme (Sarac, 2000).

These branching schemes are compared for their computational efficiencies in Section 6.
5.6. Route Generation and Selection

In this section we discuss alternative strategies for route generation and selection.

5.6.1. Route Generation Discipline

The aircraft in the operational aircraft maintenance routing problem are non-identical since each has its own legal remaining flying hours, day starting point, and maintenance requirements. Therefore, as mentioned before, there exists a pricing problem for each aircraft. When generating new columns (routes) in the pricing problem, it would be desirable to add a set of columns that form a feasible solution for (RMP). This is quite unlikely to occur due to existence of complicating resource constraints. However, it is still satisfactory to add a set of columns that covers a subset of flight legs.

For route generation, there are two obvious disciplines. Under the first discipline, the routes are generated independently for each aircraft. The motivation is to generate many promising columns as quickly as possible for each aircraft. The drawback is that the same route might be added for different aircraft. (Recall that the same route does not necessarily represent the same column.)

Under the second discipline, the routes are generated dependently. Prioritizing the aircraft in ascending order of their legal remaining flying hours, we sequentially generate disjoint routes for each aircraft. In this scheme, any flight leg that is used by an aircraft during a single route-generation run will not be available for the remaining aircraft in the same route-generation run. There are two obvious drawbacks to this route-generation scheme. First, a single run may not cover all flight legs. Further, a single run may not be able to supply a promising route for each aircraft.
The effectiveness of these alternative route-generation disciplines will be tested in the computational study section.

5.6.2. Stopping Strategy in Route Selection

In the column generation procedure, we ideally want to select the most promising routes (the routes with the minimum reduced cost) to add into (RMP). However, it is not necessary to find the column with the minimum reduced cost at each route-generation run. It might be initially satisfactory to add any route with the negative reduced cost for the sake of reducing time in the route-generation process. With this in mind, the route-generation process might be terminated by the detection and selection of the first negative reduced cost column. If no negative reduced cost column is detected by the pricing problems, this simply indicates the optimality of (RMP).

Although these premature terminations in the sub-problem solution process might save time and possibly reach a feasible solution quickly, they might also increase both the size of (RMP) and the number of restricted master problems to be re-optimized. The alternative to this flexible termination strategy is to always find and select the most promising route in the route generation process. Since the choice between these alternatives is not clear, we intend to compare them in the computational study to follow.

6. Computational Study

In the study, the proposed branch-and-price algorithm is developed using the C++ programming language. In the program, the commercial software CPLEX 7.5 is used as an LP optimizer to solve the LP relaxation of (RMP) using dual simplex. The algorithm and its
computational runs are executed on a Dell Dimension 8100. The use of such a PC environment to test our algorithm as opposed to using any other platform is due to the availability of commercial software CPLEX 7.5 on the PC.

In the study, we intend to determine the best configuration of the column generation and selection factors introduced in Section 5.6, namely, the route-generation discipline and stopping criterion in the solution of the pricing problem. In addition, we aim to identify the most efficient branching rule of the two proposed in Section 5.5. The common feature of these factors is that each factor has two alternatives.

The choice of route-generation discipline as explained in Section 5.6.1 is the choice between dependent (Dep.) and independent (Ind.) route-generation disciplines. The dependent route generation generates a set of disjoint routes for each route-generation run, whereas the independent route generation allows the same flight leg to be used by more than one aircraft during a single route-generation run.

The choice of stopping strategy in solving the pricing problems involves whether to confirm optimality (Opt.) of the best route or terminate the selection process prematurely (Per.) upon the detection of the first promising route. As mentioned in Section 5.6.2, the alternative of premature stops is motivated by the desire to save time in confirming optimality during the execution of generalized permanent labeling algorithm.

The choice of branching strategy as explained in Section 5.5 is whether to use the aircraft specific follow-on (ACFO) or aircraft specific ending flight leg (ACEF). The goal is to determine whether one strategy is more efficient than the other.
Since there exist two clear alternatives for each factor, this gives us $2^3 = 8$ different cases to test. To compare these eight factor combinations, 10 randomly generated problem instances were created. (The details of these random generations are explained in Section 6.1.)

The results are analyzed using not only solution time statistics but also solution quality statistics. The solution time measures, the time to the first feasible, the time to the best solution, and total solution time, are statistically tested through a two-level factorial design using the commercial statistical software MINITAB 13.

6.1. Test Problems

The ability to solve an instance to optimality strongly depends on the size of the set of feasible solutions, which primarily depends on both the legal remaining flying hours of the aircraft and resource availability at the maintenance stations in a balanced flight-leg network. Hence, we vary these characteristics randomly during the generation of 10 problem instances.

Before constructing the problem instances, we first constructed a daily balanced flight-leg network using the published schedule of a commercial U.S. airline. We constructed a network of 175 flight legs over 19 cities while maintaining necessary conditions for a balanced flight-leg schedule, which is itself a challenging problem. Of these 19 cities, 14 are overnight stations; 5 out of those 14 overnight stations are maintenance stations. With this flight-leg network, we then determined the size of the sub-fleet needed to ensure full coverage of the flight legs. It turned out that 32 aircraft were sufficient to cover the entire flight leg schedule.

After constructing a balanced flight-leg network and its sub-fleet, we construct the problem instances as follows. We first randomly generated aircraft-specific data. We determine if an aircraft is a high-time aircraft or not. If the aircraft is high-time, then we generate its
maintenance requirements among four possible choices along with its maintenance man-hour requirement (which is assumed to follow a uniform distribution between 6 and 10 hours) and its legal remaining flying hours (which is assumed to follow a uniform distribution between 10 and 20 hours). Upon generating aircraft specific data, we generated overnight station specific data. For each pair of maintenance station and maintenance type, we randomly determine both the number of available maintenance man-hours and the number of available maintenance slots. (For details see Sarac, 2000.)

6.2. Analysis of Results

As indicated in the previous section, 10 problem instances are solved under 8 different conditions, which are labeled using the format of “Discipline/Stopping/Branching” as follows: Ind/Opt/ACFO, Ind/Opt/ACEF, Ind/Pre/ACFO, Ind/Pre/ACEF, Dep/Opt/ACFO, Dep/Opt/ACEF, Dep/Pre/ACFO, and Dep/Pre/ACEF. The results are first compared by their time performance statistics and then using solution quality statistics.

6.2.1. Solution Time Analysis

The results are first evaluated using the time performance measures, the time to the first feasible, the time to best solution, and the total solution time. The time to the best solution is not necessarily equal to the total solution time since the problems may require additional time to confirm the optimality. In some instances, we even terminate the execution after 2 hours if a feasible solution is at hand. We believe that the time to the best solution is more important than the total solution time since finding a good feasible solution is imperative for the problem at hand when an unexpected event occurs and interrupts the existing system.
The times to the first feasible solution are presented in Table 2. For problem instances 8 and 9, the initial feasible solution heuristic described in Section 5.2 was able to find a feasible solution immediately. That is why the reported times are significantly lower for these problem instances. Therefore, they are neglected in calculation of the average, standard deviation, minimum, and maximum values in Table 2.

Table 2. Time (in seconds) until the first feasible solution under individual cases.

<table>
<thead>
<tr>
<th>Problem Instance</th>
<th>Ind / Opt / ACFO</th>
<th>Ind / Opt / ACEF</th>
<th>Ind / Pre / ACFO</th>
<th>Ind / Pre / ACEF</th>
<th>Dep / Opt / ACFO</th>
<th>Dep / Opt / ACEF</th>
<th>Dep / Pre / ACFO</th>
<th>Dep / Pre / ACEF</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>666</td>
<td>669</td>
<td>4755</td>
<td>1015</td>
<td>548</td>
<td>549</td>
<td>511</td>
<td>512</td>
</tr>
<tr>
<td>2</td>
<td>1113</td>
<td>1114</td>
<td>1088</td>
<td>1151</td>
<td>913</td>
<td>911</td>
<td>591</td>
<td>591</td>
</tr>
<tr>
<td>3</td>
<td>747</td>
<td>747</td>
<td>152</td>
<td>149</td>
<td>254</td>
<td>250</td>
<td>167</td>
<td>166</td>
</tr>
<tr>
<td>4</td>
<td>750</td>
<td>750</td>
<td>62</td>
<td>61</td>
<td>768</td>
<td>652</td>
<td>68</td>
<td>68</td>
</tr>
<tr>
<td>5</td>
<td>793</td>
<td>792</td>
<td>271</td>
<td>270</td>
<td>978</td>
<td>674</td>
<td>615</td>
<td>615</td>
</tr>
<tr>
<td>6</td>
<td>842</td>
<td>842</td>
<td>94</td>
<td>93</td>
<td>275</td>
<td>274</td>
<td>234</td>
<td>236</td>
</tr>
<tr>
<td>7</td>
<td>914</td>
<td>912</td>
<td>223</td>
<td>223</td>
<td>760</td>
<td>760</td>
<td>844</td>
<td>843</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>655</td>
<td>656</td>
<td>2647</td>
<td>1095</td>
<td>708</td>
<td>708</td>
<td>301</td>
<td>301</td>
</tr>
<tr>
<td>Average</td>
<td>810</td>
<td>810.25</td>
<td>1161.5</td>
<td>507.125</td>
<td>560.5</td>
<td>597.25</td>
<td>416.375</td>
<td>416.5</td>
</tr>
<tr>
<td>St.Dev.</td>
<td>149.41</td>
<td>148.96</td>
<td>1697.34</td>
<td>486.07</td>
<td>270.97</td>
<td>230.99</td>
<td>265.08</td>
<td>264.84</td>
</tr>
<tr>
<td>Min</td>
<td>655</td>
<td>656</td>
<td>62</td>
<td>61</td>
<td>254</td>
<td>250</td>
<td>68</td>
<td>68</td>
</tr>
<tr>
<td>Max</td>
<td>1113</td>
<td>1114</td>
<td>4755</td>
<td>1151</td>
<td>978</td>
<td>911</td>
<td>844</td>
<td>843</td>
</tr>
</tbody>
</table>

We then performed a two-level factorial analysis, excluding instances 8 and 9, in order to see if any factor has a (statistically) significant effect on this time performance measure. The route generation discipline turns out to be the only significant (at 93% confidence) factor influencing the time to the first feasible solution. The magnitude of the estimated effect was -302 seconds, indicating that dependent (rather than independent) route generation, i.e., adding sets of disjoint routes at each route generation iteration, reduces the time to find the first feasible solution by about 5 minutes on average. The other factors, the stopping rule and branching scheme, had no significant effect on this time.
We then performed an analysis for the times to the best-known solution, which are reported in Table 3. The time to the best solution is not the total solution time since the confirmation of optimality may take additional time.

Table 3. Time (in seconds) to the best solution.

<table>
<thead>
<tr>
<th>Problem Instance</th>
<th>Ind / Opt / ACFO</th>
<th>Ind / Opt / ACEF</th>
<th>Ind / Pre / ACFO</th>
<th>Ind / Pre / ACEF</th>
<th>Dep / Opt / ACFO</th>
<th>Dep / Opt / ACEF</th>
<th>Dep / Pre / ACFO</th>
<th>Dep / Pre / ACEF</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>666</td>
<td>669</td>
<td>4755</td>
<td>1024</td>
<td>548</td>
<td>549</td>
<td>511</td>
<td>512</td>
</tr>
<tr>
<td>2</td>
<td>1292</td>
<td>1293</td>
<td>6755</td>
<td>1151</td>
<td>5061</td>
<td>4835</td>
<td>597</td>
<td>597</td>
</tr>
<tr>
<td>3</td>
<td>747</td>
<td>747</td>
<td>450</td>
<td>448</td>
<td>254</td>
<td>250</td>
<td>167</td>
<td>166</td>
</tr>
<tr>
<td>4</td>
<td>776</td>
<td>776</td>
<td>73</td>
<td>71</td>
<td>866</td>
<td>860</td>
<td>68</td>
<td>68</td>
</tr>
<tr>
<td>5</td>
<td>793</td>
<td>792</td>
<td>271</td>
<td>270</td>
<td>1170</td>
<td>674</td>
<td>648</td>
<td>668</td>
</tr>
<tr>
<td>6</td>
<td>842</td>
<td>841</td>
<td>344</td>
<td>343</td>
<td>275</td>
<td>274</td>
<td>234</td>
<td>234</td>
</tr>
<tr>
<td>7</td>
<td>914</td>
<td>912</td>
<td>634</td>
<td>635</td>
<td>760</td>
<td>760</td>
<td>844</td>
<td>843</td>
</tr>
<tr>
<td>8</td>
<td>826</td>
<td>690</td>
<td>547</td>
<td>546</td>
<td>635</td>
<td>637</td>
<td>696</td>
<td>696</td>
</tr>
<tr>
<td>9</td>
<td>609</td>
<td>613</td>
<td>489</td>
<td>489</td>
<td>644</td>
<td>645</td>
<td>326</td>
<td>326</td>
</tr>
<tr>
<td>10</td>
<td>655</td>
<td>656</td>
<td>5688</td>
<td>3758</td>
<td>729</td>
<td>730</td>
<td>301</td>
<td>301</td>
</tr>
<tr>
<td>Average</td>
<td>812</td>
<td>798.9</td>
<td>2000.6</td>
<td>873.5</td>
<td>1094.2</td>
<td>1000.6</td>
<td>441.2</td>
<td>441.1</td>
</tr>
<tr>
<td>St.Dev.</td>
<td>192.92</td>
<td>195.98</td>
<td>2622.75</td>
<td>1064.72</td>
<td>1419.21</td>
<td>1358.75</td>
<td>257.78</td>
<td>257.36</td>
</tr>
<tr>
<td>Min</td>
<td>609</td>
<td>613</td>
<td>73</td>
<td>71</td>
<td>254</td>
<td>250</td>
<td>68</td>
<td>68</td>
</tr>
<tr>
<td>Max</td>
<td>1292</td>
<td>1293</td>
<td>6755</td>
<td>3758</td>
<td>5061</td>
<td>4835</td>
<td>844</td>
<td>843</td>
</tr>
</tbody>
</table>

It is clear from Table 3 (and validated through statistical analysis) that the use of dependent route generation along with premature stopping gives the best result for the time to the best solution on the average. In other words, premature stopping quickly adds sets of disjoint routes into the problem so that the best-known solution is reached quicker than by the other methods. None of the other main factors or their interaction effects are statistically significant to report.

Lastly, we analyze the total solution times as shown in Table 4. In some instances (i.e. instance 2), the execution of the program is terminated after 2 hours (7,200 seconds) if a feasible solution is at hand. Therefore, the maximum execution time in Table 4 is about 7200 seconds.

Once again, the use of dependent generation along with premature stopping appears to be the best choice, which is more apparent (although not statistically validated) once problem
instance 2 is eliminated from the analysis. (This instance is neglected in calculation of the average, standard deviation, minimum, and maximum values in Table 4.) The intuition might be that a quick generation of a set of disjoint routes leads to optimality quicker. Premature stopping does not only generate a larger route pool within the same time period as compared to optimal stopping, it is also advantageous if the optimal route is the first generated route. In addition, the use of dependent route generation keeps the size of the route pool under the control as opposed to the independent route generation.

Table 4. Total Solution Time (seconds).

<table>
<thead>
<tr>
<th>Problem Instance</th>
<th>Ind / Opt / ACFO</th>
<th>Ind / Opt / ACEF</th>
<th>Ind / Pre / ACFO</th>
<th>Ind / Pre / ACEF</th>
<th>Dep / Opt / ACFO</th>
<th>Dep / Opt / ACEF</th>
<th>Dep / Pre / ACFO</th>
<th>Dep / Pre / ACEF</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>681</td>
<td>684</td>
<td>7203</td>
<td>7205</td>
<td>592</td>
<td>593</td>
<td>548</td>
<td>549</td>
</tr>
<tr>
<td>2</td>
<td>7209</td>
<td>7210</td>
<td>7221</td>
<td>7207</td>
<td>7213</td>
<td>7212</td>
<td>7204</td>
<td>7204</td>
</tr>
<tr>
<td>3</td>
<td>774</td>
<td>776</td>
<td>491</td>
<td>488</td>
<td>282</td>
<td>279</td>
<td>204</td>
<td>204</td>
</tr>
<tr>
<td>4</td>
<td>792</td>
<td>789</td>
<td>150</td>
<td>151</td>
<td>878</td>
<td>7205</td>
<td>125</td>
<td>124</td>
</tr>
<tr>
<td>5</td>
<td>805</td>
<td>805</td>
<td>294</td>
<td>294</td>
<td>7209</td>
<td>683</td>
<td>694</td>
<td>694</td>
</tr>
<tr>
<td>6</td>
<td>854</td>
<td>854</td>
<td>392</td>
<td>391</td>
<td>328</td>
<td>328</td>
<td>271</td>
<td>271</td>
</tr>
<tr>
<td>7</td>
<td>928</td>
<td>928</td>
<td>657</td>
<td>655</td>
<td>800</td>
<td>801</td>
<td>885</td>
<td>885</td>
</tr>
<tr>
<td>8</td>
<td>7210</td>
<td>696</td>
<td>556</td>
<td>554</td>
<td>645</td>
<td>646</td>
<td>714</td>
<td>712</td>
</tr>
<tr>
<td>9</td>
<td>641</td>
<td>641</td>
<td>505</td>
<td>505</td>
<td>653</td>
<td>652</td>
<td>375</td>
<td>375</td>
</tr>
<tr>
<td>10</td>
<td>682</td>
<td>681</td>
<td>7214</td>
<td>7214</td>
<td>765</td>
<td>764</td>
<td>336</td>
<td>333</td>
</tr>
<tr>
<td>Average</td>
<td>1485</td>
<td>762</td>
<td>1940</td>
<td>1940</td>
<td>1350</td>
<td>1328</td>
<td>461</td>
<td>461</td>
</tr>
<tr>
<td>St.Dev.</td>
<td>2149</td>
<td>94</td>
<td>2990</td>
<td>2991</td>
<td>2206</td>
<td>2211</td>
<td>261</td>
<td>261</td>
</tr>
<tr>
<td>Min</td>
<td>641</td>
<td>641</td>
<td>150</td>
<td>151</td>
<td>282</td>
<td>279</td>
<td>125</td>
<td>124</td>
</tr>
<tr>
<td>Max</td>
<td>7210</td>
<td>928</td>
<td>7214</td>
<td>7214</td>
<td>7209</td>
<td>7205</td>
<td>885</td>
<td>885</td>
</tr>
</tbody>
</table>

We also looked into the problem instances where the best and the worst time performances were observed for the total solution time in order to gain some additional insight. Since there are many parameters (starting city of each aircraft, the structure of network, the concentration of maintenance type on that particular day, resource availability for both man-hours and slots, etc.) that might affect the solution time performances, it is difficult to gain an insight from two problem instances. Nonetheless, our observations for problem instances 3 and 2 (the best and the
worst total solution time instances, respectively) indicate that the resource availability may affect the performance of the problem solution time significantly. If the resource constraints get tight (e.g., slot availability is about 20% less in problem instance 2 as compared to problem instance 3 although the number of slots required is approximately the same), the solution time naturally grow larger.

Although the time statistics favor the use of dependent route generation with premature stopping, one may question the quality of the solutions. In the next section, this aspect of our computational study is addressed.

6.2.2. Solution Quality Analysis

In this section, we assess the quality of the solutions using the following fours statistics:

i. Integrality gap for optimal solution (best known solution),

ii. Integrality gap for the first feasible solution,

iii. Green time utilization for high-time aircraft,

iv. Number of high-time aircraft with green time utilization over the average utilization.

The results for all solution quality performance measures are summarized in Table 5 for the eight factor-combinations. For each solution quality measure, minimum, average, and maximum values are presented.
Table 5. Solution quality statistics.

<table>
<thead>
<tr>
<th>Disp./Stop./Branch</th>
<th>Integrality Gap for Optimal Solution</th>
<th>Integrality Gap for 1st Incumbent Solution</th>
<th>Average Green Time Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Min</td>
<td>Avg</td>
<td>Max</td>
</tr>
<tr>
<td>Ind/Opt/ACFO</td>
<td>0%</td>
<td>1%</td>
<td>7%</td>
</tr>
<tr>
<td>Ind/Opt/ACEF</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Ind/Pre/ACFO</td>
<td>0%</td>
<td>4%</td>
<td>27%</td>
</tr>
<tr>
<td>Ind/Pre/ACEF</td>
<td>0%</td>
<td>5%</td>
<td>40%</td>
</tr>
<tr>
<td>Dep/Opt/ACFO</td>
<td>0%</td>
<td>0%</td>
<td>4%</td>
</tr>
<tr>
<td>Dep/Opt/ACEF</td>
<td>0%</td>
<td>0%</td>
<td>2%</td>
</tr>
<tr>
<td>Dep/Pre/ACFO</td>
<td>0%</td>
<td>1%</td>
<td>8%</td>
</tr>
<tr>
<td>Dep/Pre/ACEF</td>
<td>0%</td>
<td>1%</td>
<td>8%</td>
</tr>
</tbody>
</table>

The results indicate that the use of optimal (rather than premature) stopping gives better quality solutions in terms of the “integrality gap for the optimal solution”. However, the differences between the average integrality gap under optimal and premature stopping are small. The difference is even less significant under the dependent route generation, whereas it is magnified under independent route generation.

There are two significantly large integrality-gap values under independent route generation, 27% and 40%. These problem instances were terminated after 2 hours since feasible solutions were already identified. These two problem instances used the new branching schemes heavily, in 346 nodes and in 200 nodes respectively. In fact, these two problem instances explored the largest number of nodes. Although it is not possible to come up with any statistically significant conclusion, the use of the aircraft specific ending flight leg (ACEF) branching scheme generated better solution quality with a fewer number of new branching scheme executions within the same time frame as compared to aircraft specific follow-on branching scheme.

On the whole, the integrality gap for the optimal solution does not vary significantly from condition to condition on the average. The results for the “integrality gap for first incumbent
(feasible) solution” are similar. The maximum value of 49% is due to the initial feasible solution that is found by the initial solution heuristics in the algorithm.

The observations for the previous two solution quality measures are also true for “average green time utilizations solution” quality measure. Green time utilization is simply the percentage of legal remaining flying hours utilized by an aircraft in a particular solution. Average green time utilization is the average of green time utilization percentages for a particular problem instance. It is clear from Table 5 that solution qualities are not different from one condition to other. In other words, the green time utilization on the average is almost identical for all conditions.

In summary, the algorithm reaches the same solution quality on the average under each condition. The main difference is the time used to reach these solutions. This observation highlights the significance of the solution time results.

Table 6. Other solution statistics.

<table>
<thead>
<tr>
<th>Route/Stop/Branch</th>
<th># of Nodes Searched</th>
<th>% of Solution Time used for Route Generation</th>
<th># of Route Generated</th>
<th>Branching Scheme Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Min</td>
<td>Avg</td>
<td>Max</td>
<td>Min</td>
</tr>
<tr>
<td>Ind/Opt/ACFO</td>
<td>7</td>
<td>248.2</td>
<td>1835</td>
<td>50%</td>
</tr>
<tr>
<td>Ind/Opt/ACEF</td>
<td>7</td>
<td>69.5</td>
<td>463</td>
<td>79%</td>
</tr>
<tr>
<td>Ind/Pre/ACFO</td>
<td>1</td>
<td>357.3</td>
<td>2066</td>
<td>45%</td>
</tr>
<tr>
<td>Ind/Pre/ACEF</td>
<td>1</td>
<td>371.9</td>
<td>1949</td>
<td>44%</td>
</tr>
<tr>
<td>Dep/Opt/ACFO</td>
<td>3</td>
<td>161.7</td>
<td>1118</td>
<td>51%</td>
</tr>
<tr>
<td>Dep/Opt/ACEF</td>
<td>3</td>
<td>94.3</td>
<td>416</td>
<td>76%</td>
</tr>
<tr>
<td>Dep/Pre/ACFO</td>
<td>1</td>
<td>134.8</td>
<td>1201</td>
<td>57%</td>
</tr>
<tr>
<td>Dep/Pre/ACEF</td>
<td>1</td>
<td>134.6</td>
<td>1199</td>
<td>58%</td>
</tr>
</tbody>
</table>

In Table 6, other solution statistics are reported, including the number of nodes generated, the percentage of solution time used for route generation, the number of routes generated, and branching scheme statistics. There are two branching scheme statistics, maximum number of
nodes where the new branching scheme is used and the number of problem instances where the new branching scheme is used.

Dependent route generation results in fewer nodes and routes than independent route generation for all minimum, maximum, and average values as seen in Table 6. This is intuitive since dependent route generation adds sets of disjoint routes at each node whereas independent route generation allows the additions of identical routes for different aircraft. The use of dependent route generation increases the likelihood of finding a feasible solution, and accordingly, reduces the need to generate more nodes (and subsequent routes) to search as opposed to the use of independent route generation.

The use of optimal stopping has also the same effect as the use of dependent route generation. The two largest numbers of nodes, 2066 and 1949, occurred under the conditions where independent route generation is used along with premature stops. These two conditions also correspond to the same instance where integrality gaps are the poorest and the new branching schemes are employed multiple times.

It is also apparent from Table 6 that 75-80% of the solution time is used to generate routes on the average. Considering the number of pricing sub-problems (32) and the run time complexity of dynamic programming, this observation is not surprising. However, it is also obvious from Table 6 that there are some problem instances, for which CPLEX and route generation times are equally split.

Although the choice of new branching scheme, either aircraft specific follow-on (ACFO) or aircraft specific ending flight leg (ACEF), does not have a statistically significant effect on any of the performance measures, time statistics and the number of nodes generated are observed to increase if new branching schemes need to be employed.
The same observations for the number of nodes can be made for the number of routes generated. In case of employing any of the branching schemes, since the number of generated nodes increase, the number of routes generated also appears to increase. There are two conditions, Dep/Pre/ACFO and Dep/Pre/ACEF, where the new branching schemes are never employed. In these cases, both the number of nodes and the number of routes generated were at their smallest values. On the other hand, there are two conditions where new schemes are used 346 and 200 times for the same problem instance. That problem instance is also the one that generated the maximum number of nodes and routes. Under those conditions, the problem had a difficult time in finding an integer solution and was forced to use the new branching schemes heavily. In most cases, it was also observed that the use of the new branching schemes results in either infeasibility or an integer solution that is not better than the existing incumbent. In other words, they facilitated quick fathoming.

In summary, none of the solution quality measures favor any condition over the others; in fact, the solution quality values are found to be fairly close to each other under any of these circumstances. This observation supports the significance of the results obtained from the solution time statistics. Furthermore, the results of solution time statistics are also supported by the observations made from the other statistics like the number of nodes and number of routes generated.

We also performed a scalability analysis. Three different problem sizes, small, medium, and large, were utilized to perform this analysis, as shown in Table 7. For each problem size, 10 problem instances are generated and tested under previously defined eight conditions. Time measurements are all in seconds. Table 7 presents overall average, minimum, and maximum values for the defined performance measures.
Table 7. Scalability analysis.

<table>
<thead>
<tr>
<th>Problems</th>
<th>Problem Size</th>
<th>Time to 1st Feasible</th>
<th>Time to the Best Known Solution</th>
<th>Total Solution time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># of Nodes</td>
<td># of Arcs</td>
<td>Min</td>
<td>Avg</td>
</tr>
<tr>
<td>Small</td>
<td>51</td>
<td>200</td>
<td>1</td>
<td>2.1</td>
</tr>
<tr>
<td>Medium</td>
<td>101</td>
<td>457</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>Large</td>
<td>175</td>
<td>1444</td>
<td>1</td>
<td>537.2</td>
</tr>
</tbody>
</table>

Although the general observation of using dependent route generation along with premature stopping remains consistent throughout this analysis, it gets more magnified when the problem size gets larger. This is to be expected. The other obvious observation from Table 7 is that the larger the problem size gets, the larger the time statistics become. The increase in time performance measures is more apparent in the time to the 1st feasible. In other words, finding the first feasible gets more difficult as the problems size gets larger. However, the average time for this performance measure is still quite fast. This observation also justifies the earlier assumption in Section 6.2.1 that the finding a feasible solution is imperative in case of any disruption.

7. Concluding Remarks

We have described a branch-and-price approach to solving an operational aircraft routing problem that faces airlines on a daily basis. This problem is unique in the literature in that it takes an operational, rather than long-term planning, view of aircraft maintenance routing. Further, we incorporate resource availability constraints into the model, which cannot be ignored when realistically solving an operational problem. The presence of these additional resource constraints requires new modifications to the branch-on, follow-on branching rule frequently used in branch-and-price approaches to similar problems.
We performed a computational study for three factors, route (column) generation discipline, route generation stopping rule, and new branching scheme, in order to gauge their interactive effect on reducing solution time measures, time to the 1st feasible, time to the best-known solution, and total solution time, to this problem. The statistical and numerical results presented in the previous section show that the use of dependent route generation along with premature stopping is the most effective condition. Although this is not statistically significant for the total solution time, we believe that this might be due to either truncations or the number of problem instances used. Nonetheless, it is clear that the use of dependent route generation together with premature stopping has a promising decreasing effect on all solution time measure. This might intuitively be since this combination adds sets of disjoint routes quicker than the other conditions. Moreover, each addition intends to add a feasible solution if one exists. The use of new branching schemes shows no effect on solution time measures.

We then analyzed the solution quality measures, integrality gap for the optimal solution, integrality gap for the first feasible, and average green time utilization for high-aircraft. No condition showed significant differences in terms of solution quality. Under all conditions, the average solution quality is approximately identical. However, optimal stopping during the route generation appear to generate slightly better solution quality. This was more apparent under independent route generation. Nonetheless, the solution qualities were not so much different from condition to condition, on the average. This conclusion magnifies the results obtained for the solution times.

We further analyzed some other performance statistics, minimum, maximum, and average for the number of nodes generated in the B&B tree, the number of routes generated, percentage of solution time used for route generation along with the maximum number of new
branching scheme deployments and the number of instances for which the new branching scheme is used. The results suggest that the known branch-on follow-on branching scheme can be used as a heuristic in most instances.

The scalability analysis in Section 6.2.2 showed that detection of the first feasible solution gets more difficult as the problem size gets larger. Also, the strategy of using dependent route generation along with the premature stopping is best regardless of the problem size, though the effect becomes more apparent as the problem size gets larger.

Acknowledgements

The authors would like to thank the Editor and two anonymous referees, whose comments on an earlier draft of this paper significantly improved the work and the clarity of its presentation.

References


