An Efficient Airline Re-Fleeting Model for
the Incremental Modification of Planned
Fleet Assignments

AHMAD I. JARRAH
Transport Dynamics, Inc., Princeton, New Jersey 08540

JON GOODSTEIN AND RAM NARASIMHAN
United Airlines, Corporate Research and Development, WHQKB, 1200 E. Algonquin Rd., Elk Grove, Illinois 60007

Airlines typically manage their annual business cycle by subdividing the year into a sequence of scheduling periods that span about a month each. Fleet assignment represents an important step in the planning process for each new scheduling period and is usually undertaken using computer-based optimization models. Once an initial fleet assignment solution is achieved and before “freezing” the assignments, planners spend significant effort in analyzing, modifying, and committing the individual components of the solution throughout the flight network. This manual step results in local modifications to the initial solution, and is necessary to reflect business judgment calls that cannot be captured by the optimization model. In addition, planners find it imperative to modify the initial fleet assignment solution to react to inevitable changes to the planning environment related to the planned schedule, demand forecast, number of available aircraft, crew staffing levels, and a host of other scheduling constraints. The above-described process of incrementally fine-tuning and modifying the initial fleet assignment solution is referred to as re-fleeting. It is characterized by (1) the requirement of quick solution time to allow frequent re-fleeting exercises, (2) the need for multiple alternative high-quality solutions to choose from, and (3) the constraint that the new solution differs from the prior one in a controlled and limited fashion. We demonstrate in this paper that global fleet assignment model formulations can be used to address the re-fleeting problem in an effective fashion.

Airlines typically manage their annual business cycle by subdividing the year into a sequence of scheduling periods that span about a month each. Fleet assignment represents an important step in the process of planning for a new scheduling period. The objective of this step is to match available seat capacity to profitable passenger demand in a fashion that seeks to maximize the overall network contribution. Figure 1 places fleet assignment in the overall schedule-planning context. The first step in the process is schedule construction which, at United Airlines (UA), is typically undertaken 10–14 weeks prior to the start date of the schedule and which focuses on specifying the flights to be flown along with their frequencies. An important consideration at this stage is the incorporation of profitable non-stop and one-stop services into the schedule.

The next planning step is fleet assignment (7–10 weeks prior to schedule start date) followed by through assignment (6–7 weeks prior to schedule start) whose objective is to specify the one-stop services that use the same aircraft (known as throughs), hence allowing passengers to stay on board. Special attention is paid to throughs because of their marketing advantage over other one-stop services.

Next, maintenance-feasible aircraft routings are generated for each of the aircraft used in the schedule. The motivation here is to ensure, at a planning
level, that the aircraft perform their sequence of assigned flights while receiving the required scheduled maintenance checks at the right time and place. The final two planning steps involve the building of in-flight and cockpit crew pairings (or several-days-long work patterns), and the stringing of the pairings into several-week-long assignments often called bidlines or rosters.

The fleet assignment step itself is usually done in two phases:

1. An initial fleet assignment solution is generated, typically, using a fleet assignment model. The solution obtained is required to be feasible; i.e., each fleet type in the solution is capable of performing its flight missions. Factors involved here include the fleet types’ flying ranges, marketing constraints, noise restrictions at certain airports, and existence of sufficient maintenance opportunities for the aircraft. Another important goal at this stage is to make efficient usage of the seat capacities of the available fleet types by providing the appropriate capacity for the various flights. In general, fleet types with larger seat capacities are matched with flights that have attractive fares and high passenger volumes, whereas fleet types with smaller seat capacities are used for the lower-demand flights.

2. Subsequently, planners spend significant effort in analyzing, modifying, and committing the individual components of the solution throughout the flight network. This manual step results in local modifications to the initial solution, and is necessary to reflect business judgment calls that cannot be captured by the optimization model (some of these issues are discussed in the next section). In addition, planners find it imperative to modify the initial fleet assignment solution to react to changes in the assumptions made about the planning environment. The assumptions include those related to the planned schedule, demand forecasts, number of available aircraft, crew staffing levels, and a host of other scheduling constraints. Note that changes in the planning environment may continue to occur up to the start date of the schedule and that any re-fleeting work done beyond week seven (see Figure 1) may also result in the need to re-work parts of the through assignment, aircraft routing, and crew scheduling solutions.

The above-described process of incrementally fine-tuning and modifying the initial fleet assignment solution is referred to as re-fleeting and is the focus of this paper. At UA the re-fleeting problem is characterized by the following requirements:

1. Quick solution time to allow frequent and interactive re-fleeting exercises,
2. Controlled and limited differences in the new solution compared to the prior one to allow a smooth transitioning of the current fleet assignment into a final “closed” solution, and
3. Multiple re-fleeting solutions per re-fleeting scenario thus allowing the users to compare and choose from among alternative solutions the most “appropriate” solution (this refers, once more, to business judgment calls that cannot be captured by the optimization models and that are discussed in the next section).

In this paper we achieve the following:

1. We demonstrate the viability of the usage of a multi-commodity integer flow network with side constraints to address the re-fleeting problem. This formulation has been used previously in fleet assignment models (see HANE et al., 1995). However, we show that the same formulation can...
be used to comprehensively address the questions that arise during re-fleeting with very quick turn-around times, while limiting the number of changes to the original fleet assignments.

2. We describe the preprocessing steps that were used to reduce the problem size and detect infeasibilities early on.

3. We present five modules that we developed for the re-fleeting model to allow users to run different classes of re-fleeting problems. All the modules use the same basic model; however, they represent a layer that shields the users from having to select tens of parameters to achieve the desired effect from the model. Table I lists the modules along with their purpose and scenarios that result in their usage. The scenarios described are real ones accumulated from interactions with the users of the model.

4. We present an efficient solution scheme that seeks to find the \( \kappa \) best solutions for a re-fleeting question posed by any one of the above-mentioned modules. The algorithm does not seek to generate the \( \kappa \) best provably optimal solutions; rather solutions with sufficiently small optimality gap are deemed acceptable. We demonstrate that, for typical values of \( \kappa (\leq 7) \), the solutions can be obtained very efficiently (typically in less than five minutes) for real-life instances of the problem.

Although the context of the re-fleeting model is a planning rather than operational one, the very fast response times we experienced with the model indicates that there is a potential for extending the model for usage in an operational environment.

1. PREVIOUS RELATED WORK

The fleet assignment problem has received significant attention, and models have been successfully developed to solve large-scale instances of the problem for many airlines around the world (ABARA, 1989; BARNHART et al., 1997; CLARKE et al., 1996; DASKIN and PANAYOTOPoulos, 1989; DESROSiers et al., 1995; Hane et al., 1995; and SUBRAMANIAM et al., 1994). Usually, the problem is modeled for a typical day that is assumed to repeat indefinitely. This assumption is close enough to reality for the domestic networks of U.S. carriers, where the schedule is almost identical for each day of the week during a scheduling period. For international schedules, it is usually necessary to consider a typical week, rather than one day, to accurately capture the problem (Barnhart et al., 1997).

In contrast, very little attention has been paid to the re-fleeting problem. BERGE and HOPPERSTAD (1993) presented two heuristic solution approaches for solving multi-day fleet assignment problems. Conceivably, these heuristics, being fast, could be adapted to address the re-fleeting problem. However, the first heuristic (DELPRO) has the shortcoming that it is local in nature because it finds swaps in a restricted class. In the second approach (SMCF, or sequential min-cost flow), it is not possible to model all the side feasibility constraints required in the fleet assignment solution, such as those related to maintenance and crew rest (see Clarke et al., 1996). Furthermore, the approach will generate solutions that may differ significantly from the original solution.

TALLURI (1996) discussed the following problem: given a balanced daily flight schedule, if a flight’s currently assigned fleet type is to be swapped to another fleet type, what is the least costly (or most profitable) way to achieve the swap? To answer the question, a flight network is first constructed for the two fleet types involved (similar to the network in Hane et al., 1995). Subsequently, the flight arcs corresponding to the swapped-to fleet type are reversed, and a shortest path algorithm is used to find the best set of swaps that involve \( x \) overnights, where \( x \) is defined ahead of time. Typical values for \( x \) are zero, for a same-day sequence of swaps, and one or two for sequences of swaps that involve one or two overnights, respectively.

Although the above approach yields optimal solutions quickly for problems involving two fleet types, it suffers from several shortcomings that precluded its adoption for addressing the re-fleeting needs at UA. First, the approach only addresses one instance of the re-fleeting problem, namely, the re-balancing of a schedule when the fleet type is modified for an individual flight. In addition, only two fleet types are considered in the proposed search for swaps for balancing the schedule. This overlooks potential superior swaps that may involve three or more fleet types.

Furthermore, the users at UA’s scheduling department expressed a preference for being presented with multiple “best” solutions for each re-fleeting problem that they want to solve. The rationale behind this is that, although a model-generated solution will meet all the model’s constraints, there are certain considerations that cannot be captured in the current fleet assignment formulations, as well as in the swapping approach of Talluri (1996). Generating several alternative high-quality solutions gives the schedulers more options to assess with respect to those unmodeled considerations. Although the detailed discussion of such consider-
### Modules of the Re-fleeting Model

<table>
<thead>
<tr>
<th>Module Name</th>
<th>Purpose</th>
<th>Scenarios of Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Popping</td>
<td>Finds the least costly way to remove a user-specified number of aircraft from the schedule</td>
<td>1. A DC10-10 needs to be removed to undergo heavy maintenance. Accordingly, it is desired to pop a DC10-10 from the schedule, while considering swapping with and/or canceling flights flown by 757s, 277s, A320s, A319s, and 737s. The purpose of considering the other fleet types is to avoid the cancellation of the high-demand missions typically flown by the DC10-10. 2. A 767 currently used for domestic flights is needed to fly international missions; hence, it is necessary to cancel some domestic flights. It is desired to pop a 767 from the domestic schedule, while considering swapping with and canceling flights flown by the 757s, 277s, A320s, A319s, and 737s.</td>
</tr>
<tr>
<td>Change-of-Gauge</td>
<td>Moves a user-specified number of aircraft from one group of fleet types to another group of fleet types such that the incremental contribution is maximized</td>
<td>1. Toward the middle of the planning period, it is announced that the newly purchased A320 has been received and can be made ready to fly next month (one month sooner than planned). Hence, it is possible to retire a 727 to use the A320 in its place. 2. A 757 must undergo heavy maintenance (several months in duration), and a 737 has just completed heavy maintenance. It is necessary to replace the 757 with a 737. Other fleets should be considered in the analysis because it is not allowed for the 737 equipment type to fly 757 missions. 3. A 757 and 767 have just completed maintenance, one DC10-10 must undergo heavy maintenance, and we’re retiring another DC10-10. Hence, it is necessary to replace two DC10-10s with a 757 and a 767.</td>
</tr>
<tr>
<td>Swapping</td>
<td>Swaps fleet assignments within a user-specified group of fleet types to maximize incremental contribution</td>
<td>1. A new maintenance requirement is added for the 727 aircraft. It is desired to re-fleet the schedule without affecting the solution for the wide-body aircraft. Accordingly, a run is conducted to look for profitable swaps between 737s, 727s, and A320s. 2. It is necessary to modify the fleet assignment solution as a result of modified demand and fare forecasts. As it is late in the planning process, the swapping module is used to modify the solution while restricting the number of changes in the solution.</td>
</tr>
<tr>
<td>Utilization</td>
<td>Shifts block time (i.e., flying hours) between two groups of fleet types while maximizing incremental revenue</td>
<td>1. The current fleet assignment solution results in a utilization of Airbus pilots that exceeds the latest projected availability of these pilots. Hence, it is necessary to Shift 20 hours per day from the A320s to the 727s, 737s, and 757s. 2. Similarly, the 757 pilots are underutilized in the current fleet assignment solution. It is desired to shift 10 hours per day from the A320s and 727s to the 757s to remedy the situation.</td>
</tr>
</tbody>
</table>
| Balancing    | Finds the most profitable set of swaps that will balance the schedule when the fleet assignments are changed for a user-specified set of flights | 1. For competitive and marketing reasons, it is necessary to assign an A320 to the 7:00 am ORD DCA flight in lieu of the 737 currently assigned. 2. The current fleet assignment solution stipulates the following assignments for the departures out of Seattle at 6:00 and 8:00 a.m.:  

<table>
<thead>
<tr>
<th>Flight Origin</th>
<th>Flight Destination</th>
<th>Flight Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEA</td>
<td>ORD</td>
<td>0600</td>
</tr>
<tr>
<td>SEA</td>
<td>DEN</td>
<td>0800</td>
</tr>
</tbody>
</table>

It is noticed that the 0600 originating flight causes an illegal overnight for the 737 crew and that the 0800 departure would have been legal for that same crew. Accordingly, it is desired to know the best way to switch the equipment types on these two flights. 3. The scenario in 2 also arises if it is desired to create a maintenance layover of sufficient duration at Seattle for the 737 fleet. |
ations is beyond the scope of this paper, we briefly discuss two of them:

1. Current Fleet Assignment Models assume the departure and arrival times of flights to be fixed; in reality, schedulers have the latitude of slightly modifying these times to get improved fleet assignment solutions (see Rexing et al. (1998) for a recent paper on this aspect of the problem).

2. All current Fleet Assignment Models assume as input a passenger demand and an average fare for each flight. In reality, demands and fares are associated with services rather than with flights. A service could be a flight or a connection consisting of two or more flights. Assigning fleet types to flights without analyzing the impact on the connecting passengers might yield less than desirable solutions in certain instances (see Kniker (1998) for a recent research effort on the topic).

In this paper, we demonstrate the applicability of a multi-commodity integer flow network with side constraints to address the re-fleeting problem. The formulation gives the users the ability to consider multiple fleet types, and the latitude to solve re-fleeting problems other than that of re-balancing a schedule due to a change in fleet type for one flight. The model also uses constraints that limit the number of changes to the original solution. We also designed the solver of the model to give $\kappa$ solutions with some pre-defined tolerance from optimality, where $\kappa$ is user-defined.

### 2. MODEL FORMULATION

A TIME–SPACE NETWORK is constructed to depict the flight operations undertaken by each one of the fleet types for a typical day in the scheduling period. Side constraints are then added to reflect the multi-commodity nature of the problem and to capture the necessary additional constraints.

#### 2.1 Underlying Network Structure

In Figure 2, the $x$-axis is used to identify the various cities served by the fleet type under consideration, whereas the $y$-axis depicts time. There are three types of arcs: flight arcs for transferring aircraft from their departure cities to their arrival cities, ground arcs for transferring aircraft through the time dimension at the same location, and overnight arcs, which are similar to ground arcs except that they loop from the end of the day at a station back to the beginning of the day to ensure balance in the number of aircraft at each station.

Note that the only flight arcs that appear for a fleet type are those flights that the fleet type is capable of operating. Nodes are used to identify the departure times of flights and the ready times of the arriving aircraft, where the ready time of an aircraft is defined as its arrival time plus a minimum ground time, which is needed to prepare the aircraft for its subsequent departure.

We used two techniques to reduce the problem size: node aggregation and island analysis. Hane et al. (1995) reported the reduction in size that was achieved using these two techniques in their implementation of the fleet assignment model. Figure 3 shows how the node aggregation technique is used. Here, the flights are scanned in a chronological order, and new nodes are formed only when an arrival follows a departure. This simple scheme significantly reduces the problem size without precluding any potential fleet assignments.

Further reduction in the problem size can be achieved by means of island analysis. The underlying principle here is that, at most stations, airlines do not typically retain aircraft on the ground on a continuous basis. In other words, at certain points in time during the day, no aircraft will remain on the ground. Of course, this does not apply to all stations. In particular, airlines often retain extra aircraft at their hubs (major connection stations) and at some
of their busy spokes (stations with no connecting traffic). The same is true for maintenance stations where aircraft undergo maintenance on a continual basis.

As in Hane et al. (1995), to achieve effective reduction in problem size, we apply island analysis to the schedule. Figures 4 and 5 demonstrate the concepts involved. Here, the input flight schedule is depicted schematically for a certain station. The shown pattern of arrivals and departures is such that no aircraft remains on the ground during two periods of time within the day. This results in fragmenting the station into two independent mini-stations with zero starting and ending aircraft count on the ground. These islands are then treated separately during the generation of the time–space network. Note that some of the flights in the schedule do not appear in the time–space network; this implies that these flights cannot be flown by the fleet type for which the network is constructed. This often results in further reduction in problem size; for example, it is evident from the time–space network that the last arrival cannot take a non-zero value.

Fig. 4. For stations that are neither hubs nor maintenance bases, the activity pattern can often be divided into independent islands with self-contained flight activities.

Fig. 5. At a station $s$ that is fragmented into islands, if an arrival $i$ from island $u$ is used for a departure $j$ from a different island $v$, then one aircraft is required to stay on the ground throughout the day.

2.2 Mathematical Statement of the Basic Model

We start by defining the indices, sets, parameters, and variables used in the model:

Sets and Indices:

i, j = indices for flights
k = index for fleet types
s = index for stations
n = index for nodes

$P^k(n)$ = predecessor node of node $n$ in network for fleet $k$

$O(i)$ = fleet type originally assigned to flight $i$ in the input schedule

Parameters:

$\Delta$ = total number of fleet changes that are allowed

$\alpha^k(i)$ = ready time for aircraft type $k$ undertaking flight $i$; equals the arrival time of $i$ plus a minimum ground time, which varies by fleet type and is needed to prepare the aircraft for its next departure

$\delta(j)$ = departure time of flight $j$

$x^k_i$ = number of aircraft of fleet type $k$

$\rho^k_i$ = incremental contribution involved in switching the fleet type of flight $i$ to $k$; equals 0 if $k = o(i)$

$\tau^0$ = arbitrary point in time chosen for counting the aircraft used; in our model we chose 3 a.m. Central time

Sets:

$K$ = set of fleet types
$S$ = set of stations
$F$ = set of flights
$L^k$ = set of flights that can be flown by fleet type $k$

$N^k(s)$ = set of nodes at station $s$ in the time–space network of fleet type $k$

$I^k(n)$ = set of flights that are incoming to node $n$ in the time–space network of fleet type $k$

$O^k(n)$ = set of flights that are outgoing from node $n$ in the time–space network of fleet type $k$

$D^k(\tau^0)$ = set of nodes in the time–space network of fleet type $k$ that start before and finish after $\tau^0$

$G^k(\tau^0)$ = set of ground arcs in the time–space network of fleet type $k$ that start before and finish after $\tau^0$

$F^k(\tau^0)$ = set of flight arcs in the time–space network of fleet type $k$ that start before and finish after $\tau^0$
Decision Variables:

\[ x_i^k = 1 \text{ if fleet } k \text{ is assigned to flight } i; \quad 0, \text{ otherwise} \]

\[ g_n^k = \text{flow on ground arc out of node } n \text{ in the times–space network corresponding to fleet } k; \]

restricted to non-negative values.

The formulation can now be written as

\[ \text{Maximize} \sum_{k \in K} \sum_{i \in L_k} \rho_i^k x_i^k \quad (1) \]

subject to

\[ \sum_{i \in P(n)} x_i^k - \sum_{j \in O(n)} x_j^k + g_{p(n)}^k - g_n^k = 0, \quad \forall k \in K, \quad \forall s \in S, \quad \forall n \in N^k(s) \quad (2) \]

\[ \sum_{i \in K} x_i^k = 1, \quad \forall i \in F \quad (3) \]

\[ \sum_{n \in D(n, p^k)} \left( g_{p(n)}^k + \sum_{i \in P(n), k \in C_p(i)} x_i^k - \sum_{j \in O(n), k \in C_p(i)} x_j^k \right) \]

\[ + \sum_{n \in G(n, p^k)} g_n^k + \sum_{i \in F(n, p^k)} x_i^k = \chi^k, \quad \forall k \in K \quad (4) \]

\[ \sum_{k \in K} \sum_{i \in L_k} x_i^k \leq \Delta \quad (5) \]

\[ x_i^k \in \{0, 1\}, \quad \forall k \in K, \quad \forall i \in L^k \quad (6) \]

\[ g_n^k \geq 0, \quad \forall k \in K, \quad \forall s \in S, \quad \forall n \in N^k(s). \quad (7) \]

The objective function (Eq. 1) maximizes the sum of the incremental network contribution resulting from swapping fleet types. The network contribution consists of the revenue accrued by the flights (which depends on seat capacity assigned) minus operating costs. Equation 2 represents the conservation-of-flow relationships at the nodes in the time–space network. Note that many of the ground arcs can be set to zero as a result of island analysis.

Equation 3 ensures that a fleet type is assigned to each flight. Equation 4 counts the aircraft of each fleet type by taking a “snapshot” of the network of each fleet at time \( r^0 \). The first term in the equation captures the aircraft count at time \( r^0 \) for all the nodes that are “sliced” by \( r^0 \). Note that the number of aircraft on the ground at time \( r^0 \) inside such nodes consists of the flow on the incoming ground arc to the node plus the flows on arcs inbound to the node prior to \( r^0 \) minus the flows on arcs outbound from the node prior to \( r^0 \). The second term captures the aircraft count for all ground arcs that are intersected by \( r^0 \), whereas the third term counts the aircraft that are in the air at time \( r^0 \). Equation 5 places a user-specified upper limit on the number of fleet type changes. Equations 6 and 7 give the necessary binary and non-negativity requirements.

2.3 Side Constraints

We implemented several side constraints for the model. The purpose of these constraints is to ensure that the re-fleeting solutions obtained from the model remain feasible with respect to certain operational requirements. Further discussion in the paper beyond this section is based on the model without the side constraints.

Maintenance Opportunities

The incorporation of maintenance checks into fleet management formulations has been previously discussed in Clarke et al. (1996). Here, we describe the maintenance considerations at the planning level in UA’s context. On a periodic basis, aircraft are required to undertake scheduled maintenance checks (typically referred to as the A, B, C, or D checks). When generating fleet assignments, planners build “maintenance opportunities” for each fleet type to facilitate the achievement of these maintenance requirements during the daily operation of the airline.

To generate a maintenance opportunity for a certain fleet type, a planner needs to ensure that an aircraft from that fleet type spends sufficient time on the ground at an appropriately equipped maintenance base for undergoing the required maintenance. To maximize the usage of the aircraft, scheduled maintenance is typically undertaken at night.

At UA, the requirements for each daily maintenance opportunity are specified as follows:

- fleet type,
- list of appropriately-equipped maintenance bases,
- start and end of window for performing maintenance at each of the bases,
- duration of the maintenance check, and
- number of aircraft that are required to undertake maintenance daily for the fleet type under consideration.

To incorporate maintenance checking into the formulation, arcs are first added for each maintenance requirement. In Figure 6, maintenance arcs for a requirement are shown for the corresponding fleet type at an eligible maintenance base. Here, an arc starts at the beginning of the maintenance window and terminates at a node placed at the point in time when the maintenance check gets completed, assuming that the check is started at the beginning of the window. The arc has an upper bound equal to
The following terms are first defined:

\[ i, j = \text{indices for flights or maintenance arcs} \]
\[ I^k(n) = \text{set of flights and maintenance arcs that are} \]
\[ \text{incoming to node } n \text{ in the time–space network of fleet type } k \]
\[ O^k(n) = \text{set of flights and maintenance arcs that are} \]
\[ \text{outgoing from node } n \text{ in the time–space network of fleet type } k \]
\[ x_i^k = 1 \text{ if fleet } k \text{ is assigned to flight or maintenance arc } i; 0, \text{ otherwise} \]

In addition, a lower bound is placed on the flow through the arcs corresponding to each maintenance requirement; thus ensuring that that requirement is met. To present this additional constraint, the following terms are first defined:

\[ r = \text{index for maintenance requirements} \]
\[ R = \text{set of stipulated maintenance requirements} \]
\[ f(r) = \text{fleet type corresponding to maintenance} \]
\[ \text{requirement } r \]
\[ s^{f(r)}(r) = \text{number of aircraft, of fleet type } f(r), \]
\[ \text{required to undergo maintenance for requirement } r \]
\[ M^{f(r)}(r) = \text{set of maintenance arcs, of fleet type } f(r), \]
\[ \text{corresponding to requirement } r \]

The additional side constraint can now be written as

\[ \sum_{i \in M^{f(r)}(r)} x_i^r \geq s^{f(r)}(r) \quad \forall r \in R. \quad (8) \]

**Crew Staffing Levels**

Flight crews can be grouped into categories that are trained to operate one or a few fleet types that are similar in configuration. It is important to ensure that the fleet assignment solutions do not require too few or too many crew resources of a certain category. At the fleet planning stage, crew availability is enforced at an aggregate level by ensuring that the flying time for each crew category falls between pre-specified lower and upper bounds.

The following terms are first defined:

\[ c = \text{index for a crew category} \]
\[ C = \text{set of crew categories} \]
\[ E^c = \text{set of fleet types that crew category } c \text{ can operate} \]
\[ \beta_i^c = \text{block time of flight } i; \text{i.e., the flight’s total duration from its departure gate to its arrival gate} \]
\[ l^c = \text{lower bound on the total block time that crew category } c \text{ can operate} \]
\[ u^c = \text{upper bound on the total block time that crew category } c \text{ can operate} \]

The following is next added to ensure acceptable crew utilization levels:

\[ l^c \leq \sum_{k \in E^c} \sum_{i \in I^c} \beta_i^c x_i^k \leq u^c \quad \forall c \in C. \quad (9) \]

**Noise Restrictions**

Certain older aircraft (referred to as Stage II) are prohibited from operating at certain airports during certain hours of the day. Such restrictions are directly incorporated into the formulation by disallowing flight arcs corresponding to these aircraft from and into the restrictive airports during the curfew periods. Additionally, some airports stipulate their curfews as follows: a maximum percentage of the airline’s arrivals and departures during different periods of the day can use Stage II aircraft. Let \( K^{II} \) denote the set of Stage II fleet types, \( T \) the set of curfews to be enforced, \( W^{'} \) the set of departures/arrivals that correspond to curfew \( t \), and \( \omega^{'} \) the maximum number of Stage II departures/arrivals allowed during curfew \( t \). The curfew constraint can then be simply added as

\[ \sum_{k \in K^{II}} \sum_{i \in W^{'}} x_i^k \leq \omega^{'} \quad \forall t \in T. \quad (10) \]
2.3 Detecting Infeasibilities at the Pre-Processing Stage

It is essential in large-scale mathematical optimization models to detect infeasibilities in the formulation early on, and before the optimization algorithm is called. Tracing the cause of infeasibility after the optimizer is called is an arduous and often unfruitful pursuit. The following steps are undertaken in the pre-processor to detect infeasibilities early on:

1. Each flight is assigned a candidate list that contains the set of fleet types that are capable of performing the flight. This list is formed by first assessing the flight’s noise level, station prohibitions, flight range, and market restrictions. Next, the fleet types that are capable of operating the flight are identified. If for any flight the candidate list is empty, then the problem is infeasible and its requirements must be relaxed.

2. Airlines build into the schedules through flights and forced turns. A through flight consists of two or more consecutive flights that use the same aircraft and bear the same flight number; this enables passengers to remain onboard rather than deplane to look for their departure gates at busy airports. A forced turn is a predetermined assignment of an incoming aircraft to a specific outbound flight due to operational considerations such as those related to maintenance. For both through flights and forced turns, taking the intersection of the initial candidate lists described in (1) tightens the candidate lists of the flights involved. Again, if this results in an empty candidate list for at least one flight, the problem is infeasible.

3. Island analysis often results in generating islands with only one arrival and one departure. In such a case, the scheme described in (2) is applicable for tightening the flights’ candidate lists and re-evaluating feasibility.

4. Finally, for each island, a matching problem is formulated (Figure 7). Here, an inbound flight is connected to an outbound flight with an arc if the ready time of the former is less than or equal to the departure time of the latter, and if the candidate lists of the two flights overlap. The absence of a feasible solution for the matching problem at an island indicates that no feasible assignment exists for the island, and, hence, for the model. The same analysis may be done for stations where island analysis is not conducted, except that, at such stations, the only criterion that must be checked when generating arcs is the overlapping of the candidate lists. The reason for this is that an arrival can supply an aircraft to earlier departures by means of the overnight and ground arcs.

The above simple scheme proved instrumental in detecting infeasibilities early on and before the solver is invoked. Of course, there is no theoretical guarantee that the scheme will always detect infeasible problems. However, in the thousands of runs conducted so far for the model, the solver has never been passed an infeasible formulation.

3. MODULES OF THE RE-FLEETING MODEL

Several modules are defined around the model to simplify its usage. By using these modules, planners can change certain aspects of the problem and then repair the solution while restricting the changes to a pre-specified number. Here, we discuss how the basic formulation of the model can be slightly modified to address the requirements of each of the modules.

3.1 Popping Module

The Popping Module is used to find the least costly way to remove a user-specified number of aircraft from the schedule. The flight segments removed are chosen from a sub-network corresponding to a user-specified group of fleet types. The user may also specify the fleet type(s) associated with some or all of the removed aircraft.

Let $\Omega$ denote the total number of aircraft to be removed or “popped,” and let $\sigma^k$ denote the minimum required reduction in the count of fleet type $k$. Then, $\sum_{k \in K} \sigma^k$ cannot exceed $\Omega$. Also, let $x^i_0$ be a new binary decision variable that equals 1 only if flight $i$ is popped, and let $\rho_i^0$ denote the loss in contribution resulting from the removal of flight $i$. To model the Popping Module, the following constraints are needed in lieu of Eqs. 1, 3, and 4:

\[
\text{Maximize } \sum_{k \in K} \sum_{i \in D} \rho_i^k x^k_i - \sum_{i \in F} \rho_i^0 x^0_i \quad (11)
\]
subject to

\[
\sum_{k \in K} \left( \sum_{n \in D_k(\rho)} g^k_{p(n)} + \sum_{i \in \{P(n)\; | \; (x_i = 1) \}} x_i^k - \sum_{j \in \{O(n)\; | \; (y_j = 0) \}} x_j^k \right)
+ \sum_{n \in G_k(\rho)} g^k_n + \sum_{i \in F(n)} x_i^k = \sum_{k \in K} \chi^k - \Omega \tag{12}
\]

\[
\sum_{n \in D_k(\rho)} \left( g^k_{p(n)} + \sum_{i \in \{P(n)\; | \; (x_i = 0) \}} x_i^k \right)
+ \sum_{n \in G_k(\rho)} g^k_n + \sum_{i \in F(n)} x_i^k \leq \chi^k - \sigma^k, \quad \forall k \in K \tag{13}
\]

\[
\sum_{k \in K} x_i^k + x_j^k = 1 \quad \forall i \in F \tag{14}
\]

The objective function now has a term that reflects the incremental loss in revenue when a flight is removed. Equation 12 is used to reduce the overall count of the involved fleet types by a user-specified parameter, \( \Omega \), and Eq. 13 is used to stipulate the count reduction in specific fleet types. Finally, Eq. 14 allows flights to be removed.

### 3.2 Change-of-Gauge Module

This module allows for the replacement of a certain number of aircraft from one group of fleet types by the same number of aircraft from another group of fleet types such that the incremental contribution is maximized. The name of the module comes from the following: replacing aircraft from certain fleet types by other aircraft from different fleet types is referred to as up- (or down-) gauging when this results in higher (lower) seat capacity.

While up-gauging/down-gauging from one group of fleet types to another, the user may choose whether or not to allow changes to be made to the original network. This provides additional flexibility because it allows modifications to the original fleet assignment while achieving the desired up-gauging or down-gauging. Finally, the users are allowed to specify the number of aircraft to be increased or reduced for individual fleet types within the fleet groups.

Let \( K^1 \) denote the set of fleet types from which aircraft are to be removed, and \( K^2 \) the set of fleet types to which aircraft are to be added. Let \( \Omega \) denote the total number of aircraft to be moved from \( K^1 \) to \( K^2 \). Let \( \sigma^k \) denote the minimum required reduction in the count of fleet type \( k \in K^1 \), and \( \omega^k \) denote the minimum required increase in the count of fleet type \( k \in K^2 \). Once more, \( \sum_{k \in K^1} \sigma^k \leq \Omega \) and \( \sum_{k \in K^2} \omega^k \leq \Omega \). The following equations are substituted for Eq. 4 of the basic model:

\[
\sum_{k \in K^1} \left( \sum_{n \in D_k(\rho)} g^k_{p(n)} + \sum_{i \in \{P(n)\; | \; (x_i = 1) \}} x_i^k - \sum_{j \in \{O(n)\; | \; (y_j = 0) \}} x_j^k \right)
+ \sum_{n \in G_k(\rho)} g^k_n + \sum_{i \in F(n)} x_i^k = \sum_{k \in K} \chi^k - \Omega \tag{15}
\]

\[
\sum_{k \in K^2} \left( \sum_{n \in D_k(\rho)} g^k_{p(n)} + \sum_{i \in \{P(n)\; | \; (x_i = 0) \}} x_i^k \right)
+ \sum_{n \in G_k(\rho)} g^k_n + \sum_{i \in F(n)} x_i^k \leq \chi^k + \Omega \tag{16}
\]

\[
\sum_{n \in D_k(\rho)} \left( g^k_{p(n)} + \sum_{i \in \{P(n)\; | \; (x_i = 0) \}} x_i^k \right)
+ \sum_{n \in G_k(\rho)} g^k_n + \sum_{i \in F(n)} x_i^k \leq \chi^k - \sigma^k, \quad \forall k \in K^1 \tag{17}
\]

\[
\sum_{n \in D_k(\rho)} \left( g^k_{p(n)} + \sum_{i \in \{P(n)\; | \; (x_i = 1) \}} x_i^k \right)
+ \sum_{n \in G_k(\rho)} g^k_n + \sum_{i \in F(n)} x_i^k \geq \chi^k + \omega^k, \quad \forall k \in K^2 \tag{18}
\]

Equation 15 reduces the count of the aircraft in group \( K^1 \) by \( \Omega \), whereas Eq. 16 increases the count of the aircraft in group \( K^2 \) by \( \Omega \). Equation 17 specifies the minimum reduction in aircraft count for each fleet type in group \( K^1 \), whereas Eq. 18 specifies the minimum increase in aircraft count for each fleet type in group \( K^2 \).

### 3.3 Swapping Module

The swapping module allows the user to swap fleet assignments within a user-specified group of fleet types to maximize incremental contribution. This module is identical to the Fleet Assignment Model except that a limit is placed on the number of swaps allowed (Eq. 5).

### 3.4 Utilization Module

The module allows the user to shift block time between two groups of fleet types while maximizing incremental revenue. As defined earlier, the block time of a flight is its total duration from the departure gate to the arrival gate. When the crew staffing level for a group of fleet types is exceeded, the module is used to reduce the required number of crews by switching block time to another group of fleet types.

Let \( K^1 \) denote the set of fleet types from which block time is to be removed, and \( K^2 \) the set of fleet types to which block time is to be added. Let \( \pi^k \) denote the block time originally assigned to fleet
type \( k \), and let \( \beta_i^k \) denote the block time of flight \( i \) when operated by fleet type \( k \). Finally, let \( \Pi \) denote the minimum total block time to be shifted from \( K^1 \) to \( K^2 \). The following equations are added to the basic formulation:

\[
\sum_{k \in K^1} \left( \sum_{i \in D^k} \beta_i^k x_i^k \right) \leq \sum_{k \in K^2} \pi^k - \Pi, \tag{19}
\]

\[
\sum_{k \in K^1} \left( \sum_{i \in D^k} \beta_i^k x_i^k \right) \geq \sum_{k \in K^2} \pi^k + \Pi. \tag{20}
\]

### 3.5 Balancing Module

This module finds the most profitable set of swaps that will balance the schedule when the fleet assignments are changed for a user-specified set of flights. Let \( C \) denote the set of flights whose fleet assignments are changed, and \( n(i) \) denote the new fleet type for flight \( i \). The following equation needs to be added to the basic re-fleeting formulation:

\[
x_i^{n(i)} = 1 \quad \forall i \in C. \tag{21}
\]

### 4. SOLUTION APPROACH

The goal is to determine \( \kappa \) near-optimal solutions for the Re-Fleeting Model, where \( \kappa \) is user-defined. It should be clarified that we are not looking for the best \( \kappa \) solutions; rather, the goal is to present the user of the model with \( \kappa \) solutions satisfying user-specified optimality gaps. To achieve this, the approach generates a first acceptable solution, adds cuts to exclude the solution, generates a second acceptable solution, and so on. To exclude a solution, the added cuts should ensure that the new solution is either a strict superset of the solution to be excluded, or contains a strict subset of that solution.

In practice, the first case does not generate interesting solutions. The reason is that a user would hardly ever prefer, to a solution \( S \), another solution that is usually more costly and has the same fleet-type swaps as those of \( S \) along with at least one additional swap. For this reason, only the following equation is used in the model to cut off an existing solution:

\[
\sum_{i,k \in S} x_i^k \leq |S| - 1. \tag{22}
\]

The implementation of the cut varies depending on the objective of each module. For the Popping Module, the \( x_i^k \) variables are the only ones used in the cut; this enables each new solution to generate a new set of flights to be removed. For the Change-of-Gauge and Utilization modules, the \( x_i^k \) variables used are those that involve swaps between the two groups, \( K^1 \) and \( K^2 \); this is consistent with the primary objective of these modules, which is changing fleet assignments among and not within the groups. Finally, all the \( x_i^k \) variables are used for the Swapping and Balancing modules.

In the solution algorithm, we make use of function calls to CPLEX's mathematical optimization subroutine library (version 4.0). The following discusses the steps used in generating multiple solutions for the model:

1. Aggregate the problem using the optimizer's algebraic preprocessor.
2. Solve the linear relaxation of the model using dual simplex with steepest-edge pricing, starting with the basis from the latest LP solution (when it exists).
3. For each \( x_i^{k'} \), variable that has a value of one in the LP relaxation, restrict the choice of fleet types for flight \( i \) to fleet type \( k' \). all other fleet types with the same seat capacity, the fleet-type(s) with the next larger seat capacity, and the fleet-types with the next smaller seat capacity. Of course, all these fleet types must also be capable of operating flight \( i \). Using this scheme, the choice for flight \( i \) becomes restricted to the compatible fleet types that are closest in seat capacity to \( k' \).
4. Set the estimated lower bound for the MIP as the larger of the following two quantities:
   - LP optimal objective function minus a pre-defined amount (in our implementation, we used $4,000 or 10\% of the LP optimal objective function, whichever is smaller).
   - The solution in set \( B \) that has the highest objective function value if such a solution exists. This set is defined as that of all solutions found so far during the branch-and-bound searches (Steps 7 and 8 below) and that survive all the added cuts described by Eq. 22 (also generated in Steps 7 and 8).
5. Define Special Ordered Sets (SOS) using the cover constraints (Eq. 3), and order the variables by their objective function coefficients modified by subtracting the most negative coefficient from all the others (because only positive weights are allowed in CPLEX). Assuming that the ordered set of variables involved (\( S \)) is indexed by \( 1, \ldots, r, \ldots, n \), the SOS branching scheme would result in the following two sets:
   - \( S_1 = S \cap \{ x_i = 0, i = 1, \ldots, r \} \)
   - \( S_2 = S \cap \{ x_i = 0, i = r + 1, \ldots, n \} \)
   - where \( r = \min(\{ S_{i-1} x_i^k = \frac{1}{2} \}) \) and \( x^n \) corresponds to the current fractional LP solution (see WOLSEY, 1998). For this problem class, SOS branching allows for a more balanced division of variables and significantly smaller
5. COMPUTATIONAL RESULTS

We implemented the Re-Fleeting Model on an HP C160 workstation with 160 MHz and 256 MB RAM. Typical runs of the model used less than one MB of RAM. Coding was done in C, with embedded CPLEX function calls. Table II shows the characteristics of runs conducted for the various modules. These runs

<table>
<thead>
<tr>
<th>Module Name</th>
<th>Scenario</th>
<th>Fleet Types</th>
<th>Flights</th>
<th>Aircraft</th>
<th>Rows</th>
<th>Columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Popping</td>
<td>A DC10-10 is removed and swapping is allowed with the 767-200s, 757-200s, 727-200As, A320s, 737-300s, and 737-200s.</td>
<td>7</td>
<td>818</td>
<td>204</td>
<td>3,407</td>
<td>5,205</td>
</tr>
<tr>
<td>Change-of-Gauge</td>
<td>A 727-200A is replaced with a 757-200, and swapping is allowed with the 737-300s (BA and BB), A320s, and 767-200s.</td>
<td>6</td>
<td>616</td>
<td>127</td>
<td>2,340</td>
<td>3,860</td>
</tr>
<tr>
<td>Swapping</td>
<td>The schedule is re-optimized, considering swapping between the 757-200s (MQ and MU), A320s, 737-300s, and 737-500s</td>
<td>5</td>
<td>501</td>
<td>121</td>
<td>1,634</td>
<td>2,343</td>
</tr>
<tr>
<td>Utilization</td>
<td>Between 5 and 10 hours of flying per day are shifted from the 757-200s (MQ and MU) to the Airbus 320s and DC10-10s.</td>
<td>4</td>
<td>259</td>
<td>78</td>
<td>795</td>
<td>1,246</td>
</tr>
<tr>
<td>Balancing (1)</td>
<td>The equipment type is changed from 737-300 to 737-200 on the flight from DCA to ORD at 14:59. Swapping is allowed with the 737-500s</td>
<td>3</td>
<td>409</td>
<td>81</td>
<td>1,107</td>
<td>1,426</td>
</tr>
<tr>
<td>Balancing (2)</td>
<td>Same as Balancing (1) except swapping with the 737-500s is not allowed.</td>
<td>2</td>
<td>241</td>
<td>48</td>
<td>529</td>
<td>560</td>
</tr>
</tbody>
</table>

branch and bound as compared to single-variable branching.

6. Establish higher branching priority for certain variables than for the SOS. In particular,
- For the Popping Module, the branching priorities are as follows: the \( x^0_i \) variables appearing in the current set of cuts (Eq. 22), the other \( x^0_i \) variables and, finally, the SOS.
- For all the other modules, give first priority to the variables appearing in the current set of cuts and then to the SOS.

7. Perform branch and bound for a first integer solution using a depth-first strategy. The strategy allows for backtracking only in case of infeasibility or exceeding the upper bound, and then resumes depth-first search.
- If no solution is obtained, and a lower bound exists, remove the lower bound and repeat step 7.
- Else if no solution is obtained, and no lower bound exists, stop with an infeasible MIP.
- Else if a solution is obtained, add solution to current set of solutions found so far (B):
  - If solution objective function is within a pre-specified amount from the best upper bound obtained by CPLEX (in our implementation 0.1% of the best upper bound obtained by CPLEX or $100, whichever is larger), stop with the solution. If further solutions are desired, add new cut (Eq. 22) to eliminate the solution generated, remove from set \( B \) all other solutions that become infeasible because of the new cut, and go back to step 2; otherwise stop the procedure.
  - Else if solution objective function is not within 0.1% of the best upper bound obtained by CPLEX or $100 (whichever is larger), continue.

8. Switch back to CPLEX’s default branching strategy and solve the MIP for further integer solutions, while ignoring nodes whose objective value is not a pre-specified amount ($100 in our implementation) better than the best solution obtained. Add new solutions encountered in the branch-and-bound to set \( B \). Stop when the best solution obtained is within a pre-specified amount (in our implementation 0.1% or $100 whichever is larger) from the best upper bound obtained by CPLEX.
- If further solutions are desired, add new cut (Eq. 22) to eliminate the solution generated, remove from set \( B \) all other solutions that become infeasible because of the new cut, and go back to step 2.
- Otherwise, stop the procedure.
correspond to generated rather than actual problem sets; however, the requirements described are quite representative of real problems encountered daily by the planners. In each case, the objective function reflects the incremental change in network contribution due to the run. The network contribution consists of the revenue accrued by the flights (which depends on seat capacity assigned) minus operating costs.

To study the performance of the model for various problem sizes, the runs were chosen to span in size from two fleet types and 241 flights, to seven fleet

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Popping</td>
<td>1</td>
<td>11.7</td>
<td>5.9</td>
<td>17.6</td>
<td>14,713</td>
<td>12,052</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2.8</td>
<td>5.2</td>
<td>8.0</td>
<td>12,818</td>
<td>8,530</td>
<td>8</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>1.0</td>
<td>10.3</td>
<td>11.3</td>
<td>12,818</td>
<td>4,792</td>
<td>8</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1.2</td>
<td>15.8</td>
<td>17.0</td>
<td>12,818</td>
<td>803</td>
<td>16</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>1.4</td>
<td>103.9</td>
<td>105.3</td>
<td>12,818</td>
<td>-248</td>
<td>201</td>
<td>200</td>
<td>81</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>1.2</td>
<td>31.6</td>
<td>32.8</td>
<td>12,818</td>
<td>-998</td>
<td>40</td>
<td>33</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>1.2</td>
<td>42.4</td>
<td>43.6</td>
<td>12,818</td>
<td>-1,749</td>
<td>46</td>
<td>40</td>
<td>-1,749</td>
</tr>
<tr>
<td>Total Run Time:</td>
<td>235.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Change-of-Gauge</td>
<td>1</td>
<td>8.8</td>
<td>21.8</td>
<td>30.6</td>
<td>24,843</td>
<td>18,694</td>
<td>58</td>
<td>56</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.4</td>
<td>25.5</td>
<td>25.9</td>
<td>24,843</td>
<td>18,096</td>
<td>70</td>
<td>31</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.4</td>
<td>29.0</td>
<td>29.4</td>
<td>24,843</td>
<td>18,134</td>
<td>80</td>
<td>72</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.4</td>
<td>33.9</td>
<td>34.3</td>
<td>24,843</td>
<td>18,094</td>
<td>81</td>
<td>74</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.4</td>
<td>35.4</td>
<td>35.8</td>
<td>24,843</td>
<td>17,911</td>
<td>89</td>
<td>85</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.4</td>
<td>34.1</td>
<td>34.5</td>
<td>24,843</td>
<td>17,871</td>
<td>85</td>
<td>82</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>0.4</td>
<td>34.8</td>
<td>35.2</td>
<td>24,843</td>
<td>17,769</td>
<td>88</td>
<td>28</td>
<td>28</td>
</tr>
<tr>
<td>Total Run Time:</td>
<td>225.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Swapping</td>
<td>1</td>
<td>3.5</td>
<td>2.9</td>
<td>6.4</td>
<td>9,219</td>
<td>8,399</td>
<td>28</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.2</td>
<td>10.6</td>
<td>10.8</td>
<td>9,219</td>
<td>8,399</td>
<td>47</td>
<td>47</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.9</td>
<td>6.4</td>
<td>7.3</td>
<td>9,206</td>
<td>6,936</td>
<td>44</td>
<td>33</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.3</td>
<td>6.4</td>
<td>6.7</td>
<td>9,206</td>
<td>6,935</td>
<td>45</td>
<td>34</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.3</td>
<td>5.9</td>
<td>6.2</td>
<td>9,206</td>
<td>6,146</td>
<td>32</td>
<td>42</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.3</td>
<td>12.1</td>
<td>12.4</td>
<td>9,206</td>
<td>5,957</td>
<td>72</td>
<td>40</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>0.3</td>
<td>9.9</td>
<td>10.2</td>
<td>9,206</td>
<td>5,910</td>
<td>120</td>
<td>82</td>
<td>15</td>
</tr>
<tr>
<td>Total Run Time:</td>
<td>60.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Utilization</td>
<td>1</td>
<td>0.6</td>
<td>3.2</td>
<td>3.8</td>
<td>10,353</td>
<td>5,740</td>
<td>34</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.9</td>
<td>3.2</td>
<td>4.1</td>
<td>10,308</td>
<td>3,920</td>
<td>42</td>
<td>39</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.7</td>
<td>3.6</td>
<td>4.3</td>
<td>10,308</td>
<td>1,985</td>
<td>50</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.1</td>
<td>4.8</td>
<td>4.9</td>
<td>10,268</td>
<td>1,480</td>
<td>63</td>
<td>58</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.7</td>
<td>3.7</td>
<td>4.4</td>
<td>10,268</td>
<td>-340</td>
<td>46</td>
<td>43</td>
<td>-340</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.7</td>
<td>5.5</td>
<td>6.2</td>
<td>10,268</td>
<td>-947</td>
<td>79</td>
<td>37</td>
<td>-594</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>0.7</td>
<td>6.2</td>
<td>6.9</td>
<td>10,268</td>
<td>-2,468</td>
<td>110</td>
<td>68</td>
<td>-2,468</td>
</tr>
<tr>
<td>Total Run Time:</td>
<td>34.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Balancing (1)</td>
<td>1</td>
<td>0.6</td>
<td>0.8</td>
<td>1.4</td>
<td>3,782</td>
<td>3,370</td>
<td>12</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.2</td>
<td>0.6</td>
<td>0.8</td>
<td>3,728</td>
<td>2,990</td>
<td>84</td>
<td>84</td>
<td>61</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.9</td>
<td>3.8</td>
<td>4.7</td>
<td>3,728</td>
<td>1,689</td>
<td>42</td>
<td>40</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.9</td>
<td>4.8</td>
<td>5.7</td>
<td>3,728</td>
<td>1,529</td>
<td>48</td>
<td>42</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.9</td>
<td>12.7</td>
<td>13.6</td>
<td>3,728</td>
<td>1,524</td>
<td>156</td>
<td>31</td>
<td>-2,123</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>1.0</td>
<td>5.1</td>
<td>6.1</td>
<td>3,728</td>
<td>1,498</td>
<td>53</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>1.0</td>
<td>5.8</td>
<td>6.8</td>
<td>3,728</td>
<td>1,373</td>
<td>62</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>Total Run Time:</td>
<td>45.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Balancing (2)</td>
<td>1</td>
<td>0.9</td>
<td>0.3</td>
<td>1.2</td>
<td>1,131</td>
<td>-111</td>
<td>26</td>
<td>18</td>
<td>-111</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.2</td>
<td>0.4</td>
<td>0.6</td>
<td>1,131</td>
<td>-38</td>
<td>30</td>
<td>26</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.2</td>
<td>0.5</td>
<td>0.7</td>
<td>1,131</td>
<td>-370</td>
<td>34</td>
<td>25</td>
<td>-744</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.4</td>
<td>0.4</td>
<td>0.8</td>
<td>1,117</td>
<td>-542</td>
<td>36</td>
<td>32</td>
<td>-720</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.3</td>
<td>0.6</td>
<td>0.9</td>
<td>817</td>
<td>-720</td>
<td>41</td>
<td>8</td>
<td>-720</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>0.3</td>
<td>0.9</td>
<td>1.2</td>
<td>817</td>
<td>-664</td>
<td>67</td>
<td>64</td>
<td>-1,542</td>
</tr>
<tr>
<td>Total Run Time:</td>
<td>6.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Statistics are: Solution number, LP solution time, IP solution time, total time, LP objective value, IP objective value, total number of nodes in the branch-and-bound tree, node number of the near-optimal solution, node number of the first integer solution, objective value of the first integer solution, and the total number of integer solutions found in the whole branch-and-bound tree.
types and 818 flights. In practice, however, typical runs involve one to five fleet types and rarely have larger dimensions than those of the run shown for the Swapping Module.

Table III gives detailed computational statistics for the best seven solutions found for each of the runs when only 15 fleet-type swaps are allowed. Table IV gives similar statistics when 30 fleet-type swaps are allowed.

Several observations can be made regarding Tables III and IV. To start with, the run times are very reasonable for the Balancing, Utilization, and Swapping Modules. The problem sizes tested for these modules are typical of the problems addressed by
schedulers on a daily basis. The problems used to test the Change-of-Gauge and Popping Modules are larger than those addressed in practice. Nonetheless, the run times are still quite reasonable (less than 5 minutes), considering that the flights involved in the corresponding runs comprise about 30% and 40% of UA’s domestic mainline schedule, respectively.

Within each run, the first LP typically takes significantly more time to solve than the subsequent LPs; this is a testimony to the usefulness of using advanced starting basis for the second and later LPs. The results for the IP branch-and-bound trees indicate that the IP phase was, overall, quite effective: a first solution was found after a few number of nodes, multiple improved solutions were often found, and no extensive tree searching was needed to terminate with the best near-optimal solution. Note that the best solution is not necessarily the optimal one due to the following two reasons already discussed in the statement of the algorithm:

1. The branch-and-bound search terminates when a solution is found with an acceptable optimality gap (in our implementation, 0.1% of the best upper bound obtained by CPLEX or $100, whichever is larger).  

2. In the branch-and-bound search, nodes are fathomed if their objective value is not a pre-specified amount ($100 in our implementation) better than the best solution obtained so far.

An interesting finding is that the cuts rarely impact the objective function of the LP relaxation. This is a testimony to the highly degenerate nature of fleet management problems with their embedded multi-commodity networks. This frequently allows the IP solver to find multiple near-optimal solutions that are quite close in objective value.

Finally, by comparing Tables III and IV, it is evident that the increased number of swaps allowed leads to significantly improved solutions. This means that, whenever possible, users may increase the model’s capability to find good solutions by increasing the number of allowable swaps. This, of course, must be balanced with the need to stay as close as possible to the planned fleet assignments.

6. SUMMARY AND CONCLUSIONS

IN THIS PAPER, we formulated the re-fleeting problem using a multi-commodity integer flow network with side constraints. The model comprehensively addresses the re-fleeting questions that arise in practice. Efficient preprocessing and solution procedures were used to detect infeasibilities early on, and to achieve multiple near-optimal solutions in very reasonable computer run times. The Re-Fleeting Model presented can be used effectively, in conjunction with a Fleet Assignment Model, to produce high-quality fleet assignment plans. Finally, although the context of the Re-Fleeting model is a planning rather than operational one, the very fast response times we experienced with the model indicates that there is a potential for extending the model for use in an operational environment.

REFERENCES


(Received: July 1997; revisions received: August 1998, April 1999; Accepted: August 1999)