Airline Fleet Assignment with Time Windows

BRIAN REXING, CYNTHIA BARNHART, TIM KNIKER
Massachusetts Institute of Technology, Center for Transportation Studies, Cambridge, Massachusetts 02139

AHMAD JARRAH
Transport Dynamics, Inc., Carnegie Center, Princeton, New Jersey 08540

NIRUP KRISHNAMURTHY
United Airlines, P.O. Box 66100, Chicago, Illinois 60666

Recognizing that allowing variability in scheduled flight departure times can result in improved flight connection opportunities and a more cost effective fleet assignment, we present a generalized fleet assignment model for simultaneously assigning aircraft types to flights and scheduling flight departures. Our model, a simple variant of basic fleet assignment models, assigns a time window to each flight and then discretizes each window, allowing flight departure times to be optimized. Because problem size can become formidable, much larger than basic fleet assignment models, we develop two algorithmic approaches for solving the model. Our direct solution approach is good for speed and simplicity, whereas our iterative technique minimizes memory usage. Using data from a major U.S. airline, we show that our model can solve real, large-scale problems, and we evaluate the effects of schedule flexibility. In every test scenario, the model produces a fleet assignment with significantly lower costs than the basic model, and, in a separate analysis, the model is used to tighten the schedule, potentially saving aircraft.

Deciding when and where to offer flights is not the end of the airline schedule planning process. Given these initial strategic decisions, the schedule planner must resolve a series of issues before the schedule is actually operational. These issues include, among others, assigning aircraft to flight legs, routing the aircraft to ensure maintenance, and crew scheduling. The allocation of aircraft to flight legs, or fleet assignment, is the first of these issues that is addressed.

Fleet assignment has a tremendous impact on an airline's profits, because it directly affects flight operating costs and passenger revenues. Consider, for example, a large aircraft flying a leg with very few passengers, versus a small aircraft serving a route with high passenger demand. In the first case, the prohibitive cost of flying the large aircraft was unnecessary, whereas in the second case the airline lost potential revenue because passengers that would not fit on the aircraft were turned away. Clearly, an efficient fleet assignment depends on accurate forecasts of operating costs and passenger traffic. The traffic forecasts, together with known aircraft seating capacities and expected fares allow the calculation of spill costs, or lost revenue due to under-capacity. With operating costs, spill costs, aircraft data, and schedule data known, we can define the basic fleet assignment problem as:

PROBLEM. Given a flight schedule with fixed departure times and costs (operating plus spill) according
to each aircraft type on each flight leg, find the least cost assignment of aircraft types to flights, such that (1) each flight is covered by exactly one fleet type, (2) flow of aircraft by type is balanced at each airport, and (3) only the available number aircraft of each type are used.

Although not discussed in this paper, other constraints might be added to this problem to improve the likelihood that the solution will be operational. These include considerations for gate and maintenance availability, crew scheduling, and aircraft noise.

This paper describes a daily fleet assignment problem, meaning we assume that the airline’s schedule repeats day after day. This assumption is typically adopted at U.S. airlines, because they fly the same schedule every day (with some exceptions). Some airlines deviate from this repeating schedule on weekends, but re-fleeting for these exceptions is handled after the daily assignments have been made.

Models that solve the basic daily fleet assignment problem have found wide acceptance among airlines, but despite their importance, these models leave room for improvement. A noticeable weakness of current fleet assignment models is their requirement that flight departure times be fixed. In reality, the schedule used as an input to the fleet assignment model is not cast in stone; many of the scheduled times will be tweaked in the weeks leading up to the day of departure to improve such factors as gate availability and passenger connection times. It is also not uncommon for schedulers to manually re-time flights in search of an improved fleet assignment. The example in Table I illustrates how modifying the schedule might allow a better fleet assignment. Notice that both flights A and B have the same forecasted demand, and thus an assignment of the same type of aircraft to both flights might be appropriate. But, because flight A’s ready time (arrival time plus minimum ground service time) is after flight B is scheduled to depart, the same aircraft cannot cover both flights. However, if flight B could be re-timed to depart after 1000, the connection could be made, and a single aircraft could cover both flights, possibly improving the fleet assignment.

In our model, we assume that flight demand is constant and independent of the flight departure times and fleet assignments. In fact, this assumption might introduce inaccuracies because changes in departure times, connection times and fleet assignments (because some passengers prefer connecting flights that require no plane change) can certainly affect demand. We justify, however, our assumptions based on the following:

1. Changes in demand will be minimal because schedule adjustments are limited to small amounts of time; and
2. The demand estimates that are input to these conventional fleet assignment models might not be very accurate anyway because they are leg-based. This means that, in estimating revenues of a particular fleeting, conventional fleet assignment models allow passengers on multi-leg itineraries to be spilled on one or more legs and carried on the remaining legs. Obviously, this introduces error in the revenue estimates.

To assess the impact of these simplifying assumptions and approximations, additional research on itinerary-based fleet assignment is necessary.

By adding time windows, which define by how much any given flight can shift, the set of feasible fleet assignment solutions grows substantially, and the optimal solution (given our fixed demand assumption) is guaranteed to be at least as good as the basic (fixed departure time) model. So then, we model the fleet assignment with time windows problem the same as the basic problem, except that time windows, rather than fixed departure times, are given for each flight.

**LITERATURE REVIEW**

Because of their effectiveness in improving profits, not to mention the efficiency of the planning process, airline schedule development models have been given much attention. This is evident in GoPalan and Talluri (1995), a survey of mathematical models used during schedule development. Fleet assignment in particular is a well-researched topic, with Daskin and Panayotopoulos (1989), Abara (1989), Berg and Hopperstad (1993), Hane et al. (1995), Subramanian et al. (1994), Gu et al. (1994), and Clarke et al. (1996), each presenting formulations to variations of the basic daily fleet assignment model within the last eight years. Daskin and Panayotopoulos (1989) present an integer program that assigns aircraft to routes (which they define as sequences of flight legs originating and terminating at
the same airport) in single-hub networks. Lagrangian relaxation is used to find an upper bound, and heuristics are used to find specific solutions. Abara (1989) presents a model that can be used for more general airline networks, but the model has some limitations due to the choice of “feasible turns” (connection arcs) as decision variables. The model size explodes unless limits are placed on the connection opportunities. Another limitation is that different flying times and turn times (minimum ground service times) are not allowed for different fleets. Berge and Hopperstad (1993) present a “fundamentally new operating concept” for dynamically re-assigning aircraft as traffic forecasts change throughout the planning horizon. Because the re-assignment problems involve only a common-cockpit subset, or “family,” of an airline’s aircraft (so as not to destroy the crew assignments), the problems are smaller than an airline’s full daily problem. These problems are formulated as multicommodity network flow problems on space–time networks. To obtain quick solutions, two heuristic algorithms specifically designed for these re-assignment problems are suggested: one heuristic solves a sequence of single-commodity flow problems, and the other begins with a feasible assignment and performs multiple profit-improving aircraft swaps. Hane et al. (1995) use a similar multicommodity network flow formulation to solve the basic fleet assignment problem, but their use of variable aggregation, cost perturbations, dual simplex with steepest-edge pricing, and intelligent branch-and-bound strategies result in a rapid solution to realistically-sized (2500-flight, eleven-fleet) problems. Gu et al. (1994) study the complexity and behavior of the model presented in the Hane paper, whereas in Clarke et al. (1996), a sequel to the Hane paper, it is shown that better operational solutions to the fleet assignment problem can be achieved by adding variables and constraints that model maintenance and crew issues.

Levin (1971) was the first to propose a scheduling and fleet routing model with time windows, however this model did not consider multiple fleet types. Time windows were modeled by allowing departure times to occur at discrete intervals within the time window. Much more recently, Bandet (1994) and Desaulniers et al. (1994) presented formulations for the fleet assignment and aircraft routing problem with time windows. Desaulniers et al. solved their models by decomposing the problem and generating favorable flight paths with time-constrained shortest path subproblems. The models are demonstrated to be effective on problems with fewer than 400 flights per day.

CONTRIBUTION

In this paper, we consider large-scale problems of fleet assignment with time windows. The contributions of this paper include the following:

1. By discretizing each flight’s departure time window, we model the fleet assignment with time windows problem as a simple variant of the basic fleet assignment model. Ours is a generalized model and solution approach for fleet assignment, in that time windows of zero width can be used (resulting in the basic fleet assignment model) or flights can be given time windows of varying width. Furthermore, the structural similarity between our model and those models currently implemented at many airlines is a key attribute of our approach.

2. We present two algorithmic approaches for solving the fleet assignment with time windows model. Both approaches begin with network preprocessing, some of which we designed specifically for our new model. Then, we demonstrate that the model can be solved either directly or by using an iterative solution technique. Our direct solution approach (DST) is good for speed and simplicity, whereas our iterative approach (IST) minimizes memory usage (which is valuable for larger problems and problems with added enhancements).

3. Using data from a major U.S. airline, we show that our model can solve real, large-scale problems, and we evaluate the effects of schedule flexibility. We take an in-depth look at the effects of network preprocessing, compare the performance of our two algorithms, and show that allowing time windows can save aircraft and improve daily operating and spill costs. For example, on a 2037-flight, seven-fleet problem with time windows of a maximum of ±10 minutes on all flights, our model finds a solution that reduces fleet assignment costs by over $67,000 per day beyond the optimal solution with no time windows. In that solution, less than 6% of all flights are moved from their originally scheduled time. In a separate analysis on the same problem instance, we find that the flight schedule can be flown with two fewer aircraft than are necessary when departure times are fixed. Although the operational feasibility of these new solutions must be verified, the results are certainly promising.

OUTLINE

In Sections 1 and 2, we describe the mathematical models and flight networks for the basic fleet assignment problem and the enhanced fleet assignment
problem with time windows. In Section 3, we detail the network preprocessing steps and then motivate and describe the alternative solution techniques we have designed. The results we obtained in solving fleet assignment problems for a large airline are detailed in Section 4. Finally, in Section 5 we summarize this research and suggest pertinent future research.

1. BASIC FLEET ASSIGNMENT MODEL

In this section, we present the integer programming formulation to the basic fleet assignment problem, because our formulation for the fleet assignment problem with time windows is a simple extension of this model. We begin by describing a network representation of the problem and then present the basic fleet assignment formulation.

1.1 Network Representation

The flight networks are the foundation upon which the fleet assignment model is built, for they ensure feasible flight connections, conservation of aircraft flow, and a balanced number of aircraft of each type at each airport at the beginning and end of each day. A different flight network is created for each fleet type, and it is made up of parallel time lines—one time line for each airport. An example of a two-airport, four-flight network is shown in Figure 1. Each time line is composed of a series of ground arcs that connect event nodes. Event nodes represent departures or arrivals at an airport at a specific time. The parallel time lines are connected by flight arcs, which are defined by their departure and arrival nodes. Wrap-around arcs, which are just special-case ground arcs, connect the last event node to the first event node at each airport. This guarantees flow balance.

To ensure that aircraft make only feasible flight connections, arrival nodes are not actually placed at the flight’s arrival time; rather, they are placed at the flight’s ready time. If we did not place arrival nodes at each flight’s ready time, we might allow aircraft to connect to departing flights before they are actually prepared to leave. For example, in Figure 1, if flight B were put on the BOS time line at its arrival time, an aircraft would then be allowed to flow across both flight B and flight C, which is actually impossible. Turn times, which can be different for every fleet type, are typically 30 to 40 minutes for domestic flights, depending on aircraft size. Because we create a different network for each fleet type, we can model not only this disparity in turn times, but also the differences in flying times among different aircraft types.

1.2 Formulation

Once the flight networks have been built for each fleet, the basic fleet assignment problem can be solved using the formulation given in Figure 2. Stated simply, this formulation seeks to minimize the cost to cover each flight with one aircraft type (constraints 1), making sure aircraft flow is conserved at each node (constraints 2), while using no more aircraft of each type than are available (constraints 3). The third constraint set counts aircraft of each fleet type by summing the flow on all arcs of that fleet’s network that cross an arbitrarily chosen count time. A typical count time is 3:30 A.M. because only wrap-around arcs and a few (red-eye) flight arcs cross this time. Decision variables \( x \) define which fleet type covers each flight, so these variables must be integer (constraints 4). The \( y \) variables are zero-cost ground arcs that are necessary to preserve conservation of flow and to count the number of aircraft used. These variables need not be defined as integer (constraints 5), because the integrality of the flight variables ensures integral ground flow.

Objective Function Coefficients

Determining accurate cost coefficients is easily the most important, and difficult, factor in obtaining an efficient fleet assignment solution. Each coefficient is composed of two parts: an operating cost and a spill cost. The operating cost portion (fuel, maintenance, crew, etc.) is relatively easy to estimate, however it can become more complicated as passenger-related costs are added. Passenger-related costs are a function of the number of passengers carried, which is a function of aircraft size.

Given an unconstrained demand for a flight, to calculate the expected number of passengers carried on that flight for a given fleet type, we must calculate the expected spill. Recall that spill is the number of passengers turned away by an airline because demand was greater than seating capacity. Before discussing the possible methods for calculating spill,
we define the following expressions:

\[ \text{spill}_{ik} = \text{expected number of passengers spilled from flight } i \text{ if flown by fleet } k \]
\[ \mu_i = \text{mean unconstrained demand of flight } i \]
\[ \sigma_i = \text{standard deviation of demand of flight } i \]
\[ \text{cap}_k = \text{seating capacity of fleet } k \]
\[ \text{max LF}_i = \text{maximum load factor allowed on flight } i \leq 1 \]
\[ f(x_i; \mu_i; \sigma_i) = \text{the Gaussian (normal) probability density function} \]

One approach for estimating spill is

\[ \text{spill}_{ik} = \max\{0, \mu_i - \text{cap}_k\}. \]

In other words, spill is equal to zero unless the mean unconstrained demand is greater than capacity. However, unless demand is deterministic, this equation underestimates spill. Consider, for example, a flight that has a mean unconstrained demand of 100 passengers and an aircraft with a capacity of 100 seats. This equation tells us that there is an expected spill of zero passengers. Assuming that the demand actually varies from day to day, this estimate is clearly wrong, for any time the demand exceeds 100, there will be spill.

A simple and artificial way to account for this stochasticity in demand is to place a limit on the effective capacity of an aircraft:

\[ \text{spill}_{ik} = \max\{0, \mu_i - \text{max LF}_i \ast \text{cap}_k\}. \]
Using the example above with a maximum load factor of 0.9, our estimate of spill becomes 10 passengers.

However, a more realistic approach for calculating spill is to assume that demand has a normal distribution, and calculate spill accordingly, as discussed in Subramanian et al. (1994),

\[
\text{spill}_{ik} = \int_{\text{cap}_i} f_i(x_\omega; \mu_i, \sigma_i)(x_\omega - \text{cap}_k) \, dx_\omega.
\]

Given this estimate of spill, we can determine a spill cost for each flight–fleet pair. Spill cost is the expected number of passengers spilled multiplied by spill fare—an estimate of the average revenue per spilled passenger on that leg. Spill fare is another piece of data (in addition to mean unconstrained demand and its standard deviation) that is difficult to estimate, yet has significant impact on the fleet assignment solution. In our implementation, we have assumed spill fare to be equivalent to the average fare paid on that leg.

2. FLEET ASSIGNMENT WITH TIME WINDOWS MODEL

2.1 Network Representation

Time windows can be modeled with a simple, albeit computationally expensive, extension of the basic flight networks. By placing copies of a flight arc at specified intervals within that flight’s time window and requiring just one of the flight arc copies to be covered, we allow the model to choose the departure time of the flight. Figure 3 shows how the basic network of Figure 1 might look after being expanded to permit flexible departure times for each of the four flights. The departure and corresponding arrival time windows are indicated by the rectangles above (below) the BOS (ORD) time line. The impact that time windows have can be seen in this small example: a single aircraft may now cover flights B and C (by using the first copy of flight B and the last copy of flight C), whereas this connection was not possible in the basic model.

Because the scheduled time of some flights is more flexible than others, the width of each time window is a parameter that can be different for every flight. Obviously, wider time windows will allow more connections, but, if a flight is moved too far from its originally scheduled time, then the passenger demand for that flight might change. Shuttle flights, arrival slot-controlled flights (e.g., at some international and congested domestic airports), and flights that are members of banks are examples of flights that might have a tightly constrained (possibly zero-width) time window. A bank is a hub airport activity that allows passengers from many origins to connect to flights that are departing to many different destinations. Figure 4 shows what a small bank would look like at the hub airport. So as not to clutter the figure, the basic network representation is shown (i.e., only one arc is shown for each flight). One can see why bank flights might have tightly constrained time windows by considering flight E. If flight E were allowed to arrive any later than its originally scheduled time, some passengers might not have time to run across the airport and board outbound flight F. So, flight E’s time window might be, for example, \([-10, 0]\) meaning it can be re-timed up to ten minutes earlier, but cannot be re-timed any later than its originally scheduled time.

The interval at which flight copies are placed within each time window is another parameter that can affect the quality of the solution. To guarantee that flights are allowed to depart at any time within their window, copies need to be placed at one-minute intervals. However, it will be shown in Sections 4.3 and 4.4 that using a narrow interval causes an explosion in the problem size, often without the benefit of providing a substantially better solution than a broader (say five-minute) interval.

2.2 Formulation

The fleet assignment with time windows model is almost exactly the same as the basic model, only it...
Minimize \[
\sum_{k \in K} \sum_{i \in F} \sum_{n \in N_{ik}} c_{ik} x_{nik}
\]
subject to \[
\sum_{k \in K} \sum_{n \in N_{ik}} x_{nik} = 1 \quad \forall i \in F \tag{1}
\]
\[
\sum_{i \in F} \sum_{n \in N_{ik}} b_{1_{nik}} n_{nik} + \sum_{g \in G_k} b_{2_{igk}} y_{gh} = 0 \quad \forall l \in L_k, \quad \forall k \in K \tag{2}
\]
\[
\sum_{i \in F} \sum_{n \in N_{ik}} d_{1_{nik}} x_{nik} + \sum_{g \in G_k} d_{2_{igk}} y_{gh} \leq S_k \quad \forall k \in K \tag{3}
\]
\[
x_{nik} \in \{0, 1\} \quad \forall i \in F, \quad \forall n \in N_{ik}, \quad \forall k \in K \tag{4}
\]
\[
y_{gh} \geq 0 \quad \forall g \in G_k, \quad \forall k \in K \tag{5}
\]

Parameters
\[
N_{ik} = \text{set of arc copies of flight } i \text{ in fleet } k \text{'s network}
\]
\[
|N_{ik}| = \text{number of arc copies of flight } i \text{ with fleet type } k
\]
\[
c_{ik} = \text{cost to fly flight } i \text{ with fleet type } k
\]
\[
b_{1_{nik}} = \begin{cases} 
1 & \text{if copy } n \text{ of flight } i \text{ begins at node } l \text{ in fleet } k \text{'s network} \\
-1 & \text{if copy } n \text{ of flight } i \text{ ends at node } l \text{ in fleet } k \text{'s network} \\
0 & \text{otherwise}
\end{cases}
\]
\[
d_{1_{nik}} = \begin{cases} 
1 & \text{if copy } n \text{ of flight } i \text{ crosses the count time in fleet } k \text{'s network} \\
0 & \text{otherwise}
\end{cases}
\]

Decision Variables
\[
x_{nik} = \begin{cases} 
1 & \text{if copy } n \text{ of flight } i \text{ flown by fleet } k \\
0 & \text{otherwise}
\end{cases}
\]

Fig. 5. Fleet assignment with time windows model.

contains more variables and has slightly different coverage constraints. Notation that is different from the basic model’s notation and the new model are shown in Figure 5. This model is structured the same as the basic model, with the objective function minimizing cost, and the constraints ensuring flight coverage (constraints 1), conservation of flow at each node (constraints 2), restricted aircraft utilization (constraints 3), and variable integrality (constraints 4 and 5). Notice that there is still one constraint for every flight (1), node and fleet type pair (2), and fleet type (3). However, there are far more nodes in the network when time windows are used. As explained in Section 2.1, constraints on a flight’s time window width are modeled by only defining flight arcs that are within that flight’s time window.

Notice, in the first constraint set, that only one arc, among all copies and every fleet, must be covered for each flight. By choosing a single arc copy, the model effectively chooses the departure time of each flight. This model also allows the user to apply a different cost coefficient to every flight copy. So, for example, one can make it more expensive to fly a flight the further it is moved from its originally scheduled time.

3. SOLUTION APPROACH

The LP matrix of the fleet assignment with time windows model can become prohibitively large for realistically-sized problems. As such, preprocessing steps, described in Section 3.1, are useful for reducing the model’s size. After preprocessing, it is often possible to solve the fleet assignment problem directly. However, the direct solution approach requires that we include many unnecessary flight arcs. An alternative strategy, described in Section 3.2, is to iteratively add necessary flight arc copies to the model, thereby minimizing the model’s size.

3.1 Network Preprocessing

Because of the cumbersome size of fleet assignment models, substantial efforts have been made to prune the problem before the LP matrix is constructed and solved. Hane et al. (1995) provide an excellent description of techniques that can be used before solving the basic fleet assignment model. Some of the same concepts are also appropriate for the fleet assignment with time windows model—a model with many more nodes and arcs than the basic model. Described below are three techniques that we use to reduce the network size; specific
results of the impact of these techniques are given in Section 4.2.

Node Consolidation

Node aggregation, or consolidation, is one of the techniques described in Hane et al. (1995) that can also be applied to networks with flight copies. The idea here is to replace one or more arrival nodes that are followed by one or more departure nodes, with a single node. For example, in Figure 4, all ten nodes and the nine ground arcs between them can be replaced by a single node without introducing illegal connections. Rather than representing a specific instant in time, this single node now represents a time range: from the ready time of flight A to the departure time of flight J. If flight H were an arrival, then flights A–G would be consolidated followed by a ground arc into a single node for flights H–J. Each node (ground arc) removed translates into exactly one less row (column) in the LP matrix.

Deleting Redundant Arcs

This is a technique that is unique to networks containing copies of individual arcs (such as our time window flight networks). After nodes of the flight networks have been consolidated, a set of redundant arcs can be defined as arcs of the same flight that have the same cost and share the same tail node. Among these redundant arcs, the first (earliest on the time line) copy dominates the later copies, because an aircraft flowing across the first arc can make, at a minimum, the same connections as any of the later-arriving flights.

Figure 6 shows a portion of a flight network, after node consolidation, containing a single flight with five arc copies between CMH and ORD. In this example, there are two sets of redundant copies: (1, 2) and (3, 4, 5). Assuming these arcs all have the same cost, arcs 2, 4, and 5 can be deleted from the network without affecting the solution. That is, the flights arriving at CMH and the flights departing from ORD have the same connection opportunities with or without these dominated arcs. However, if costs among copies of a flight are not the same, then arcs within a redundant set can only be deleted if they have both a later departure time and a cost not less than another earlier arc within that set. In any case, each arc deleted equates to three fewer variables and two fewer constraints in the model (before node consolidation). Not only is a flight arc (and its two end nodes) removed, but a ground arc is implicitly removed from each airport’s time line because two arcs are merged into one.

We should note that, in our implementation, one arc copy is always placed at the originally scheduled departure time of each flight, and this arc is never deleted. To further favor flights that depart at the originally scheduled time, we reduce the cost of these arcs by a very small amount.

Islands

Another of the techniques presented by Hane et al. (1995) is to remove ground arcs from sparsely-used airports when it is known that no aircraft will be on the ground at that time. Removing these ground arcs leaves the time line with islands—periods of time when there is at least one aircraft on the ground. Figure 7 is an example of what a time line with islands might look like for the basic fleet assignment problem (one arc for each flight). By counting the arrival flights and departure flights, we can identify the maximum number of aircraft that might use each ground arc. For example, here we see that, if there are \( x \) aircraft on the ground in the morning, then there must be \( x - 1 \) aircraft on the ground after the first departure. We continue down the time line, adding to the count for each arrival and subtracting from the count for each departure. The ground arc(s) with the lowest count can be removed. In this example, \( x - 1 \) is the lowest count, and so the two ground arcs with that count are removed. In this case the wrap-around arc is the only ground arc that might have aircraft flow. Notice that removing ground arcs not only removes variables from the problem, but in some cases it also fixes the connections between flights. In this case, if the model chooses the fleet of this time line to cover flight A, then it must also cover flight B.

Figure 8 shows what Figure 7 might look like after flight copies are added to model time windows. As
one can see, there might be a different number of copies for each flight due to differences in time window width and the effect of deleting redundant copies of that flight. This example shows that the process of counting arrival and departure arcs to identify islands becomes more difficult when there are multiple copies of each flight. When counting arrivals and departures, one must only count the last copy of each flight. Ground arcs that sit between the first and last copies of flights should be given the same aircraft count as the previous ground arc. This means that, in Figure 8, the small ground arc between copies of flights A and B gets a count of $x - 1$. Once again, it is found that the wrap-around arc is the only ground arc that may have aircraft flow.

As mentioned previously, this technique should only be applied on time lines of little activity, because it is on these time lines that the assumptions of the counting technique are most likely to be true. To guarantee that we are not eliminating arcs that might be in the optimal solution, the counting technique described requires that arrival flights connect to departure flights emanating from the same node. At heavily used airports, it is less likely that this is true; here it is best that we leave the ground arcs and let the model choose the connections.

3.2 Iterative Solution Technique

After creating the flight networks and preprocessing, one option for solving the fleet assignment problem is to input the entire LP matrix into an LP solver and solve the problem directly. Advantages of a direct solution technique (DST) include simplicity and quickness in solving most problems. However, despite the gains made in preprocessing, DST has a disadvantage of including many unnecessary flight arc copies. That is, there are many flight arcs that are not deleted during the preprocessing phase that are unnecessary for achieving good solutions. Stated another way, even if many of the flight’s departure times are fixed (i.e., only a subset of all flights are given time windows), the model can still achieve the same optimal solution as it does when all flights have time windows. The difficulty lies in identifying which flights need time windows and which flights can be fixed. The IST algorithm, described in the following section, does just that.

The iterative approach is to first solve a fleet assignment problem with one arc for each flight–fleet pair and then add arc copies to the model at each iteration. Extra arc copies are added when it is thought that giving a flexible departure time to these flights will have a positive impact on the solution. The final iteration is reached when it is known that additional flight arc copies will not help to produce an improved solution.

By using flight arc copies only when necessary, IST minimizes problem size and memory usage. This is valuable because it allows larger (e.g., more fleets and flights) and more complex (e.g., additional enhancements) problems to be solved. For example, because of memory limitations it might not be possible to directly solve an integrated fleet assignment with time windows problem and, say, the aircraft routing problem. But, because IST uses only slightly more memory than the basic fleet assignment problem, such a simultaneous solution might be possible. BARNHART et al. (1998) show that it is possible to simultaneously solve the basic fleet assignment model and the aircraft routing problem. Certainly, simultaneous solution of multiple airline scheduling problems is a current operations research trend. It is hoped that this algorithm will help make fleet assignment with time windows a part of this trend and inspire creative iterative algorithms applied elsewhere.

The IST Algorithm

As noted above, the objective of IST is to obtain the optimal solution but with fewer flight arcs, thereby minimizing memory usage. The algorithm, summarized in the flow-chart of Figure 9, begins with the creation of flight networks (Step 0). As in the basic fleet assignment problem, one arc is created for each flight in each fleet’s network. But, rather than using flight arcs of real duration,

\[
\text{arc duration}_{\text{real}} = \text{block time} + \text{turn time},
\]

each flight arc is positioned to depart at the end of its departure time window and arrive at the beginning of its arrival time window. Thus, each flight arc has a reduced duration:

\[
\text{arc duration}_{\text{reduced}} = \text{arc duration}_{\text{real}} - \text{time window width}.
\]

Figure 10, with time windows indicated by rectangles on the time line, shows how a real-duration flight arc (a) differs from a reduced-duration arc (b). Notice that, by reducing a flight’s duration in this

![Fig. 9. Time window–expanded time line with islands.](image-url)
manner, we allow the flight to make any time window-feasible connection. That is, any flight that arrives and is ready in BOS before the end of flight A’s departure window can connect to flight A; likewise flight A can connect to any flight departing from ORD after the beginning of flight A’s arrival time window. When this is done for all flights, every viable connection between pairs of flights becomes feasible. When the fleet assignment model with reduced-duration flight arcs is solved, we obtain a solution that is super-optimal, in the sense that it is a solution that is at least as good as the optimal solution to the original fleet assignment with time windows problem, because all of the flight connections of the original problem, plus many others not in the original problem, are allowed. We call the model with reduced-duration arcs the master problem, and its initial formulation is exactly the same as the basic fleet assignment model of Figure 2 (except that flight times are now reduced).

As one might expect, the solution of the master problem (Step 1) is often infeasible for the original problem. However, if we find that the master problem’s solution is feasible for the original problem, then we know we have an optimal solution to the original problem. This is because it is not possible to find a solution that is better than the one found in the master problem, because the master problem allows, at a minimum, every feasible solution to the original problem.

Step 2 of IST determines whether or not the master solution is feasible for the original problem; if it is found that the master solution is not feasible, this step provides information to change the master problem and make it more likely that a feasible solution will be found in the next iteration. The Step 2 subproblems (one for each fleet type) are as follows:

1. Create a time window flight network (as described in Section 2.1) with only the flights covered by fleet $k$ in the master solution. Assign flight arcs a cost of zero.
2. As shown in Figure 11, add backward connection arcs with penalty costs.
3. Solve this subproblem (formulation in Figure 12) directly.

Each subproblem is essentially a single-fleet fleet assignment with time windows problem. The only added complexity is the presence of backward arcs that are added to each airport’s time line, as shown in Figure 11. These backward arcs, whose purpose is twofold, connect arriving flights to earlier departing flights. So as not to explode subproblem size, we place a cap on the number of backward arcs that can emanate from any single arrival node, and an upper bound on the time span of a backward arc. One purpose of the backward arcs is to permit a feasible solution to the subproblem by allowing arriving flights to back-up and connect with earlier departing flights. The other purpose of the backward arcs is to identify problem flights that are preventing a feasible solution.

Figure 12 also shows the notation for the subprob-
Minimize $\sum_{b \in B} c_b z_b$

subject to $\sum_{n \in N_i} x_{ni} = 1 \quad \forall i \in F_k$ (1)

$$\sum_{i \in F_k} \sum_{n \in N_i} b_{1in} x_{ni} + \sum_{g \in G} b_{2g} y_g + \sum_{b \in B} b_{3b} z_b = 0 \quad \forall l \in L$$ (2)

$$\sum_{i \in F_k} \sum_{n \in N_i} d_{1in} x_{ni} + \sum_{g \in G} d_{2g} y_g - \sum_{b \in B} d_{3b} z_b \leq S_k$$ (3)

$x_{ni} \in \{0, 1\} \quad \forall i \in F_k, \forall n \in N_i$ (4)

$y_g \geq 0 \quad \forall g \in G$ (5)

$z_b \in \{0, 1\} \quad \forall b \in B$ (6)

Parameters

$F_k$ = set of flights assigned to be flown by fleet $k$ (from master solution)

$N_i$ = set of arc copies of flight $i$

$|N|_i$ = number of arc copies of flight $i$

$S_k$ = number of aircraft of fleet $k$

$G$ = set of ground arcs

$B$ = set of backward arcs

$c_b$ = penalty cost for an aircraft to use backward arc $b$

$b_{1in} = \begin{cases} 1 & \text{if copy } n \text{ of flight } i \text{ begins at node } l \\ -1 & \text{if copy } n \text{ of flight } i \text{ ends at node } l \\ 0 & \text{otherwise} \end{cases}$

$b_{2g} = \begin{cases} 1 & \text{if ground arc } g \text{ begins at node } l \\ -1 & \text{if ground arc } g \text{ ends at node } l \\ 0 & \text{otherwise} \end{cases}$

$b_{3b} = \begin{cases} 1 & \text{if backward arc } b \text{ begins at node } l \\ -1 & \text{if backward arc } b \text{ ends at node } l \\ 0 & \text{otherwise} \end{cases}$

$d_{1in} = \begin{cases} 1 & \text{if copy } n \text{ of flight } i \text{ crosses the count time} \\ 0 & \text{otherwise} \end{cases}$

$d_{2g} = \begin{cases} 1 & \text{if ground arc } g \text{ crosses the count time} \\ 0 & \text{otherwise} \end{cases}$

$d_{3b} = \begin{cases} 1 & \text{if backward arc } b \text{ crosses the count time} \\ 0 & \text{otherwise} \end{cases}$

Decision Variables

$x_{ni} = \begin{cases} 1 & \text{if copy } n \text{ of flight } i \text{ is to be flown} \\ 0 & \text{otherwise} \end{cases}$

$y_g = \text{number of aircraft on ground arc } g$

$z_b = \begin{cases} 1 & \text{if backward arc } b \text{ is used} \\ 0 & \text{otherwise} \end{cases}$

Fig. 12. IST subproblem formulation.

The major difference between this model and the formulation of Section 2.2, besides the fact that each subproblem involves only one fleet, is the objective function. Because fleet assignment costs are not an issue during this step (that is the job of the master problem), the objective function minimizes penalty costs, thereby minimizing use of backward arcs in each fleet-specific subproblem. The constraints are the same: cover each flight exactly once (constraints 1), conserve aircraft flow at each node (constraints 2), use only the available number of aircraft (constraint 3), and maintain integer aircraft flow (constraints 4, 5, and 6). If the optimal objective function value is zero, then no backward arcs are used, and we know that the master solution is feasible for this fleet type. If feasibility can be restored
to every fleet in this manner, IST terminates with the optimal fleet assignment solution (Lemma 1). However, if backward arcs are used in one or more subproblems, we must adjust the master problem (Step 3) and re-solve.

The purpose of Step 3 of IST is to change the master problem so that, in the next iteration, it will be more likely to provide a solution that is feasible for all fleets. This is accomplished by replacing some of the reduced-duration arcs with real-duration arcs, thereby removing some infeasible flight connection sequences in the current solution. Figure 13 shows what a flight looks like in a fleet's master problem network before (a) and after (b) it has been altered. Notice that, after making this change to the network, a flight that arrives and is ready in BOS before the end of flight A's departure time window can still connect to flight A (by using the last arc copy). Also, flight A can still connect to any flight that departs ORD after the beginning of flight A's arrival time window (by using the first arc copy). But, because only one of flight A's arc copies can be covered (constraints 1, Figure 5), both of those flight connections cannot be made. Some infeasible flight sequences in the reduced-duration network are now illegal, thus the revised master problem's optimal solution is more likely to be feasible for the original problem.

Once the master problem has been revised, the formulation is a mix between the basic fleet assignment formulation and the time windows formulation. This is because some flights have only one arc that must be covered, whereas other flights (let's call them problem flights) are made up of a cluster of arc copies, only one of which must be covered. So then, given the following definition for each flight i,

$$|N_{ik}| = 1 \text{ if } i \text{ is a reduced-duration flight},$$

and

$$|N_{ik}| \geq 1 \text{ if } i \text{ is a problem flight},$$

the master problem formulation is exactly the same as the fleet assignment formulation of Figure 5.

Notice that, if we replace every reduced-duration arc with real-duration copies in the manner described above, the master problem becomes equivalent to the original fleet assignment with time windows problem. Because the purpose of IST is to avoid such a large integer program, we want to update the master problem more intelligently. We do this by altering only flights that are part of an infeasible flight sequence. When solving the subproblems, pairs of these flights are identified whenever a backward arc is used to connect them. For example, in Figure 11, if backward arc 1 is used in the solution, then we know that flights A and B are problem flights, because they are flights that prevent a feasible solution from being constructed during the subproblem. After replacing the reduced-duration arcs with real-duration copies for all problem flights and for all fleets, we return to Step 1, re-solve the master problem, and continue IST until a solution is found that is feasible for all fleets.

**LEMMA 1.** If the solution to the master problem is found to be feasible for the original problem, then that solution is optimal for the original problem.

**Proof.** Suppose that we begin with the original problem (real-duration arc copies for all flights), and we then replace one or more flights' arc copies with single reduced-duration arcs. After doing this, the number of possible flight connections, and consequently the number of feasible solutions, stays the same or increases. (See discussion accompanying Figure 10.) Because the master problem is composed only of these real-duration arc copies or of single reduced-duration arcs, the set of feasible solutions to the master problem always includes, at a minimum, the set of feasible solutions to the original problem. Furthermore, because the objective function of the master problem and original problem is the same, if the optimal solution to the master problem is feasible for the original problem, then it must also be optimal for the original problem.

A key implementation detail of IST is that, during subproblem network creation, backward arcs may not be used to connect a pair of flights that have both been identified as problem flights in a previous iteration. This is necessary to ensure that the algorithm terminates in a finite number of iterations, as shown in Lemma 2. If backward arcs were defined between a pair of previously-identified problem flights, the subproblems might add no new information to the master problem, causing the algorithm to stall.

The following property can now be stated:

**LEMMA 2.** Assuming feasibility of the original problem, IST will terminate after a finite number of iter-
ations with an optimal solution to the original problem.

Proof. At each iteration, Step 2 of the algorithm either uses zero backward arcs, or it uses one or more backward arcs. If no backward arcs are used for any subproblems, then that iteration’s master problem solution is feasible and optimal for the original problem (Lemma 1). Furthermore, because backward arcs are not placed between two previously-defined problem flights, if one or more backward arcs are used, then at least one new problem flight will be identified. In the worst case, one new problem flight will be identified at each iteration (two in the first iteration), and the master problem will be equivalent to the original problem on iteration number $F$, where $F$ equals the number of flights. At this point, the master problem is guaranteed to be feasible and optimal (assuming there exists a feasible solution to the original problem).

IST Details

This section points out some specific issues that are necessary for successfully implementing the IST algorithm of Section 3.2.

Backward Arc Costs. To minimize the number of times the master problem is solved, we would like to identify as many good problem flights as possible during each subproblem. The more problem flights that we identify, the more likely it is that the next solution of the master problem will be feasible for the original problem. In contrast, if we just randomly replace the reduced-duration arcs of the master problem with real-duration copies, we might be expanding the size of the master problem without positively affecting the solution. So, to identify as many good problem flights as possible, we apply appropriate costs to subproblem backward arcs. The appropriate cost function is anything greater than linear, with respect to the time that each flight backs up. For example,

$$\text{cost}_b = (\text{backward time of arc } b)^2$$

is a good cost function. By making cost increase greater than linearly with time, the model is encouraged to choose many short backward arcs instead of a few long backward arcs. In Figure 14, if cost were linear or less and the model needed to use a backward arc, it might use arc 1, identifying flights A and D as problem flights. But, with a greater-than-linear cost function, it is more likely that arcs 2 and 3 would be used, identifying all four flights (A, B, C, D) as problem flights. With four, rather than two, problem flights identified, the next solution to the master problem is more likely to be feasible for the original problem.

Reoptimization of the Master Problem. From one iteration to the next, as the arcs of problem flights are altered, the master problem undergoes only marginal changes. So then, it would seem that we could easily use a solution from the previous iteration as an advanced start basis. Unfortunately, those seemingly minor changes to the master problem flight networks lead to major differences in the LP matrix. For example, notice the differences between Figure 15a and 15b. In 15b, three real-duration arc copies have replaced the single reduced-duration arc of 15a. The result is a network (LP matrix) that includes many different nodes (constraints) and flight and ground arcs (variables). Because many of the constraints (variables) from the previous iteration are now obsolete, the additional constraints (variables) cannot be added to the matrix in a row (column) generation fashion; the entire LP matrix must be recreated. As such, we cannot use the final basis of the previous iteration as a starting point for the next.

One option that we have explored is to create a super master problem—a master problem that includes every arc and node that might potentially be a part of the master problem. For example, the network of Figure 15 would look like Figure 16 in the super master problem. If the flight between BOS and ORD of this figure has not been identified as a problem flight, then copies 1, 2, and 3 are given a flow capacity of zero, while reduced-duration arc 0

![Fig. 14. Backward connection arcs.](image)

![Fig. 15. (a) Single reduced-duration arc and (b) three real-duration flight arcs.](image)
remains a binary variable. Likewise, if this flight is later identified as a problem flight, variable 0 is then constrained to have zero flow, and copies 1, 2, and 3 are changed to binary variables. This super master problem approach has the advantage of using the same LP matrix at each iteration, and the final basis from the previous iteration can be used for an advanced basis start. However, because this approach causes matrix size to grow tremendously, and because the advanced basis start does not improve branch-and-bound times, our implementation of this approach has not been successful in improving upon the run times of the original IST.

4. COMPUTATIONAL EXPERIMENTS

4.1 Implementation and Data

Before analyzing and evaluating our algorithms, we first detail the solution steps used to optimize the performance of our algorithms. After the flight networks have been created and preprocessed, the LP matrix is loaded, and the following steps are used to solve each mixed integer problem (MIP):

1. Aggregate the matrix using optimizer’s algebraic preprocessor.
2. Perturb problem using the optimizer’s perturbation scheme.
3. Solve the LP using dual steepest-edge simplex.
4. Remove perturbations.
5. Reoptimize using dual steepest-edge simplex.
6. Disaggregate the matrix.
7. Reoptimize using dual steepest-edge simplex.
8. Prioritize special ordered sets.

This optimization procedure uses version 3.0 of the CPLEX callable library (1993). Step 1 reduces the problem size beyond that which was achieved using our network preprocessing. Step 2 perturbs the variable bounds to improve the simplex algorithm’s performance, particularly in the face of degeneracy. Step 3 solves the LP relaxation of our MIP using dual simplex with steepest-edge pricing. Hane et al. (1995) recommend this approach for basic fleet assignment, and our tests show that this technique is also appropriate for our model with arc copies. In steps 4–7, a solution to the LP relaxation of the original problem is obtained by removing perturbations, resolving, and disaggregating the problem. In step 8 special ordered sets (SOS) of the flight coverage constraints are identified and prioritized. These SOS are groups of binary variables of which only one can have a non-zero value. Branch-and-bound uses these sets to make intelligent branching decisions. The sets are constructed by ordering the variables such that aircraft seating capacity is non-decreasing, and then dividing the variables such that half (or as near one-half as possible) of the variable fractionality lies in one set and half in the other. Branching priority is given to the sets with the largest objective coefficient gap between the first and last variables of that set.

For computational trials, we use three data sets from a major U.S. airline (listed as P1, P2, and P3 in Table II). P1 was created from an entirely different flight schedule than P2 and P3. The difference between P2 and P3 is the granularity of fleet aggregation. For each of these data sets we vary the maximum time window width and arc copy interval to analyze our model’s performance over a range of problem types and sizes. For example, we can compare LP matrix sizes, run times, and solutions of P2 with 20-minute time windows and a 5-minute interval (P2-20.5) versus P2 with 40-minute time windows and a one-minute interval (P2-40.1). A 20-minute time window means that the maximum that any flight can shift from its originally scheduled time is ten minutes in either direction.

Our fleet assignment with time windows model is implemented in C and solved on an IBM RS/6000, Model 370 workstation with 256MB RAM. All run times are reported in minutes.

4.2 Preprocessing Results

In Section 3.1, we described the three major network preprocessing steps that are used to reduce the effective problem size and to improve run times of the DST algorithm. The impact that each of these steps has on problem size is shown in Table III. Also shown in this table is the collective impact that network preprocessing has on run times. Because

<table>
<thead>
<tr>
<th>Problem</th>
<th>Flights</th>
<th>Fleets</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>1621</td>
<td>7</td>
</tr>
<tr>
<td>P2</td>
<td>2037</td>
<td>7</td>
</tr>
<tr>
<td>P3</td>
<td>2037</td>
<td>11</td>
</tr>
</tbody>
</table>

TABLE II

Primary Data Sets
the relative effect of preprocessing was equivalent over each of the three primary data sets, Table III details the effects on different time window scenarios of the same problem (P2). Problem P2–0 refers to the basic fleet assignment problem (i.e., no time windows), whereas the other problems are named according to the example instances of Section 4.1. The column “PP Used” lists the preprocessing steps that were used before creating the LP matrix; “DC” refers to the copy-deletion step, “NC” refers to node consolidation, and “I” indicates that islands were used. The “elements” metric refers to the number of non-zero elements in the LP matrix. “CPU min.” measures run times, including the time to create networks, load the problem, solve the LP, and solve the MIP with branch-and-bound.

When redundant flight copies are deleted, the number of rows, columns, and non-zero elements of the LP matrix is greatly reduced, especially in cases in which copies are placed at a narrow interval, such as P2–20.1 where copies are placed at a one-minute interval. (Deleting copies, of course, has no impact on the basic fleet assignment model because there is only a single arc for each flight in each fleet’s network.) In the two problems with flight copies at a five-minute interval, the problem size is reduced by about 40% over all dimensions. This includes the removal of two ground arcs and two nodes for each deleted flight arc. As expected, P2–20.1 is more heavily affected—the problem size shrinks by 66% over all dimensions. This is explained by the fact that, at sparsely-used airports, a narrow spacing of flight copies results in the creation of many redundant flight arcs.

Easily the most important preprocessing step is that of node consolidation. In each of the three instances with time windows, over 80% of all rows, beyond the level achieved by the copy-deletion step, are removed. Notice also that the number of columns is reduced—by about 56% in the time window instances—the result of ground arcs being merged into nodes.

Finally, we examine the impact of islands. By looking at problem size, it would seem that the removal of ground arcs has little value as a preprocessing step. Less than 600 ground arcs (columns) are removed from any of the time window problems. This is less than one ground arc per airport-fleet time line, on average. However, by limiting the number of feasible solutions, islands significantly reduce run times. In fact, problems using islands typically solved about 10–20% faster than the same problem without islands. It should be noted that more than twice as many ground arcs were removed in P2–0 than in the other three instances. This is because the number of time lines eligible for islands was much greater in the basic problem than in the time window problems. In our implementation, a time line is eligible for islands only if it consists of no more than six (consolidated) nodes.

The collective impact that preprocessing has is evident by comparing problem sizes and run times of these instances before and after preprocessing. Although P2–20.5 is 497% larger (measured in non-zero elements) than P2–0 before preprocessing, that number drops to 266% after preprocessing. By the same token, P2–20.1 is 336% larger than P2–20.5 before preprocessing, but afterward is only 201% larger. Clearly, as raw problem size increases, so does the impact of preprocessing. More importantly, although our workstation was unable to even solve an unprocessed P2–20.5, it was able to solve much faster.

<table>
<thead>
<tr>
<th>PP Used</th>
<th>Dimension</th>
<th>P2–0</th>
<th>P2–20.5</th>
<th>P2–40.5</th>
<th>P2–20.1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>rows</td>
<td>22,698</td>
<td>105,524</td>
<td>152,076</td>
<td>349,770</td>
</tr>
<tr>
<td></td>
<td>columns</td>
<td>30,961</td>
<td>155,219</td>
<td>225,047</td>
<td>521,588</td>
</tr>
<tr>
<td></td>
<td>elements</td>
<td>73,101</td>
<td>363,542</td>
<td>526,754</td>
<td>1,219,879</td>
</tr>
<tr>
<td></td>
<td>CPU min.</td>
<td>41.3</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>DC</td>
<td>rows</td>
<td>22,698</td>
<td>60,110</td>
<td>94,024</td>
<td>120,934</td>
</tr>
<tr>
<td></td>
<td>columns</td>
<td>30,961</td>
<td>87,088</td>
<td>137,959</td>
<td>178,324</td>
</tr>
<tr>
<td></td>
<td>elements</td>
<td>73,101</td>
<td>204,193</td>
<td>323,010</td>
<td>417,287</td>
</tr>
<tr>
<td>DC, NC</td>
<td>rows</td>
<td>7502</td>
<td>12,061</td>
<td>17,477</td>
<td>20,148</td>
</tr>
<tr>
<td></td>
<td>columns</td>
<td>15,720</td>
<td>38,990</td>
<td>61,377</td>
<td>77,489</td>
</tr>
<tr>
<td></td>
<td>elements</td>
<td>42,570</td>
<td>107,948</td>
<td>169,811</td>
<td>215,568</td>
</tr>
<tr>
<td>DC, NC, I</td>
<td>rows</td>
<td>7502</td>
<td>12,061</td>
<td>17,477</td>
<td>20,148</td>
</tr>
<tr>
<td></td>
<td>columns</td>
<td>14,534</td>
<td>38,435</td>
<td>61,076</td>
<td>77,102</td>
</tr>
<tr>
<td></td>
<td>elements</td>
<td>40,190</td>
<td>106,822</td>
<td>169,188</td>
<td>214,782</td>
</tr>
<tr>
<td></td>
<td>CPU min.</td>
<td>9.4</td>
<td>32.0</td>
<td>82.3</td>
<td>89.8</td>
</tr>
</tbody>
</table>

*No feasible solution could be achieved due to insufficient memory.
larger problems in less than 90 minutes after network preprocessing.

4.3 Algorithm Performance

In this section, we detail the performance of our algorithms (DST and IST) on the three data sets under different time window scenarios. More specifically, we compare run times and problem sizes, and then take a closer look at the performance of IST.

Table IV shows run times (including time to create the networks and solve the LP and branch-and-bound) for DST and IST on 15 problem instances. (Differences in run times for the same problems in Tables III and IV occur because of varying levels of computer usage for the various runs.) The first row contains the run times for the basic problem, and the remaining rows are ordered with instances containing an increasing number of flight arcs. IST is used only on instances with time windows. With a few exceptions, a problem will be solved most quickly when solved directly. The exceptions to this rule are instances with relatively narrow time windows and very narrow flight copy intervals, such as PX–20.1. IST was superior to DST for each problem under this time window scenario, because while DST was bogged down with many unnecessary flight copies, IST only had flight copies for a select few flights. This is supported in Table V, that shows problem sizes (taken at the final iteration for IST) for each of the instances. Notice that, for these PX–20.1 instances, IST matrices are only slightly larger than the zero-time window matrix, whereas DST matrices are over five times larger.

Clearly, IST shows a poor performance in solving problems with wide time windows (e.g., 40 minutes). It is weakest for these instances because, when the master problem is solved with reduced-duration flight arcs, it results in a solution that is nowhere near feasible for the original problem. Consequently, many of these reduced-duration arcs must be replaced with real-duration copies before the master problem can identify a feasible solution. This not only requires many iterations, it also makes each iteration bigger and slower. As evidence, note that the final LP matrix of P1–40.5 was 50% larger than the matrices of the 20-minute scenarios.

The IST terminated in anywhere from four to eight iterations for the 20-minute time window scenarios, whereas the 40-minute scenarios typically required a dozen or more iterations. As expected, the number of problem flights identified decreased during each iteration. A typical problem (P2–20.5) is profiled in Table VI. Notice that the master problem size (measured by the number of non-zero elements in its LP matrix) grows at each iteration—a function of the number of problem flights added. The optimal objective function value also grows at each iteration, until the solution is finally feasible in the fifth iteration. As one can see, the single-fleet subproblems are consistently solved very quickly, representing a very small percentage of total run time.

4.4 Solution Analysis

Before analyzing the fleet assignment solutions, it must be emphasized that, because a fleet assignment model is a planning tool, it is the solution, not run time, that is most important. Although not competitive with the basic model’s run times (Table IV), the time window model’s run times are acceptable. Furthermore, because the size of IST at each iteration is only slightly larger than a basic fleet assignment model, this approach has the potential to handle additional constraints and other enhancements.
that might further improve the final schedule. The following paragraphs show the impact that flexible departure times can have on fleet assignment solutions.

As discussed in Section 2.1, time window width and flight copy interval are the two parameters that can be juggled to impact solution quality (and problem size). Tables VII and VIII show the impact that these two parameters have on the three data sets, respectively. In Table VII, we compare optimal objective function values of the basic problem to scenarios allowing 20-minute and 40-minute windows, with copies placed at a 5-minute interval. Notice that, in each case, the total daily improvements are significant ($65,442 day^{-1} = $23.9 million/year) for the 20-minute windows, and the 40-minute windows allow an additional 35–68% improvement. We discount the results for the 40-minute time windows, however, because it is likely that the constant demand assumption does not hold for windows of this width.

The daily solution cost and improvement is broken down by the two major components—operating cost and spill cost. These rows show that improvements are not necessarily made up entirely of operating costs at the expense of spill costs, or vice versa; indeed it is problem specific.

Table VIII compares improvements in the optimal objective function value when using various flight copy intervals. As indicated in the first column, each of these instances use 20-minute time windows. Recall from Section 3.1 that a flight copy is always placed at the originally scheduled time. In other words, even in the 20-minute interval case, there are always at least three arc copies of every flight (before preprocessing). Notice that, as copies are placed at a finer interval, the solutions do not improve as much as one might expect. In fact, P2–20.1 results in a solution that is worse than P2–20.20 and P2–20.5! This unexpected occurrence is due to our branch-and-bound tolerance of $3000. Table VIII is certainly good news: by placing copies at a very broad interval, we can quickly solve fleet assignment with time windows problems and still obtain solutions that are nearly as good as the finest-grain case. (The one-minute interval is the finest grain case, because flights can only be scheduled to depart at whole minutes.)

Although Tables VII and VIII show that time windows permit an improved solution, the schedule planner will certainly be interested in how the actual schedule changes. When compared to the basic problem, how many flights are re-fleeted and re-timed? Table IX answers these questions for our data sets. A rather significant number of flights—between 10 and 20% of all flights—are assigned a different fleet when using 20-minute time windows. Notice that P3 has many more re-fleetings than P2, even though these problems represent the same flight schedule. This is because the number of viable re-fleetings is much greater in P3 because the fleet types are disaggregated to a finer level. In each

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Master LP Elements</th>
<th>Master Problem Time (min.)</th>
<th>Best Solutions Value</th>
<th>Subproblems Time (min.)</th>
<th>New Problem Flights</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>39,009</td>
<td>7.7</td>
<td>29,262,285</td>
<td>0.5</td>
<td>52</td>
</tr>
<tr>
<td>2</td>
<td>39,832</td>
<td>9.8</td>
<td>29,267,445</td>
<td>0.5</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>40,091</td>
<td>7.0</td>
<td>29,268,557</td>
<td>0.5</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>40,275</td>
<td>7.6</td>
<td>29,270,542</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>40,565</td>
<td>8.4</td>
<td>29,272,062</td>
<td>0.5</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Flight Copy Interval (min.)</th>
<th>P1–20</th>
<th>P2–20</th>
<th>P3–20</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>122,743</td>
<td>67,320</td>
<td>65,442</td>
</tr>
<tr>
<td>5</td>
<td>126,553</td>
<td>68,107</td>
<td>65,442</td>
</tr>
<tr>
<td>1</td>
<td>126,882</td>
<td>68,107</td>
<td>67,139</td>
</tr>
</tbody>
</table>

*No feasible solution could be achieved due to insufficient memory.
instance, the number of flights re-timed is substantially less than the number of re-fleetings. Clearly, shifting the departure time of just one or a few flights allows the model to profitably re-fleet sequences of several flight legs. After all, it is the re-fleetings, not the re-timings, that are responsible for objective function value improvements. Displayed in the bottom row is the average (absolute value) time shift for those flights that were re-timed. Because the maximum allowable shift for 20-minute windows is 10 minutes, the high numbers in Table IX (all are above 8.4 minutes) suggest that most re-timed flights moved to one of the extremes of their time window. This is not unexpected because all arc copies of a particular flight have the same cost (not including the arc at the originally scheduled time). That is, we have provided no incentives for flights to depart, for example, only five minutes away from their scheduled time instead of ten minutes. However, as mentioned in Section 2.2, the arc copy costs can be easily changed to make a flight more expensive as a function of the time that it shifts away from its originally scheduled time. The drawback of such an approach was alluded to in Section 3.1—flight copies with unequal costs might not be redundant, thus fewer arcs will be deleted.

Our final analysis, summarized in Table X, answers the question of whether or not time windows can allow an airline to fly its current schedule with fewer aircraft. To establish a base case (TW = 0), we first minimize the number of aircraft required with no time windows. (Note that, in these experiments, our objective is not to minimize costs, but rather to minimize the number of aircraft used.) Then, we solve the same problem using 20-minute time windows. By tightening flight sequences, the model was able to fly the same schedule with two fewer aircraft in each instance. Furthermore, the daily fleet assignment cost was lower despite using two fewer aircraft! The final column of Table X shows the number of flights that were re-timed to obtain this new solution. Notice that these numbers are much higher than the corresponding numbers in Table IX, indicating the importance of re-timing when trying to save aircraft. Certainly, the operational feasibility of these solutions must be verified with maintenance routing and crew scheduling models, but the potential economic impact of this result is tremendous.

5. CONCLUSIONS AND FUTURE RESEARCH

WE HAVE DESCRIBED a generalized model and solution approach for large-scale fleet assignment problems. Because their solutions directly affect flight operating costs and passenger revenues, existing models that solve the daily fleet assignment problem have found wide acceptance among airlines, and they have been well studied. However, there is a notable weakness of these models—fixed departure times. Fleet assignment models are typically solved several months before the day of departure, and in that time, it is not uncommon for small manual re-timings to be made in search of an improved fleet assignment or operational feasibility. By permitting flights’ departure times to shift within user-specified time windows, the model presented in this paper identifies the profitable re-timings optimally, rather than manually and heuristically.

The fleet assignment with time windows problem is modeled as a simple variant to the basic (i.e., no time windows) problem: each time window has flight arc copies placed at discrete intervals throughout the window. By covering only one of these arc copies with an aircraft, the model implicitly chooses the departure time of each flight. It is important to note that time window size can be defined differently for each flight, allowing the schedule planner to, for example, restrict the flexibility of shuttle flights, arrival slot-controlled flights, and bank flights.

As one might expect, the additional nodes and arcs associated with the thousands of extra flight copies make the LP matrix of our model considerably larger than that of the basic model. Consequently, network preprocessing is a necessity. Fortunately, methods described in earlier papers, such as node consolidation and islands, as well as another step designed specifically for our model—deleting redundant flight copies—can be used to reduce the problem to a tractable size. At that point, one option

| Table IX |
| Re-fleeting and Re-timing Statistics |
| Aircraft | TW = 20.5 | TW = 20.5 | TW = 20.5 |
| Number Re-fleeted | 236 | 207 | 374 |
| Percent Re-fleeted | 14.56 | 10.16 | 18.36 |
| Number Re-timed | 129 | 111 | 121 |
| Percent Re-timed | 7.96 | 5.45 | 5.94 |
| Avg. time shift | 8.84 min. | 8.41 min. | 8.54 min. |

| Table X |
| Minimizing the Number of Aircraft Used |
| Aircraft | TW = 0 | TW = 20 |
| Aircraft Required | Cost | Aircraft Required | Cost | Number Re-timed |
| P1 | 365 | 28,919,421 | 363 | 28,754,585 | 765 |
| P2 | 428 | 29,394,172 | 426 | 29,338,177 | 590 |
| P3 | 428 | 29,328,542 | 426 | 29,266,965 | 588 |
for solving the problem is a DST. This approach, in which the preprocessed problem is simply loaded and solved, usually finds a solution more quickly than our second approach—the IST. IST finds an optimal solution to the fleet assignment with time windows problem by repeatedly adding useful flight copies to a master problem that begins with only one arc for each flight–fleet pair. This approach has the advantage of using a minimal number of arc copies, performing more reliably (due to reduced memory requirements), if not more quickly, than DST on problems with many flights and fleets.

Using data from a large U.S. airline, both solution approaches were tested under various time window width and interval scenarios. DST was indeed quicker in solving most instances, but IST found the same solution by iteratively solving LP matrices only marginally larger than that of a basic fleet assignment problem. Nearly all instances were solved within three hours, an acceptable time for a planning model.

More important than run times are the solution improvements achieved via schedule flexibility. When flight times were allowed to shift by up to ten minutes in either direction, a solution improvement of at least $65,000/day over the basic case was found in each of our test problems. In these solutions, about 10–20% of the flights were reflected, whereas less than 8% of the flights required re-timing. Interestingly, time windows did not need to be discretized into fine intervals to achieve those improvements; a copy interval of five to ten minutes was sufficient in every case. In a separate analysis, it was shown that the time window model can be used to tighten the schedule and save aircraft.

In terms of solution quality, an airline can take a number of steps to improve the performance of the model. As always, improvements in the accuracy of the demand estimates and the objective function coefficients, in particular the spill cost portion, are valuable in obtaining quality fleet assignment solutions. Additionally, the new model must have properly-defined time windows for each flight so that the solution is both acceptable from a marketing perspective and operationally feasible. The first step in developing these inputs should be to define time windows for special-case flights (e.g., shuttle flights, bank flights, etc.). Then, as users become more familiar with the model, they can explore the effects of various time window widths on the remaining flights and evaluate the impacts of placing penalty costs on re-timed flights.

To continue the evolution of this model as a valuable planning tool, the airline should look to enhance this model beyond just the basic problem definition and constraints presented in this paper. For example, one necessary step is to add constraints that increase the likelihood of obtaining a feasible maintenance routing and efficient crew pairing solution. Taking this one step further, it might be possible to actually integrate two or more of these models into a single, larger model—achieving a simultaneous solution to what are traditionally thought of as separate problems. Because of memory limitations, IST would be a likely candidate to serve as the backbone solver for such a model.

Those interested in improving run times of the model might want to study various reoptimization schemes of IST. Because our goal in developing this solution approach was to minimize memory usage, we spent little time in optimizing its speed. However, there are opportunities for doing just that. For example, one may examine the possibility of only solving LPs until the final iteration, thereby avoiding master problem branch-and-bound times. Another alternative is to reoptimize the master problem at each iteration using an advanced basis start. A deeper study of these and other techniques might result in quicker solutions.

Finally, we mention an application to which our fleet assignment with time windows model has potential value: free flight. Free flight is the end objective of an evolutionary air traffic control plan designed to give airlines greater control over the flight paths flown by their aircraft. From the airlines perspective, one major benefit of this process is that block times will be reduced. However, with this reduction in flying times comes an uncertainty from the schedule planning perspective. That is, how should the schedule change to take advantage of shorter block times? Flying the existing schedule (using the same departure times) is an inefficient option, because there will likely be a great deal of slack (under-used aircraft). Our time windows model can re-time flights to tighten this slack, thereby freeing aircraft that might be used for new routes or additional departures in existing markets. Hopefully, our model, and the concepts behind our model, can be put to use in a number of settings.

REFERENCES


(Received: June 1997; revisions received: June 1998; accepted: August 1998)