Case Analysis - Managing Operations in the Time-Shared Jet Business

1 Introduction

A growing number of executives these days believe that having their own aircraft is really the best flight plan, because using commercial airlines for business trips may sometimes be time consuming and unreliable. On the other hand, a private jet is not always affordable for many companies, especially if they are of small or medium size. Hence, selling shares of aircraft to customers who could otherwise not afford their own is a rapidly growing industry (see Business Week, September 11, 1995 [11] and The New York Times, November 16, 1995 [7]). Customers become *partial owner*s of aircraft, which would provide them with an allotted number of flying hours per year. A partial owner will call the company that manages these aircraft and book a trip by specifying the departure time, departure location, destination and exclusive use information (if a customer request consists of more than one trip, the customer may ask to use the aircraft between these trips). If the customer has enough flying hours left, the company must provide an aircraft to that customer for that trip. Through the sharing of aircraft, customers avoid the high cost of ownership and other associated overheads of establishing a corporate flight department with its own maintenance staff and pilots. There are tax advantages, too, as partial owners are allowed to take a depreciation expense.

For the company which sells shares of aircraft, there are two major types of costs: operating costs (fuel, maintenance, etc.) for flying the aircraft and the penalty costs for not being able to meet some customer requests without subcontracting extra aircraft (usually at a very high cost). Hence, from a cost/profit perspective, we need to solve the following aircraft scheduling problem: find a feasible schedule for the flights of aircraft, such that the operating costs and the penalty costs are minimized. The model and algorithms we develop here are currently being used by one such company.

Briefly, the aircraft scheduling problem has the following features:

- There are n aircraft and each aircraft can serve only one customer at a time.
- Over time, demands for trips come from customers: customer j asks for a trip from a departure location $\alpha(j)$ to a destination $\beta(j)$ with departure time $dtime(j)$ and total travel time $tt(j)$. In fact, a customer may ask for a set of trips, sometimes with an exclusive use between them.
- At the beginning of the scheduling horizon, each aircraft is at a known location or is serving a customer.
- Each aircraft has a maintenance history, which provides the number of flight hours and landings remaining before the next scheduled maintenance. Additionally there

is a preset time for the next scheduled maintenance. We call these the *maintenance constraints*.

- At any point in time, there may be some trips which have been scheduled earlier to some aircraft and these schedules cannot be changed. We call these the *scheduled trips constraints*.

- If a customer request cannot be met because of unavailability of aircraft, a charter flight can be subcontracted (we assume that this is always possible). Subcontracting is quite expensive and also causes discomfort to the customers, and therefore should be avoided if possible.

- Given the current locations and status (scheduled trips and maintenance information) of aircraft, the goal is to find a schedule for the flights, without violating the maintenance restrictions, such that all customer requests are met. This should be done with a minimum total number of flight hours required. The flight hours consist of time for relocating the aircraft to departure locations (which we call *empty flight hours* or *positioning legs*) and hours of subcontracting.

The time shared jet aircraft scheduling problem differs significantly from other aircraft scheduling problems studied in the literature. Problems which are similar to this aircraft scheduling problem, but usually more restricted, have been studied under the labels of vehicle routing and machine scheduling problems. We briefly describe the results that are closest to our problem.

In the general pickup and delivery problem (GPDP) a fleet of vehicles is available with a given capacity, start location and end location for each vehicle. There are transportation requests which specify the size of the load to be transported and the origin and destination of the transportation. A set of routes has to be constructed in order to satisfy the transportation requests, while minimizing the empty moves of the vehicles. In GPDP with time windows, there is a time interval $[r_j, d_j]$ in which the service of request j must take place, where r_j is the earliest start time (or release time) and d_j is the latest end time (or due date) for the service of request j (see [20] for a survey on routing and scheduling problems with time windows). For a recent survey on the pickup and delivery problems, see [19]. A special case of GPDP with time windows is studied in [17], where each vehicle can serve at most one customer at a time. Revenue will be received if a transportation request is satisfied, but it is not required to satisfy all the requests. The objective is to construct minimum cost (maximum profit) tours for each vehicle, subject to the additional constraints of time of availability for each vehicle. A mixed integer programming formulation along with an algorithm and computational results can be found in [17].

A special case of GPDP with time windows is the multiple depot vehicle scheduling problem (MDVSP). In MDVSP, the start time of request j is r_j and the end time is d_j, so the time window $[r_j, d_j]$ is tight. Each vehicle can serve only one customer at a time (this is the full-truck-load case in GPDP). Each vehicle belongs to a "depot", which is the start location of that vehicle, and the vehicle must return to its depot after completing its route. There

are two common objectives: minimizing the number of vehicles to satisfy all the requests and minimizing the cost of empty moves. If there is more than one depot, MDVSP (with the second objective) is NP complete [4]. Several heuristic algorithms have been proposed for MDVSP (see for example [3] [4] [5] [6] [9] [10]). An exact branch and bound algorithm is proposed in [8], which uses and additive lower bounding procedure. Computational results are reported for up to 3 depots and 70 requests. In [18], MDVSP is formulated as a set partitioning problem with side constraints and the bounds for several different relaxations are compared. An exact algorithm based on column generation is proposed and computational results for up to 300 requests and 6 depots are reported.

The aircraft scheduling problem is a variant of MDVSP. Aircraft can be anywhere at any point in time, so there is no end location (our model can be modified very easily to include the end locations typical of MDVSP). Aircraft have three different types of maintenance restrictions and some pre-scheduled trips (to be considered fixed). These constraints are not present in MDVSP.

A special case of the aircraft scheduling problem, where there is only a single location for departures, arrivals and initial location of the aircraft, and no maintenance restrictions is studied as a machine scheduling problem in [2]. There are n jobs (trips) to be processed by k identical machines (aircraft). Each job j has a fixed start time r_j and a fixed end time d_j (departure and arrival times for the trips) and a processing time (total travel time). No preemption is allowed and each job has a value w_j (a function of the total travel time). An $O(n^2 \log n)$ algorithm is presented, which maximizes the value of the jobs completed by k identical machines (minimize the hours of subcontracting). It is also shown that the problem is NP-complete under the following restriction: associated with each job there is a subset of machines on which it can be processed (previously scheduled trips in the aircraft scheduling problem). For a fixed number of machines k, an $O(n^{k+1})$ algorithm is presented. We present a dynamic programming algorithm for an extension of this machine scheduling problem.

In [13], the authors study a variant of the scheduling problem discussed in [2], where each job j is available for processing only within a time interval $[r_j, d_j]$ with duration p_j and value w_j. There is a single machine and the objective is to maximize the total value of the jobs which are scheduled. This problem is NP-complete. They discuss an integer programming formulation, heuristics and an exact algorithm and report computational results for instances up to $n = 200$.

Another special case, called generalized Fixed Job Scheduling Problem (FSP) is studied in [15]. Again, there are n jobs and each job has a fixed start time and finish time and a value representing the job's priority and job class. The machines can be split up into a number of disjoint machine classes and each job class is allowed to be processed on certain machine classes. The objective is to find a schedule for a subset of the jobs of maximum total value. The authors present a complete classification of the computational complexity of two classes of combinatorial problems related to this problem, but they do not look at optimization methods. In [16], the complexity of a variant of this problem is studied, where the machines are available in specific time intervals (shifts) only and a job can be processed on a machine

as long as the interval between start time and finish time of the job is a subinterval of the shift of the machine.

Our contributions are the following. We introduce the time shared jet aircraft problem, and we believe that this new application will be of interest to researchers and practitioners. We show that the jet aircraft scheduling problem is NP-complete and discuss three special cases. We show that the second and third special cases are also NP-complete. We provide a polynomial time network flow based algorithm for the first special case and a dynamic programming algorithm for the second special case. We formulate the aircraft scheduling problem as a 0-1 integer program tailored particularly to handle maintenance constraints and pre-scheduled trips. Our goal was to utilize a commercially available package for integer programming to the extent possible and not to develop very sophisticated methods for this application per se. We also provide a heuristic to handle larger and difficult instances that are not solvable by commercial packages for integer programming. We show that for certain special cases the heuristic solution is indeed optimal and for other cases computational results show that it performs well.

The paper is organized as follows. Section 1 provides the problem description, computational complexity and discusses special cases. Section 2 develops the integer programming formulation. Section 3 describes a heuristic and the computational results are reported in Section 4. We conclude in Section 5 by discussing other real world issues and future research directions.

2 Problem Description

2.1 Notation and Constraints

We consider a predetermined time period, called *scheduling horizon*, and do the scheduling for that time period. At the beginning of the scheduling horizon, the following information will be available about the aircraft, trips and positioning legs.

Aircraft information $(i = 1, ..., n)$

$\gamma(i)$:	initial location of aircraft i
$mhour(i)$:	number of flight hours remaining before maintenance for aircraft i
$mland(i)$:	number of landings remaining before maintenance for aircraft i
$mtime(i)$:	number of actual hours remaining before maintenance for aircraft i
$mno(i)$:	label (numeric) of the trip, which is the scheduled maintenance for aircraft i. $mno(i) = 0$, if there is no scheduled maintenance for aircraft i during the scheduling horizon.

Trip information ($j = 1, ..., m$)

$\alpha(j)$:	departure location for trip j
$\beta(j)$:	destination for trip j
$dtime(j)$:	departure time for trip j
$fly(j)$:	flight time for trip j
$tt(j)$:	total travel time for trip j
$land(j)$:	number of landings for trip j
$sch(j)$:	label (numeric) of the aircraft, which is scheduled to trip j; $sch(j) = 0$ if not scheduled

Positioning leg information

$ftime(x, y)$:	flight time from city x to city y
$ttime(x, y)$:	total travel time from city x to city y
$landings(x, y)$:	number of landings during the flight from city x to city y

Sometimes a customer trip may consist of more than one flight leg. All the legs with no exclusive use will be considered as different trips. If there is exclusive use between two or more legs, these legs will be combined to a single trip j, where $\alpha(j)$ is the departure location of the first leg, $\beta(j)$ is the destination of the last leg, $fly(j)$ is the sum of the flight times for all legs and $tt(j)$ is the sum of the travel times for all legs plus the time for exclusive use. Even if there is no exclusive use, $tt(j)$ may be larger than $fly(j)$ for some trips because of possible delays at the airports before and after landing or fuel stops. $land(j)$ will typically be 1, but again due to fuel stops or exclusive use, it may be larger than 1.

Each scheduled maintenance will also be considered as a scheduled trip. For each scheduled maintenance, we create a pre-scheduled trip j with $\alpha(j) = \beta(j) =$ maintenance location, $dtime(j) =$ start time of the maintenance, $fly(j) = land(j) = 0$, and $tt(j) =$ duration of the maintenance. If aircraft i has scheduled maintenance, say $mno(i) = j$, we assume that the aircraft will not have any maintenance restrictions after $dtime(j) + tt(j)$ until the end of the scheduling horizon. Without loss of generality, we assume that all aircraft are free, i.e. not serving customers at the beginning of the scheduling horizon. If an aircraft is serving a customer at the beginning of the scheduling horizon, a pre-scheduled trip k can be created for this partially served trip, where $\alpha(k) =$ departure location of the trip, $\beta(k) =$ destination of the trip, $dtime(k) =$ beginning of the scheduling horizon, $fly(k) =$ remaining flight time for the trip, $tt(j) =$ remaining travel time for the trip, $land(j) =$ number of remaining landings for the trip.

There are two main objectives in this optimization problem:
1) minimize the cost of positioning legs, and
2) minimize the cost of subcontracting.

Since the cost is a linear function of flight hours or subcontracted hours, the first objective is equivalent to minimizing the empty flight hours and the second objective is equivalent to minimizing the subcontracted hours. We assume that each subcontracted hour costs C (a

constant) times the cost of an empty flight hour and therefore our objective is simply be to minimize the empty flight hours weighted accordingly.

The main constraints are:

1) **Meet the customer requests.** If a customer is asking for a trip from a certain location at a certain time, an aircraft (possibly subcontracted) must be ready at that location at the desired time.

2) **Do not violate the maintenance restrictions of the aircraft.** After its scheduled maintenance, an aircraft can fly at most X hours and land at most Y times before its next scheduled maintenance. Furthermore, scheduled maintenance has to be done every Z hours, no matter how many hours the aircraft has flown or how many times it has landed. The final schedule should not violate any of these maintenance restrictions (X, Y, Z are known).

3) **Do not change scheduled trips.** There may be some trips (or maintenance) which are scheduled to the aircraft in advance. The final schedule should take these initial assignments into account and not change them while scheduling the other trips.

At the beginning of the scheduling horizon, we know the remaining maintenance slack and the trips already scheduled to the aircraft. To construct a feasible schedule, we should answer two types of questions. First, which trips can be served by each aircraft and second, which pairs of trips can be consecutively served by the same aircraft. So, we create two 0-1 matrices called AT and TT ('A' stands for aircraft and 'T' for trip). The matrix AT has a row for each aircraft and a column for each trip. The matrix TT has a row and column for each trip. (These matrices are used later in the description of a polynomial time algorithm for one special case and for the creation of variables for the 0-1 integer program.) So we set

$\mathrm{AT}(i,j) = 1$, if aircraft i can serve trip j (if trip j was the only trip to be scheduled), and $\mathrm{TT}(j,k) = 1$, if trip k can be served immediately after trip j by the same aircraft.

The following cases deserve special mention:

- If (1) $mno(i) = 0$ for aircraft i and $dtime(j) + tt(j) > mtime(i)$ (i.e. it is not possible for aircraft i to serve trip j without violating the maintenance restrictions regarding the actual hours) or (2) $mno(i) = q$ and $dtime(j) < dtime(q)$ and $dtime(j) + tt(j) > mtime(i)$, then $\mathrm{AT}(i,j) = 0$.
- If trip j is scheduled to be served by aircraft i, i.e. $sch(j) = i$, then $\mathrm{AT}(i,j) = 1$ and $\mathrm{AT}(p,j) = 0$ for $p \neq i$.
- If $ttime(\gamma(i), \alpha(j)) > dtime(j)$ then $\mathrm{AT}(i,j) = 0$.
- If two trips j and k are pre-scheduled to be served by different aircraft, then $\mathrm{TT}(j,k) = \mathrm{TT}(k,j) = 0$.
- If $dtime(j) + tt(j) + ttime(\beta(j), \alpha(k)) > dtime(k)$, then $\mathrm{TT}(j,k)=0$.
- Let trips $j_1, \ldots, j_q$ be scheduled to aircraft i, such that $q \geq 2$ and $dtime(j_t) \leq dtime(j_{t+1}), t = 1, \ldots, q-1$. $\mathrm{TT}(j_t, k) = 0$ for every k such that $dtime(k) \geq dtime(j_{t+1})$,

$t = 1, \ldots, q-1$. Similarly, $\mathrm{TT}(k, j_t) = 0$ for every k, such that $dtime(k) \leq dtime(j_{t-1})$, $t = 2, \ldots, q$.

In all the other cases, we have $\mathrm{TT}(j,k)$ and $\mathrm{AT}(i,j)$ equal to 1. (The matrix TT is square, but not symmetric. If $\mathrm{TT}(j,k) = 1$, it means that trip j has pickup time earlier than trip k, and therefore we must have $\mathrm{TT}(k,j) = 0$.)

The following example will be used throughout to illustrate our problem and heuristic.

Example: There are 4 aircraft, 8 trips and 10 cities. The scheduling horizon is 800 minutes. There are two trips, 1 and 2, which are previously scheduled to aircraft 3 and 2, respectively. Only aircraft 1 has maintenance restrictions during the scheduling horizon: it can fly at most 337 minutes ($mhour(1) = 337$) and land at most 9 times ($mland(1) = 9$) before its next maintenance. Furthermore, aircraft 1 is not available for scheduling after 630 minutes ($mtime(1) = 630$).

The information about the trips is given in Table 1. Figure 1 displays the departure locations, destinations and total travel times of the trips.

Table 1: Information about the trips in Example

Trip j	Departure location $\alpha(j)$	Destination $\beta(j)$	Departure Time $dtime(j)$	Flight hours $fly(j)$	Total hours $tt(j)$	No. of landings $land(j)$	Scheduled aircraft $sch(j)$
1	2	2	210	220	250	1	3
2	3	7	650	90	120	1	2
3	9	5	298	120	150	1	0
4	6	8	35	150	180	1	0
5	8	5	293	258	288	1	0
6	8	10	385	141	411	2	0
7	4	1	14	201	231	1	0
8	6	6	188	60	90	1	0

At the beginning of the scheduling horizon, aircraft 1, 2, 3, 4 are in locations 6, 7, 2, 4, respectively. Table 2 displays the flight times between every pair of locations for possible positioning legs. In general, the total travel time between a pair of locations is longer than the flight time (due to landing delays etc.). However, to keep this example simple, we will assume that the total travel times for positioning legs are equal to the flight times, and the number of landings between every pair of cities is 1 (only at the destination).

Based on trip, aircraft and positioning leg information, we create the two matrices AT and

Figure 1: Trips

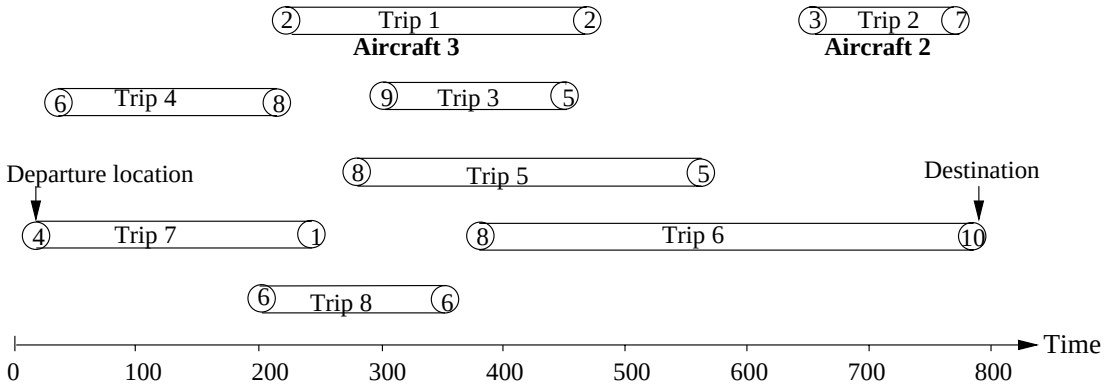


Table 2: Flight times for positioning legs in Example

City	1	2	3	4	5	6	7	8	9	10
1	0	150	190	201	108	95	108	124	67	134
2	150	0	277	342	258	242	190	228	175	30
3	190	277	0	150	192	212	90	67	124	247
4	201	342	150	0	120	150	175	134	170	319
5	108	258	192	120	0	30	153	134	120	242
6	95	242	212	150	30	0	162	150	124	228
7	108	190	90	175	153	162	0	42	42	162
8	124	228	67	134	134	150	42	0	60	201
9	67	175	124	170	120	124	42	60	0	150
10	134	30	247	319	242	228	162	201	150	0

TT.

$$\mathrm{AT} = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \qquad \mathrm{TT} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

2.2 Complexity and Special Cases

In this section, we will show that the aircraft scheduling problem is NP-complete. We consider the following three problems, which are special cases (relaxations) of the aircraft scheduling problem:

P1: There are no scheduled trips and no maintenance restrictions.
P2: There are no maintenance restrictions, but there are scheduled trips.
P3: There are no scheduled trips, but there are maintenance restrictions

We have the following complexity results for these problems.

Lemma 1 *Problem P1 is polynomially solvable.*

Lemma 2 *Problem P2 is NP-complete.*

Lemma 3 *Problem P3 is NP-complete.*

Proof of Lemma 1: Suppose there are n aircraft and m trips. This problem can be modeled as a minimum cost flow problem on a directed acyclic graph. The graph has the following nodes and arcs:

Nodes
- For each aircraft there is a node p_i, $i = 1, \ldots, n$
- For each trip there are two nodes u_j and v_j, $j = 1, \ldots, m$
- There is a dummy source node s and a dummy sink node t

Arcs
- (s, p_i) and (s, u_j) at cost 0, $i = 1, \ldots, n$, $j = 1, \ldots, m$
- (v_j, t) at cost 0, $j = 1, \ldots, m$
- (p_i, v_j) if $\mathrm{AT}(i, j) = 1$, at cost $ftime(\gamma(i), \alpha(j))$, $i = 1, \ldots, n$, $j = 1, \ldots, m$
- (u_j, v_k) if $\mathrm{TT}(j, k) = 1$, at cost $ftime(\beta(j), \alpha(k))$, $j = 1, \ldots, m$, $k = 1, \ldots, m$
- (u_j, v_j) at cost $C.fly(j)$, $j = 1, \ldots, m$

All arcs have capacity 1. In this graph, we want to find a flow of value m from node s to node t. Because of the capacities on the arcs, each arc will have either no flow, or a flow of 1 unit.

It is easy to see that there is a one to one correspondence between the minimum cost flow on this graph, and the optimum schedule for the aircraft scheduling problem P1. If there is a positive flow on the arc (p_i, v_j), this means that trip j will be the first trip served by aircraft i. Positive flow on the arc (u_j, v_k) means that trip k will be served immediately after trip j by the same aircraft. Finally, a positive flow on (u_j, v_j) means that trip j is subcontracted. □

Proof of Lemma 2: We use a reduction from Numerical Three Dimensional Matching (N3DM). Our proof follows closely the proof of Theorem 4 in [15]. See Appendix A for details. □

A dynamic programming algorithm for problem P2 is presented in Appendix B, which extends the work in [2].

Proof of Lemma 3: We will reduce the knapsack problem to this special case of the aircraft scheduling problem (no scheduled trips, but have maintenance restrictions).

The knapsack problem can be formulated as a 0-1 integer problem as follows:

$$z = \max \sum_{j=1}^{m} p_j x_j \quad \equiv \quad z' = \min \sum_{j=1}^{m} p_j (1 - x_j)$$

$$\text{s.t.} \quad \sum_{j=1}^{m} w_j x_j \leq B$$

$$x_j \in \{0, 1\} \qquad j = 1, \ldots, m$$

Given such an instance of the knapsack problem, construct the following instance of the aircraft scheduling problem:

There is one aircraft and m trips. There is a trip j corresponding to each variable x_j. Let all trips have the same departure location and destination, i.e. $l = \alpha(j) = \beta(j)$, $j = 1, \ldots, m$. Each trip has $fly(j) = p_j$, $tt(j) = p_j + w_j$ and $land(j) = w_j$, with departure time $dtime(1) = 1$ and $dtime(j) = dtime(j-1) + tt(j-1)$, $j = 2, \ldots, m$. The aircraft is initially at location l and it does not have any maintenance restrictions in terms of the number of flight hours and actual hours left before maintenance. More precisely, the aircraft has B landings, M flight hours and M actual hours before maintenance, where M is a sufficiently large number (e.g. $M > \sum_{j=1}^{m} p_j + w_j$ will do).

We want to find a schedule which minimizes the number of empty flight hours, plus the (weighted) number of subcontracted hours.

We will show that there is a 1-1 correspondence between the solution of this aircraft scheduling problem and the solution of the knapsack problem.

First note that since there is only one city, the number of empty flight hours will be zero in any schedule. Furthermore, the service time intervals of the trips are completely disjoint, i.e. the aircraft can serve any subset of these trips as long as the total number of landings of these trips does not exceed B. Then, the aircraft scheduling problem is equivalent to finding a subset $S \subset \{1, \ldots, m\}$ of the trips, such that the sum of the landings of these trips does not exceed B, i.e. $\sum_{j \in S} w_j \leq B$ and the sum of the subcontracted hours, i.e. $\sum_{j \notin S} p_j$ is minimized.

Given the optimum solution for this aircraft scheduling problem, construct the optimum solution x for the knapsack problem as follows: set $x_j = 1$, if trip j is served by the aircraft in the optimal schedule and $x_j = 0$ otherwise. Similarly, given the optimum solution for the knapsack problem, we can construct the optimum schedule (i.e. the subset S) by putting trip j into S, if $x_j = 1$. Note that z' will be equal to the objective function value of the scheduling problem. □

From the previous two lemmas, we have:

Theorem 4 *The aircraft scheduling problem is NP-complete.*

3 Formulating the Problem as a 0-1 Integer Program

The following variables are used in the formulation:

$$S_j = \begin{cases} 1, \text{ if trip } j \text{ is subcontracted} \\ 0. \text{ otherwise} \end{cases} \qquad j = 1, \ldots, m, \;\; sch(j) = 0$$

$$Z_{ijk} = \begin{cases} 1, \text{ if aircraft } i \text{ serves trip } j \text{ just before trip } k \\ 0, \text{ otherwise} \end{cases} \qquad \begin{array}{c} \mathrm{AT}(i,j) = 1, \;\; \mathrm{AT}(i,k) = 1 \\ \mathrm{TT}(j,k) = 1 \end{array}$$

$$Z_{i00} = \begin{cases} 1, \text{ if aircraft } i \text{ does not serve any trips} \\ 0, \text{ otherwise} \end{cases} \qquad i = 1, \ldots, n$$

$$Z_{i0k} = \begin{cases} 1, \text{ if trip } k \text{ is the first trip served by aircraft } i \\ 0, \text{ otherwise} \end{cases} \qquad \begin{array}{c} i = 1, \ldots, n \\ \mathrm{AT}(i,k) = 1 \end{array}$$

$$Z_{ij0} = \begin{cases} 1, \text{ if trip } j \text{ is the last trip served by aircraft } i \\ 0, \text{ otherwise} \end{cases} \qquad \begin{array}{c} i = 1, \ldots, n \\ \mathrm{AT}(i,j) = 1 \end{array}$$

In what follows we assume that variables Z_{ijk}, Z_{ij0} and Z_{i0k} will be not be created unless $\mathrm{TT}(j,k)$, $\mathrm{AT}(i,j)$ and $\mathrm{AT}(i,k)$ are 1.

Our objective is to minimize the flight hours for positioning legs and subcontracting:

$$\min \sum_{i=1}^{n}\sum_{k=1}^{m} ftime(\gamma(i),\alpha(k))Z_{i0k} + \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{m} ftime(\beta(j),\alpha(k))Z_{ijk} + C\sum_{j=1}^{m} fly(j)S_j$$

The first summation in the objective function counts (by adding over all aircraft) the number of empty flight hours from the initial location to the departure location of the first trip it serves. The second summation counts the number of empty flight hours for the positioning leg from the destination of trip j to the departure location of trip k, if these two trips are served consecutively by the same aircraft. Finally, the last summation counts the subcontracted hours. Note that since we want to minimize the amount of subcontracting (in fact completely avoid subcontracting if possible), each subcontracted hour is counted as C empty flight hours.

Optimization will be done subject to the following constraints:

$$\sum_{i=1}^{n}\sum_{j=0}^{m} Z_{ijk} + S_k = 1 \quad \text{for every } k, \text{ such that } sch(k) = 0 \tag{1}$$

$$\sum_{j=0}^{m} Z_{ijk} = 1 \quad \text{for every } k, \text{ such that } sch(k) = i \tag{2}$$

Constraint (1) ensures that each unscheduled trip will either be served by one of the aircraft, or will be subcontracted. Constraint (2) ensures that if trip k is pre-scheduled to aircraft i, it will be served by aircraft i.

If an aircraft serves trip k after trip j, trip j is either the first trip served by this aircraft, or it is served after another trip, say trip q. This is captured by (3).

$$\sum_{k=0}^{m} Z_{ijk} - \sum_{q=0}^{m} Z_{iqj} = 0 \quad \text{for every } i \text{ and } j, \text{ such that AT}(i,j) = 1 \tag{3}$$

We want to know the first trip served by each aircraft to calculate and include the flight hours for the initial positioning leg. Constraint (4) ensures that each aircraft has a first trip it served, or it did not serve any trips at all.

$$\sum_{k=0}^{m} Z_{i0k} = 1 \quad \text{for every } i \tag{4}$$

Constraints (5), (5′), (6), (6′) ensure that the maintenance restrictions are satisfied.

$$\sum_{k=1}^{m} (ftime(\gamma(i),\alpha(k)) + fly(k))Z_{i0k} + \tag{5}$$

$$\sum_{j=1}^{m}\sum_{k=1}^{m}(ftime(\beta(j),\alpha(k))+fly(k))Z_{ijk} \leq mhour(i) \quad \forall i, \text{ such that } mno(i)=0$$

$$\sum_{k:TT(k,q)=1}(ftime(\gamma(i),\alpha(k))+fly(k))Z_{i0k}+ \tag{5'}$$

$$\sum_{j:TT(j,q)=1}\sum_{k:TT(k,q)=1}(ftime(\beta(j),\alpha(k))+fly(k))Z_{ijk} \leq mhour(i)$$

$$\forall i, \text{ such that } mno(i)=q$$

Constraints (5) and (5′) ensure that the total number of flight hours of aircraft i does exceed the available flight hours before the end of the scheduling horizon or before the next maintenance. The first summation in (5) counts the number of empty flight hours from the initial location of the aircraft to the departure location of the first trip served by that aircraft plus the flight hours of that trip. The second summation counts the flight hours for the positioning leg from the destination of trip j to the departure location of trip k, plus the flight hours for trip k.

$$\sum_{k=1}^{m}(landings(\gamma(i),\alpha(k))+land(k))Z_{i0k}+ \tag{6}$$

$$\sum_{j=1}^{m}\sum_{k=1}^{m}(landings(\beta(j),\alpha(k))+land(k))Z_{ijk} \leq mland(i) \quad \forall i, \text{ such that } mno(i)=0$$

$$\sum_{k:TT(k,q)=1}(landings(\gamma(i),\alpha(k))+land(k))Z_{i0k}+ \tag{6'}$$

$$\sum_{j:TT(j,q)=1}\sum_{k:TT(k,q)=1}(landings(\beta(j),\alpha(k))+land(k))Z_{ijk} \leq mland(i)$$

$$\forall i, \text{ such that } mno(i)=q$$

Constraints (6) and (6′) verify that the total number of landings of aircraft i does not exceed the available landings before the end of the scheduling horizon or before the next maintenance.

Note that by the construction of the matrix AT, and hence by the construction of the variables, the maintenance restrictions regarding the actual hours remaining before maintenance will automatically be satisfied.

The advantage of this formulation is that we do not generate variables and constraints that will not be a part of the optimization. This advantage is obtained by preprocessing the data to create matrices TT and AT which implicitly contain all the timing information.

4 Heuristic

The problem of scheduling aircraft is NP-complete. There are two major complications in this problem which make it difficult: *maintenance restrictions* and *scheduled trips*. Because of these difficulties, our heuristic follows a "divide and conquer" approach. We consider subproblems of the original problem which are in the form P1, P2 or P3, obtain "partial" schedules and then combine these partial schedules to obtain an overall feasible schedule. After constructing a feasible schedule, we try to improve this schedule by simple exchanges.

In Section 1, we showed that P1 is polynomially solvable. In this section we describe three algorithms, Subroutine 3 and two versions of Subroutine 2, to solve problems P3 and P2 respectively. These are heuristics (since these problems are NP complete), but the first version of Subroutine 2 is optimal if there is only one aircraft.

Based on the maintenance information, aircraft can be divided into three groups:

A1: Aircraft with no maintenance restrictions (during the scheduling horizon)
A2: Aircraft with maintenance restrictions and scheduled maintenance
A3: Aircraft with maintenance restrictions, but no scheduled maintenance

The construction phase of the heuristic consists of the following steps:

STEP 1: Consider only those aircraft in groups A1 and A2, and all the (remaining) unscheduled trips. For aircraft with scheduled trips, we determine when the service of the last scheduled trip ends, and we attempt to schedule only those trips which have departure times later than this last trip. The subproblem of Step 1 is in the form of problem P1, hence can be solved by creating a graph and then finding a minimum cost flow on this graph, as discussed in Section 1.2. Due to the scheduled trips and maintenance restrictions, the graph of STEP 1 will be slightly modified, which we will discuss in the next section.

STEP 2: Consider the same aircraft as in STEP 1, and all the (remaining) unscheduled trips, but this time the scheduling will be done for the entire horizon using either Subroutine 2.1 or Subroutine 2.2. This subproblem is in the form of P2.

STEP 3: Consider the aircraft in groups A2 and A3 and all the (remaining) unscheduled trips. This subproblem (in the form P3) will be solved using Subroutine 3.

STEP 4: Consider all aircraft and remaining unscheduled trips, and schedule using Subroutine 4, which follows a greedy approach.

Steps 1, 2, 3, 4 together form a *construction heuristic*, i.e they generate a feasible schedule from scratch. By applying Steps 1, 2, 3 in different orders, followed by Step 4, we obtain different versions of the construction heuristic. The second phase of the heuristic is an *exchange procedure*, which is an *improvement step*. Given the schedule generated by the construction phase, the exchange procedure tries to change some parts of this schedule to obtain a lower cost schedule. Any trip unscheduled after this stage will be subcontracted. Figure 2 summarizes the heuristic.

Next, we provide the details of the heuristic.

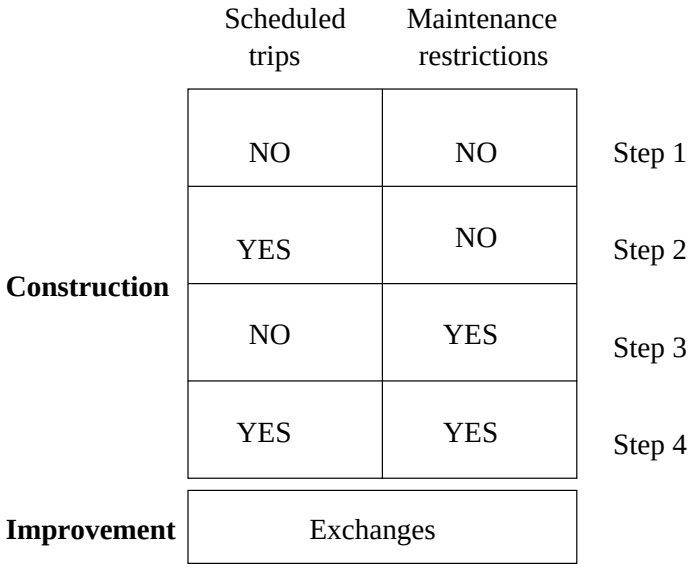


Figure 2: Heuristic

4.1 STEP 1

In STEP 1 of the heuristic, we consider all aircraft in groups A1 and A2. We consider all the (remaining) unscheduled trips. For aircraft with scheduled trips, we will only schedule new trips which have later departure time than the last trip scheduled to the aircraft. In particular, if an aircraft has scheduled maintenance, we will only schedule the trips with departure time later than the time of the maintenance of that aircraft.

First, we construct a directed acyclic graph G (similar to the graph described in Section 1.2 for problem P1), and then find a minimum cost flow, which will give us the best schedule for the set of aircraft and trips we have considered.

We construct G as follows:

Nodes
- For each aircraft create a node p_i, if the aircraft has no scheduled trips. If the aircraft has scheduled maintenance or scheduled trips, create a node s_{ij}, where j is the trip with latest departure time scheduled to this aircraft.
- For each remaining unscheduled trip create two nodes u_j and v_j.
- Create a dummy source node s and a dummy sink node t.

Arcs
- Create arcs (s, p_i), (s, s_{ij}) and (s, u_j) at cost 0
- Create arcs (v_j, t) at cost 0
- Create arcs (p_i, v_j) if $\mathrm{AT}(i, j) = 1$, at cost $ftime(\gamma(i), \alpha(j))$
- Create arcs (s_{ij}, v_k) if $\mathrm{TT}(j, k) = 1$ at cost $ftime(\beta(j), \alpha(k))$
- Create arcs (u_j, v_k) if $\mathrm{TT}(j, k) = 1$, at cost $ftime(\beta(j), \alpha(k))$

- Create arcs (u_j, v_j) at cost $C.fly(j)$

All arcs have capacity of 1 unit.

In this graph, we want to find a flow of value F from node s to node t, where F is the number of unscheduled trips. Because of the capacities on the arcs, each arc will have either no flow, or a flow of 1 unit. It is easy to see that there is a one to one correspondence between the minimum cost flow on this graph, and the optimum schedule for the aircraft and trips we have considered so far. If there is a positive flow on the arc (p_i, v_j) or (s_{ik}, v_j), this means that trip j will be served by aircraft i. Positive flow on the arc (u_j, v_k) means that trip k will be served immediately after trip j by the same aircraft. Finally, a positive flow on (u_j, v_j) means that trip j is subcontracted.

In graph G, there are $O(n+m)$ nodes and $O(m(n+m))$ arcs. The complexity of finding the minimum cost flow in this graph is at most $O(m(n+m)^2[\mathrm{m} + \log\,(n+m)])$ (see [1]).

Example: (continued)
Figure 3 shows the graph constructed in Step 1 and the minimum cost flow of value 6 (since there are 6 unscheduled trips). Based on the minimum cost flow, we schedule trips 6 and 7 to aircraft 4.

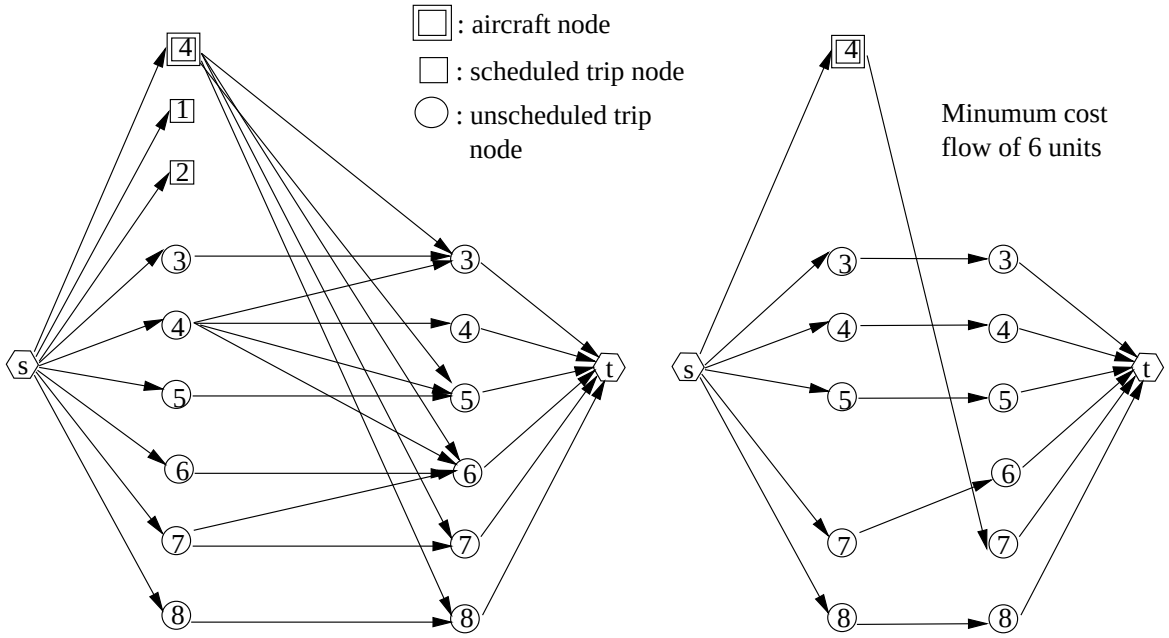


Figure 3: Graph for Step 1 and the minimum cost flow

At the end of Step 1, we have the following partial schedule:
Trip 1→ Aircraft 3
Trip 2 → Aircraft 2
Trips 6 and 7 → Aircraft 4

4.2 STEP 2

In this step, we consider the same aircraft as in STEP 1, but the main difference is in the scheduling of the aircraft, which already have some scheduled trips. In STEP 1, for aircraft with scheduled trips, we only tried to schedule those trips which have a departure time later than the last trip already scheduled to the aircraft. In this part, we do the scheduling for the entire horizon using a shortest path approach. We describe two (alternative) algorithms with similar flavor.

Subroutine 2.1

Scheduling will be done one aircraft at a time, so the order in which the aircraft is scheduled matters. We give priority to the aircraft with largest number of scheduled hours currently. For each aircraft in groups A1 and A2, possibly with scheduled trips $j(1), \ldots, j(c)$, we create a directed acyclic graph and solve a shortest path problem. The graph has the following nodes and arcs:

Nodes
- Create a node p_i if the aircraft is in group A1
- Create a node p_{ij} if the aircraft is in group A2, where j is the trip corresponding to the scheduled maintenance of this aircraft.
- Create nodes $s_{j(1)}, \ldots, s_{j(c)}$ for each trip scheduled to this aircraft (assume that if trip $j(k)$ has earlier departure time than trip $j(q)$, then $k < q$). For aircraft in group A2, consider only the scheduled trips with departure time later than the scheduled maintenance.
- Create a node v_j for each remaining unscheduled trip j.

Arcs
- Create arcs $(p_i, s_{j(1)})$ or $(p_{ij}, s_{j(1)})$ at cost 0.
- Create arcs (p_i, v_j) if $\mathrm{AT}(i,j) = 1$ and $dtime(j) + tt(j) \leq dtime(s_{j(1)})$, at cost $-C.fly(j) + ftime(\gamma(i), \alpha(j))$.
- Create arcs (p_{ij}, v_k) if $\mathrm{TT}(j,k) = 1$, at cost $-C.fly(k) + ftime(\beta(j), \alpha(k))$.
- Create arcs (v_j, v_k) if $\mathrm{TT}(j,k) = 1$, at cost $-C.fly(k) + ftime(\beta(j), \alpha(k))$.
- Create arcs $(s_{j(q)}, s_{j(q+1)})$ at cost 0, $q = 1, \ldots, c-1$.
- Create arcs $(s_{j(q)}, v_k)$ if $\mathrm{TT}(j(q), k) = 1$, at cost $-C.fly(k) + ftime(\beta(j(q)), \alpha(k))$, $q = 1, \ldots, c$.

Now we want to find shortest paths from p_i (or p_{ij}) to $s_{j(1)}$, from $s_{j(1)}$ to $s_{j(2)}$, ... , from $s_{j(c-1)}$ to $s_{j(c)}$ in this graph. Each node on a path, other than the first and the last nodes, corresponds to an unscheduled trip (see Figure 4). We will schedule those trips to aircraft i. Note that those trips will be served by the aircraft in exactly the same sequence as they appear on the paths.

Each graph we create has at most $O(m)$ nodes and $O(m^2)$ arcs. The complexity of finding the shortest path in such a graph is $O(m^2)$. Since we create the graph and find the shortest path once for each aircraft we consider in this step, the complexity of Step 2 is $O(nm^2)$.

Example: (continued)
After Step 1, there are three aircraft, namely, 2, 3 and 4, which have scheduled trips but

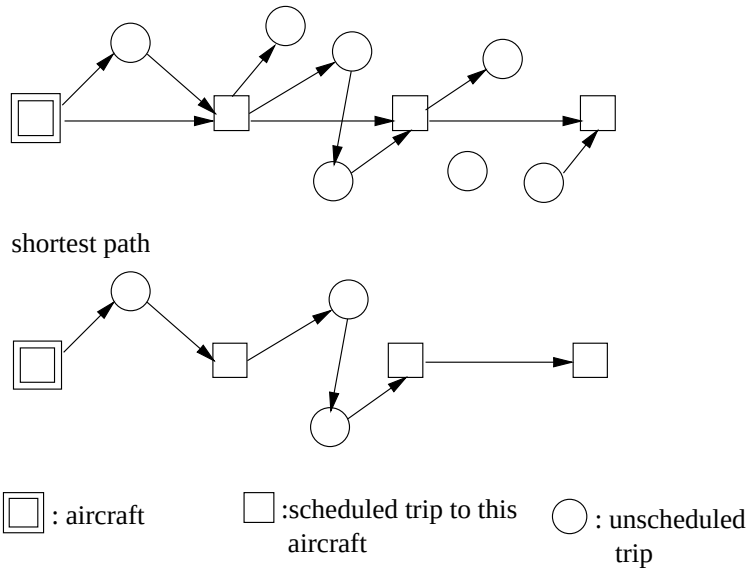


Figure 4: Graph for an aircraft (in group A1) in Subroutine 2.1 and the shortest path

no maintenance restrictions. After constructing a graph for each of these aircraft and the remaining trips, we are able to schedule one more trip to aircraft 2. Since the schedules of the other aircraft did not change, in Figure 5 we present the graph and the shortest path for aircraft 2 only.

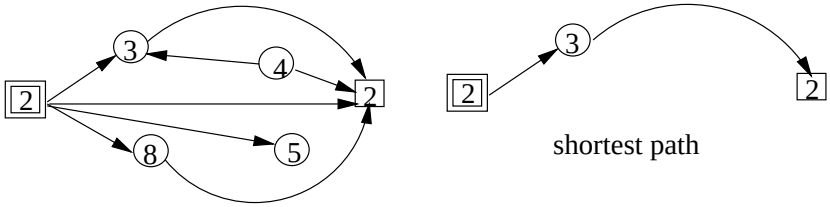


Figure 5: Graph for aircraft 2 in Subroutine 2.1, and the shortest path

The only node on the shortest path corresponding to an unscheduled trip is 3, so we schedule trip 3 to aircraft 2. At the end of Step 2, there are three trips, namely 4, 5 and 8, yet to be scheduled.

Subroutine 2.2

Now we will describe another algorithm, which is an alternative to Subroutine 2.1. Instead of optimizing one aircraft at a time, we will optimize all aircraft in groups A1 and A2 simultaneously by considering the departure times of the currently scheduled trips. Again, we create a graph similar to the one in Subroutine 2.1, but this time we consider all aircraft in groups A1 and A2 and all trips at the same time. This graph is the union of the graphs generated in Subroutine 2.1. Using the same notation as in Subroutine 2.1, the algorithm is as follows:

Step 2.2.1: Following the steps in Subroutine 2.1, create a graph by considering all the aircraft and trips at the same time.

Step 2.2.2: Sort the scheduled trips in increasing order of departure times.

Step 2.2.3: Pick the scheduled trip with the earliest departure time, which is not considered yet (by this algorithm), say trip j, and let s_j be the node in the graph corresponding to that trip. Consider the aircraft scheduled for that trip, say aircraft i. If there is no other trip scheduled to aircraft i before trip j, let $u = p_i$. Otherwise let k be the trip with latest departure time scheduled to aircraft i before trip j, and let $u = s_k$. Find a shortest path from u to s_j in this graph. Each node on this path, other than the first and the last nodes, corresponds to an unscheduled trip. Remove these nodes (except s_j) from the graph and schedule the corresponding trips to aircraft i.

Step 2.2.4: If there are still unscheduled trips and other scheduled trips which are not considered yet (by this algorithm), repeat Step 2.2.3.

In graph G, there are $O(n+m)$ nodes and $O(m(n+m))$ arcs. Complexity of Step 2.2.2 is $O(m \log m)$. The complexity of Step 2.2.3 is $O(m^2)$. Step 2.2.3 will be repeated at most m times, hence the complexity of Subroutine 2.2 is at most $O(m^3)$.

4.3 STEP 3

We consider the aircraft in groups A2 and A3 and all the (remaining) unscheduled trips. The idea is first to solve a relaxation of this subproblem, as if there are no maintenance restrictions. Then, if there are violated maintenance restrictions, "unschedule" some of the trips scheduled in the relaxed solution. We will solve this subproblem using the following algorithm.

Subroutine 3:

Step 3.1: Solve the relaxation of this subproblem as if there are no maintenance restrictions (using the minimum cost flow approach of STEP 1). Label all the aircraft as "not verified". Label all the trips scheduled in the relaxed solution as "tentatively scheduled", and all the remaining trips as "unscheduled".

Step 3.2: While there are aircraft with violated maintenance restrictions in the relaxed solution, pick such an aircraft, say aircraft i.

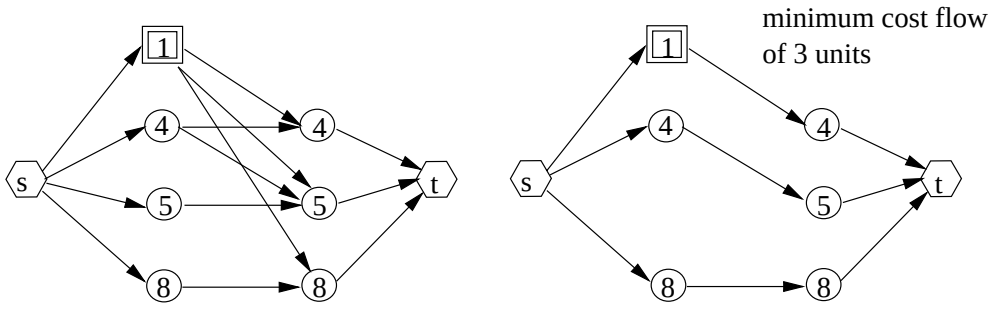
Step 3.2.a: While aircraft i has violated maintenance restrictions:
Let j be the trip with the latest departure time tentatively scheduled to this aircraft. Label trip j as "unscheduled".

Step 3.2.b: Label aircraft i as "verified" and all the tentatively scheduled trips to aircraft i as "scheduled". Label all the remaining tentatively scheduled trips as "unscheduled". Consider all the aircraft which are labeled as "not verified" and all the unscheduled trips. Solve the relaxation of this subproblem, label all trips scheduled in the relaxed solution as "tentatively scheduled" and repeat Step 3.2.

In Subroutine 3, we will solve the minimum cost problem (as in Step 1) at most once for each aircraft, hence the complexity of this subroutine is at most $O(mn(n+m)^2[m+\log(n+m)])$.

Example: (continued)
After the first two steps of the heuristic, the only aircraft with no scheduled trips (but with maintenance restrictions) is aircraft 1. We construct the graph for aircraft 1 and for the remaining unscheduled trips, 4, 5 and 8 (see Figure 6).

Figure 6: Graph for Step 3 and the minimum cost flow of 3 units



Based on the minimum cost flow, we tentatively schedule trips 4 and 5 to aircraft 1 (step 3.2 of Subroutine 3). When we check the maintenance restrictions, we see that the total flight time for the trips (tentatively) scheduled to aircraft 1 (trips 4 and 5) is 408 minutes, but aircraft 1 can fly at most 337 minutes. Since the maintenance restrictions of aircraft 1 are violated, we pick the last trip tentatively scheduled to this aircraft, namely trip 5, and unschedule it (Step 3.2.a of Subroutine 3). After this modification, all maintenance restrictions are satisfied, so aircraft 1 is "verified" and trip 4 is scheduled to aircraft 1 (Step 3.2.b of Subroutine 3). This is the only trip we could schedule in Step 3, so at the end of Step 3 of the heuristic, trips 5 and 8 are still to be scheduled.

4.4 STEP 4

The remaining trips will be scheduled to the aircraft by a greedy approach using the following algorithm.

Subroutine 4:

Step 4.1: For each aircraft calculate the number of flight hours and landings left before scheduled maintenance based on previously scheduled trips.

Step 4.2: For each aircraft, calculate the number of hours still available for scheduling during the horizon. This shows the 'utilization' of the aircraft.

Step 4.3: Pick the unscheduled trip with the largest flight time. Considering current schedules and maintenance information, find all aircraft which can be scheduled to this trip. Among such aircraft (if any), choose the one with smallest number of available hours for scheduling (based on Step 4.2 above) and schedule this trip to that aircraft. Update the

maintenance information and available hours left for that aircraft. If there are still unscheduled trips, repeat Step 4.3, otherwise stop.

In terms of complexity, step 4.3 of Subroutine 4 dominates Steps 4.1 and 4.2. Step 4.3 will be executed at most once for each trip and its complexity is $O(m)$. Hence, the complexity of Subroutine 4 is at most $O(m^2)$.

Example: (continued)
In Step 4 of the heuristic, we were not able to schedule any of the remaining unscheduled trips (5 and 8), so the schedule did not change.

4.5 Exchange

The idea of the exchange procedure is to change the schedules of some trips, as long as there is a decrease in the cost.

Step 1: Try to decrease the cost of subcontracting. Pick an unscheduled trip (if any), say trip j. Try to find another trip, say trip k, which is scheduled (by the construction heuristic) to aircraft i, such that we can schedule trip j in place of trip k and we can (possibly) reschedule trip k to another aircraft. Keep the schedule with the lowest cost.

Step 2: Try to reschedule some trips to another aircraft. Pick a trip, say trip j, which is scheduled (by the construction heuristic) to aircraft i. Try to schedule this trip to another aircraft (without changing the schedules of other trips), such that the cost of the new schedule will be lower.

Step 3: Try to change the schedules of two trips. Pick two trips j and k, such that trip k is scheduled to aircraft i (by the construction heuristic) and trip j is not pre-scheduled to any aircraft. Try to schedule trip j to aircraft i, and trip k to another aircraft. Keep the schedule with the lowest cost.

At each step of the exchange procedure, the constraints regarding the scheduled trips, maintenance restrictions and meeting the demand on time should be met. Note that as long as there is improvement in the schedule, the steps of the exchange heuristic can be repeated in any order. In our computational experiments, we repeated each step of the exchange heuristic at most three times. The complexity of the second step is $O(m^2)$ and the complexity of the first and last steps is at most $O(m^3)$, hence the complexity of the exchange heuristic is $O(m^3)$.

Example: (continued)
We could not obtain any improvement in the schedule after the exchage procedure. So, the schedule found by the heuristic is the following:
Trip 4 $\rightarrow$ Aircraft 1
Trips 2 and 3 $\rightarrow$ Aircraft 2
Trip 1 $\rightarrow$ Aircraft 3
Trips 6 and 7 $\rightarrow$ Aircraft 4
Trips 5 and 8 will be subcontracted. It turns out that this is the optimum schedule for this

example.

The complexity of our heuristic is $O(mn(n+m)^2[m+\log(n+m)])$, determined by the complexity of Step 3.

Proposition 5 *Suppose TT(j,k) = 0 for every j,k, such that sch(j) = 0 and sch(k)* $\neq$ *0. Furthermore, suppose that each aircraft has either no maintenance restrictions, or is scheduled for maintenance during the scheduling horizon (groups A1 and A2). Then, STEP 1 of the heuristic finds the optimum solution.*

Proof: In this case, all aircraft and unscheduled trips will be considered in STEP 1 of the heuristic, hence STEP 1 will find the optimum solution.

Note that the conditions of Proposition 5 will hold if almost all the scheduled trips are at the beginning of the scheduling horizon, and all the unscheduled trips are later than the scheduled ones. In other words, pre-scheduled and unscheduled trips are separated in time, and they are not randomly distributed over the scheduling horizon. We believe that this will be the case in most real life situations.

5 Computational Results

To test the model and the heuristic, we created three different sets of test problems randomly based on patterns from real data. We used problem names such as 20_30, where 20 is the number of aircraft and 30 is the number of trips. In all cases, the scheduling horizon is 3 days (4320 minutes) and all distances are Euclidean. In each instance, the total number of locations are equal to the number of aircraft plus the number of trips, randomly selected from a 20 by 20 grid. The flight time between two adjacent points on the grid is assumed to be 30 minutes. For positioning legs, we assume that there is a fuel stop (which adds 30 minutes and one landing) for every 3 hours of flight. Trips have a 20 % chance of having a long total travel time (to model exclusive use), and 80 % chance of being a single leg. From the locations that have been selected, for each trip, we randomly select a departure and a destination (which could be the same for 'long' trips) location. The departure time for a trip is randomly selected between 0 and 4000 minutes. The total travel time, flight time and number of landings for single leg trips are the same as those of a positioning leg between the departure and destination locations. The 'long' trips are obtained by combining either 2 or 3 single leg trips and adding a random time for exclusive use. In test sets 2 and 3, a certain percentage of these trips is randomly assigned to aircraft after checking for feasibility. (These are the pre-scheduled trips.) The initial location of each aircraft is randomly selected from the locations selected earlier. We also randomly select the aircraft that have maintenance restrictions for test sets 1 and 3. For these aircraft, we then set (randomly) the number of landings, flight time and actual time remaining. In test set 3, among aircraft with maintenance restrictions we randomly select some and assign a scheduled maintenance.

In the first test set, there are no scheduled trips, but 20% of the aircraft have maintenance restrictions. In the second test set, aircraft do not have maintenance restrictions, but 25% of the trips are scheduled trips. In the third test set, 10% of the aircraft have maintenance restrictions and 25% of the trips are scheduled trips. All scheduled trips have departure time in the first half of the scheduling horizon. In the real world application that motivated this work, the optimization is done on a rolling horizon and most of the pre-scheduled trips are at the beginning of the scheduling horizon (most of which were perhaps scheduled in the previous optimization run). The scheduling horizon was always less than three days. During the scheduling horizon, less than 10% of the aircraft would have maintenance restrictions and almost all the aircraft with maintenance restrictions would have scheduled maintenance, which provides these aircraft with a large number of flight hours and landings. In this sense, we believe that the third test set is the closest to real life.

Results of the experiments are listed in tables 3- 5. While solving the problems by Cplex, we used primal simplex, the node selection strategy was best bound and the pricing strategy was hybrid reduced-cost and Devex pricing. Each data point is found by taking the average of five problems which are generated randomly using different seeds. The column "Cplex time" denotes the time spent by Cplex 3.0 on a Sparc 20 workstation to find the optimum solution (in seconds). If there is a '*' next to Cplex time, this means that at least one of the five problems could not be solved to optimality by Cplex, because either the memory was exhausted or the time limit was exceeded. (In this case the average optimal was taken over the instances that were solved.) The column "% optimality gap" is the percentage difference between the heuristic and the optimum solution. Even for the largest instances, the heuristic did not take more than a few seconds, so we do not report its time. Each instance was solved using four different versions of the heuristic and the column "heuristic solution" denotes the best of the solutions found. In the first two versions, STEP 1 is followed by STEP 2 and STEP 1 follows STEP 2, respectively. In the last two versions, STEP 3 is followed by STEP 2 and STEP 3 follows STEP 2, respectively. In all versions, the initial steps are followed by STEP 4 and then by the exchange procedure.

The results of the first set are summarized in Table 3 (12 groups containing 5 problems each). The results of the second set are summarized in Table 4 (13 groups of 5 problems each). The results of the third set are summarized in Table 5 (16 groups of 5 problems each). In each test set, larger problems could not be solved by Cplex because either the memory was exhausted or the time limit was exceeded.

Computational experiments show that for small and medium size problems (up to 20 aircraft and 50 trips), the integer programming formulation can be efficiently solved by Cplex and the optimum solution can be found in reasonable time. We also observed that problems with scheduled trips could be solved much faster compared to the problems of the same size with few or no scheduled trips. The most difficult class of problems were in the first test set, where aircraft have maintenance restrictions but no scheduled trips or scheduled maintenance. For the problems that could be solved to optimality by Cplex, we computed the *integrality gap*, which is the difference of the optimum solution and the LP relaxation as a percentage of the optimum solution. For 31 groups, the average integrality gap was less

Table 3: No scheduled trips, 20% of the aircraft have maintenance restrictions

Problem	**heuristic solution**	**optimum solution**	**Cplex time**	**% optimality gap**
10_20	3104.8	3094.2	1.91	0.4
10_30	17661	17649	9.42	0.2
10_40	35049	34762	112.44	1.3
20_30	1980.6	1928.2	645.88	2.7
20_40	5040.4	4953.8	181.1	2.3
20_50	5621	5544.6	3632.9	1.9
20_60	15578	15418	11715.1*	1.5
30_40	9044.3	8825.5	6662.8*	3.2
30_50	13124	13034	44883.8	1.5
30_60	5902.4	5756.8	21650.6	3.4
30_70	8368	8219.8	25034*	2.6
30_80	17568	17139	12357*	0.4

Table 4: No maintenance restrictions, 25% of the trips are pre-scheduled

Problem	**heuristic solution**	**optimum solution**	**Cplex time**	**% optimality gap**
10_20	3851.2	3826.6	0.69	0.7
10_30	17316.4	17284.6	1.71	1
10_40	28168.2	28126.8	3.52	0.3
20_30	3145.8	3122.8	6.43	0.7
20_40	6896.2	6865.6	29.01	0.9
20_50	6007.8	5866.6	87.03	2.7
20_60	13880.2	13675.6	144.59	1.9
30_40	9283	9138.6	1780.8	2
30_50	7980.4	7980.4	132.37	0
30_60	15806.8	15640.4	374.12	1.6
30_70	8599.8	8369.4	950.4	2.9
30_80	11810	11558.8	1660.25	2.1
30_90	14464.4	14151.2	3068.76	2.2

Table 5: 10% of the aircraft have maintenance restrictions, 25% of the trips are pre-scheduled

Problem	**heuristic solution**	**optimum solution**	**Cplex time**	**% optimality gap**
10_20	6719	6700	0.68	0.6
10_30	18409.4	18409.4	2.63	0
10_40	47439.2	47219	2.1	0.6
20_30	3443	3432.2	5.93	0.3
20_40	6619.2	6568	55.1	1.2
20_50	6659	6497.8	92.68	2.9
20_60	13753.5	13528.3	129.8	2.5
20_70	25456.5	25141.5	164.78	1.5
30_40	5311.75	5219.75	48.11	2.5
30_50	14687.8	14490.3	188.92	1.4
30_60	7853.4	7637	392.82	3.1
30_70	11251.8	11120.8	762.49	1.2
30_80	14056.2	13829.6	1444.57	2
30_90	16886.6	16424.8	3261.72	2.7
30_100	28678.8	28028.5	3890.11*	2.9

than 1%, for 8 groups it was between 1% and 5%, and for one group it was larger than 5%. Among all the problems that could be solved to optimality by Cplex the largest integrality gap was 28.57%.

For problems with 10 aircraft, the heuristic solution was within 1% of the optimum, and for problems with 20 and 30 aircraft, the heuristic was within 2% of the optimum on the average. Only in 3 groups out of 41, the heuristic performance was above 3% of the optimum (the maximum being 3.4%). In all cases, *the heuristic never subcontracted more trips than the optimum solution.*

6 Conclusions and Future Research

We have formulated a new aircraft scheduling problem as a 0-1 integer program, showed its complexity, studied several special cases and developed a heuristic. We generated random problems based on real data and tested both the integer programming formulation and the heuristic on these problems. The integer program is solved using Cplex. The heuristic has the advantage of having a very short running time and it performs well while finding provably optimal solutions in certain special cases. The average deviation of the heuristic solution from the optimum (over the 205 problems) was 1.6%. In all cases, the heuristic never subcontracted more trips than the optimum solution.

We mention two extensions of our model. (1) Suppose there are k types of aircraft, type 1 being the most expensive (in terms of the operating costs) and type j being more expensive than type $j+1$, $j = 1, \ldots, k-1$. Each customer i is a partial owner of a particular type $t(i)$ of aircraft. Any trip requested by customer i must be scheduled to an aircraft of type $t(i)$ or better (or must be subcontracted). This problem can be solved by the following approach. Let j be a type $t(i)$ trip, if it is requested by customer i. Consider all type 1 trips and schedule them to type 1 aircraft. Iteratively, consider all type t trips and aircraft, $t \leq k$, where all the trips with type less than t are already pre-scheduled to the aircraft, and schedule the type t trips. This provides the flexibility of scheduling a type t trip to a better type of aircraft (usually costs much less than subcontracting a type t aircraft), if there are not enough type t aircraft available or if all type t aircraft are far away from the departure location of this trip. (2) After serving its last trip, each aircraft must have enough flight hours and landings left to reach the closest maintenance location. We have adapted our formulation and heuristic to accomodate these extensions for the application that motivated this research.

We have assumed that all the trip requests are known at the beginning of the scheduling horizon. In real life, requests for trips may not be known in advance and arrive over time, so the demand is stochastic. Furthermore, we have assumed that the departure times are fixed (a reasonable assumption in this application). In a different model, customers would request to leave (or reach their destination) within a time interval. While this provides some flexibility in terms of scheduling, it makes the problem more complicated.

As part of the real world implementation, we have created a duty roster for the crew members and considered crew scheduling issues. These are presented in [14].

Appendix

A Proof of Lemma 2

We will prove the lemma by a reduction from N3DM. The construction is very similar to the proof in [15].

Instance of N3DM: Integers t, d and a_t, b_t, c_t for $i = 1, \ldots, t$, satisfying the following relations:

$$\sum_{i=1}^{t}(a_i + b_i + c_i) = td \quad \text{and} \quad 0 < a_i, b_i, c_i < d \ \text{ for } i = 1, \ldots, t.$$

The goal is to find permutations ρ and σ of $\{1, \ldots, t\}$, such that:

$$a_i + b_{\rho(i)} + c_{\sigma(i)} = d \ \text{ for } \ i = 1, \ldots, t.$$

It is well known that N3DM is NP complete [12]. Given an instance of N3DM, we define the following quantities:

$A_i = i$ for $i = 1, \ldots, t$
$B_j = t + j$ for $j = 1, \ldots, t$
$X_{ij} = 2t + (i-1)t + j$ for $i, j = 1, \ldots, t$
$S = t^2 + 2t$
$T = S + 2d + 1$.

An instance of P2 with t^2 aircraft and 3 locations can be constructed as follows: aircraft $1, \ldots, t$ are initially located in location 1 and aircraft $t+1, \ldots, t^2$ in location 2. The flight time between every pair of locations is a positive number. The start and end times of the trips are given in the form (r_j, d_j) below (r_j denotes the departure time and $d_j - r_j$ is the total travel time of trip j). For each trip, let the flight time be equal to the total travel time.

The following trips have departure location 1 and destination 3.

$(0, A_i)$, for $i = 1, \ldots, t$

The following trips have departure location 2, destination 1, and the k-th trip in this group is pre-scheduled to aircraft k.

$(S + d - c_k, T)$ for $k = 1, \ldots, t$

The following trips have departure location 2 and destination 3.

$t - 1$ times $(0, B_j)$ for $j = 1, \ldots, t$
$(S + a_i + b_j, T - 1)$ for $i, j = 1, \ldots, t$

The following trips have departure location 3 and destination 1

(A_i, X_{ij}) for $i, j = 1, \ldots, t$
(B_j, X_{ij}) for $i, j = 1, \ldots, t$
$t^2 - t$ times $(T - 1, T)$

The following trips have departure location 1 and destination 2.

$(X_{ij}, S + a_i + b_j)$ for $i, j = 1, \ldots, t$

We now show that there exists a feasible solution for N3DM, if and only if all aircraft are serving customers during the interval $[0, T]$ and there are no empty flights and no idle time. (Note that all trips start and end during the interval $[0, T]$.)

Suppose that there exists a feasible schedule for a *subset* of trips, such that all aircraft are busy serving trips during the interval $[0, T]$, i.e. there are no empty flights and no idle time. (The other trips will be subcontracted.) Trips $(0, A_i)$ must be scheduled to some aircraft in location 1, since these trips depart from location 1 (there are t such trips and t aircraft in location 1). These trips must be followed directly by some of the trips (A_i, X_{ij}). Similarly, the $t^2 - t$ trips $(0, B_j)$ must be scheduled to the $t^2 - t$ aircraft in location 2, followed by some of the trips (B_j, X_{ij}).

As there are exactly t^2 trips that end at time T, all of them must be served. Trips $(S + d - c_k, T)$ are pre-scheduled to aircraft k, $k = 1, \ldots, t$. Therefore trips $(T - 1, T)$ must be served by aircraft $t + 1, \ldots, t^2$, following some of the trips $(S + a_i + b_j, T - 1)$.

There is no idle time or empty flights, so we should schedule the remaining trips $(X_{ij}, S + a_i + b_j)$ to the aircraft. Therefore we get schedules of the form

$$(0, A_i)(A_i, X_{ij})(X_{ij}, S + a_i + b_j)(S + d - c_k, T)$$

for aircraft $1, \ldots, t$, where each i, $1 \leq i \leq t$, occurs exactly once. We get schedules of the form

$$(0, B_j)(B_j, X_{ij})(X_{ij}, S + a_i + b_j)(S + a_i + b_j, T - 1)(T - 1, T)$$

for aircraft $t + 1, \ldots, t^2$, where each j, $1 \leq j \leq t$, occurs exactly $t - 1$ times. Hence, among the trips $(X_{ij}, S + a_i + b_j)$ that are served by aircraft $1, \ldots, t$, each i and each j occurs exactly once.

From the schedule of aircraft $1, \ldots, t$, we have $S + a_i + b_j = S + d - c_k$, i.e. $a_i + b_j + c_k = d$. So we define $\rho(i) = j$ and $\sigma(i) = k$ whenever trip $(X_{ij}, S + a_i + b_j)$ is served by aircraft $k \leq t$.

Conversely, given a feasible solution for N3DM, a feasible solution for the aircraft scheduling problem can be found, which is clearly optimal, since there will be no idle time and no empty flights during the interval $[0, T]$. □

B Dynamic Programming Algorithm for Problem P2

Here we present a dynamic programming algorithm for problem P2 of Section 1.2. The fixed start and end times of the trips provide a natural decomposition of the scheduling horizon for a dynamic programming recursion. The status of the aircraft at any point in time can be described using a n-vector v, such that:

$$v_i = \begin{cases} 0 & , \text{ if aircraft } i \text{ did not serve any trips yet, waiting idle at location } \gamma(i) \\ -j & , \text{ if aircraft } i \text{ finished serving trip } j \text{ and waiting idle at location } \beta(j) \\ j & , \text{ if aircraft } i \text{ is currently serving trip } j \\ m+1 & , \text{ if aircraft } i \text{ finished serving its last trip and is not going to serve any other trips} \end{cases}$$

Note that if $sch(j) = i$, then $v_q = j$ or $v_q = -j$ is not possible for $q \neq i$.

The status vector can change only at the departure or arrival times of the trips (since we minimize the empty flight hours, an aircraft will make an empty move to another location, only if that location is the departure location of a trip and the aircraft will serve that trip). So without loss of generality we may assume that there are $2m$ distinct events, which are departures or arrivals of the trips. Let P_t be the collection of possible status vectors at the time of event t. If $v \in P_t$:

$v_i = -j$ is possible only if $dtime(j) + tt(j) \leq t$

$v_i = j$ is possible only if $dtime(j) \leq t$ and $dtime(j) + tt(j) > t$

Let w be the status vector at the time of event t and v be the status vector at the time of event $t+1$. Let c_{wv} be the cost of moving from status vector w to status vector v from one event to the next (finish the service of a trip, or start the service of a trip). Noting that the status of at most one aircraft can change from one event to the next, $c_{wv} = \infty$ if v and w differ for more that one aircraft. If v and w differ for at most one aircraft, say aircraft i, define c_{wv} as follows:

$c_{wv} = ftime(\gamma(i), \alpha(j))$, if $w_i = 0,\ v_i = j$ and $\mathrm{AT}(i,j) = 1$

$c_{wv} = ftime(\beta(j), \alpha(k))$, if $w_i = -j,\ v_i = k$ and $\mathrm{TT}(j,k) = 1$

$c_{wv} = -C.fly(j)$, if $w_i = j,\ v_i = -j$

$c_{wv} = 0$, if $w_i = -j$ or $0,\ v_i = m+1$

$c_{wv} = 0$, if $v = w$

$c_{wv} = \infty$, otherwise

Let $f_t(v)$ be the minimum cost of having status vector v at the time of event t. Let $f_0(v_0) = 0$ for $v_0 = (0, 0, \ldots, 0)$ and $f_0 = \infty$ for all other v.

$$f_t(v) = \min_{w \in P_{t-1}} \{f_{t-1}(w) + c_{wv}\}, \quad t = 1, \ldots, 2m+1$$

The goal is to find $f_{2m+1}(v)$, where $v = (m+1, m+1, \ldots, m+1)$.

Let $\bar{m}$ be the maximum number of trips an aircraft can serve during the scheduling horizon (could be found easily using the network model for problem P1 in Section 1.2). There are at most $O((\bar{m})^n)$ vectors in any P_t, $t = 1, \ldots, 2m$. Hence the dynamic programming algorithm runs in time $O((\bar{m})^{n+1})$.

References

[1] R.K. Ahuja, T.L. Magnanti and J.B. Orlin, Network Flows: Theory, Algorithms and Applications, Prentice Hall, NJ, 1993.

[2] E.M. Arkin, E.B. Silverberg, "Scheduling jobs with fixed start and end times", *Discrete Applied Mathematics* 18, 1-8 (1987).

[3] L. Bianco, A. Mingozzi, S. Ricciardelli, "A Set Partitioning Approach to the Multiple Depot Vehicle Scheduling Problem", *Optimization Methods and Software*, Vol.3, 163-194 (1994).

[4] A.A. Bertossi, P. Carraresi, G. Gallo, "On Some Matching Problems Arising in Vehicle Scheduling Models", *Networks* 17, 271-281 (1987).

[5] L. Bodin, B. Golden, A. Assad, M. Ball, "Routing and Sceduling of Vehicles and Crews: The State of the Art", *Computers and Operations Research*, 10, 63-211 (1983).

[6] L. Bodin, D. Rosenfield, A. Kydes, "UCOST: A Micro Approach to a Transit Planning Problem", *Journal of Urban Analysis*, 5, 47-69 (1978).

[7] A. Bryant, "Gulfstream in Venture for Time Sharing of its Corporate Aircraft", *The New York Times*, November 16, p.4 (1985).

[8] G. Carpaneto, M. Dell'Amico, M. Fischetti, P. Toth, "A Branch and Bound Algorithm for the Multiple Depot Vehicle Scheduling Problem", *Networks*, 19, 531-548 (1989).

[9] A. Ceder, H.I. Stern, "Deficit Function Bus Scheduling with Deadheading Trip Insertions for Fleet Size Reduction", *Transportation Science*, 15, 338-363 (1981).

[10] M. Dell'Amico, M. Fischetti, P. Toth, "Heuristic Algorithms for the Multiple Depot Vehicle Scheduling Problem", *Management Science*, Vol.39, No.1, 115-125 (1993).

[11] C. Del Valle, "Can't afford a Lear? Buy a piece of one", *Business Week*, September 11, p.132 (1995).

[12] M. Garey, D. Johnson, Computers and Intractibility: A Guide to the Theory of NP-Completeness, W. H. Freeman and Co., New York, 1979.

[13] N.G. Hall, M.J. Magazine, "Maximizing the value of a space mission", *European Journal of Operational research* 78, 224-241 (1994).

[14] P. Keskinocak, S. Tayur, "Assigning off-days for multiple types of workers in seven-days-a-week operations", GSIA Working Paper, CMU, 1996.

[15] A.W.J. Kolen, L.G. Kroon, "On the computational complexity of (maximum) class scheduling", *European Journal of Operational Research* 54, 23-38 (1991).

[16] A.W.J. Kolen, L.G. Kroon, "On the computational complexity of (maximum) shift class scheduling", *European Journal of Operational Research* 64, 138-151 (1993).

[17] W.B. Powell, D.H. Gittoes, "A generalized labelling algorithm for the dynamic assignment problem", Dept. of Civil Eng. and Op. Research Report, Princeton University, 1994.

[18] C.C. Ribeiro, F. Soumis, "A Column Generation Approach to the Multiple-Depot Vehicle Scheduling Problem", *Operations Research*, Vol.42, No.1, 41-52 (1994).

[19] M.W.P. Savelsbergh, M. Sol, "The general pickup and delivery problem", *Transportation Science*, Vol. 29, No.1, 17-29 (1995).

[20] M.M. Solomon, J. Desrosiers, "Time Window Constrained Routing and Scheduling Problems", *Transportation Science*, Vol.22, No.1, 1-13 (1988).