Theory and Methodology

Fleet assignment and routing with schedule synchronization constraints

Irina Ioachim \textsuperscript{a,*}, Jacques Desrosiers \textsuperscript{b}, Françoise Soumis \textsuperscript{c}, Nicolas Bélanger \textsuperscript{c}

\textsuperscript{a} Delta Technology, 1001 International Boulevard, Department 709, Atlanta, GA 30354-1801, USA
\textsuperscript{b} GERAD and École des Hautes Études Commerciales, 5255 avenue Decelles, Montréal, Canada H3T 1V6
\textsuperscript{c} GERAD and École Polytechnique, 5255 avenue Decelles, Montréal, Canada H3T 1V6

Received 1 July 1997; accepted 1 July 1998

Abstract

This paper introduces a new type of constraints, related to schedule synchronization, in the problem formulation of aircraft fleet assignment and routing problems and it proposes an optimal solution approach. This approach is based on Dantzig–Wolfe decomposition/column generation. The resulting master problem consists of flight covering constraints, as in usual applications, and of schedule synchronization constraints. The corresponding subproblem is a shortest path problem with time windows and linear costs on the time variables and it is solved by an optimal dynamic programming algorithm. This column generation procedure is embedded into a branch and bound scheme to obtain integer solutions. A dedicated branching scheme was devised in this paper where the branching decisions are imposed on the time variables. Computational experiments were conducted using weekly fleet routing and scheduling problem data coming from an European airline. The test problems are solved to optimality. A detailed result analysis highlights the advantages of this approach: an extremely short subproblem solution time and, after several improvements, a very efficient master problem solution time.

Keywords: Dantzig–Wolfe decomposition; Time windows; Weekly aircraft fleet assignment; Routing; Scheduling; Air transportation; Dynamic programming; Branch and bound

1. Introduction

This paper solves an aircraft fleet routing and scheduling problem that occurs in the long range planning process of many airlines. The problem involves covering, at minimum cost, a given set of flights which present schedule flexibility on the departure time. For each flight the data include an identifier, origin and destination stations, a duration, a minimum associated ground time at the destination and a time window on either the departure or its arrival. Waiting is permitted before and inside the time windows at no cost. Moreover, a flight is characterized not only by its identifier, but also by the day of the week when it is to be flown. When flights with the same identifier are
flown on different weekdays, the departure has to be scheduled at the same time every day, for marketing purpose. These same departure time constraints, which ensure the required schedule synchronization, are the main new problem feature with respect to previous aircraft scheduling problems [6,13,18]. Other requirements, such as maintenance or restrictions on the aircraft types found together at a given base are not usually addressed in long range planning and are not included in the proposed problem formulation.

Before the eighties, most of the research on fleet scheduling problems, both in airline and in urban transit contexts, focused on the assignment of fixed time-tabled tasks such as flights or routes to homogeneous fleets. An extensive survey of this research in the airline context is presented in Ref. [13]. The network flow algorithms were instrumental for the solution of many such problems [15]. However, one of the first papers to consider variable flight departure times is [22]. The author proposed an optimal approach based on the discretization of the time windows and a network problem formulation with side constraints. To reduce the size of the resulting network, which is huge for wide flight departure time windows, the author proposes a node aggregation technique which was later used in other applications (for an example in school busing, see Ref. [28]).

For the daily aircraft routing problem with fixed time-tabled flights and multiple aircraft types, a multi-commodity formulation is presented in Ref. [1]. This network based problem formulation is solved by the simplex method and the fractional variables were then rounded to obtain an integer solution. An aggregation of the flow variables defined in Ref. [1] was recently studied in Ref. [18] as part of another multi-commodity formulation. The authors solved the resulting linear program with an interior point method and used a heuristic branch and bound approach to obtain an integer solution. Introducing time window flexibility for the daily fleet assignment problem, [6] considered a nonlinear multi-commodity flow formulation and an equivalent set partitioning formulation obtained from the former one by using an extension of the Dantzig–Wolfe decomposition method [4]. This solution approach is embedded into an optimal branch and bound tree where branching decisions are imposed on the flow variables. This decomposition approach has been successfully applied to various fleet assignment, crew scheduling and crew rostering problems, in urban transit as well as in airline and rail contexts. Recent surveys of the research on routing and scheduling problems with time constraints are presented in Refs. [7,11,26]. Lagrangian relaxation approaches have also been devised for some of these problems and can be found in Refs. [5,10,21].

The aircraft fleet routing and scheduling problem considered in this paper covers a one week period for long-haul operations. It involves times windows and also requires schedule synchronization constraints over the different days of the week. In this problem, all flights to be performed during a week are scheduled simultaneously rather than producing schedules for a shorter period (daily schedules, for example) and then repeating them. The resulting problem is not only larger, but must also include these same departure time constraints to provide a synchronized schedule. The solution of this more difficult problem is important since it produces more reliable aircraft schedules that generate greater savings and need less modifications when put in operation. This weekly fleet assignment problem was recently tackled in Ref. [27]. The author proposed a heuristic method and an optimal approach. This latter approach consists in solving a huge linear formulation of the problem as a mixed integer program and will be further discussed in Section 2.2

Although presented in this aircraft fleet assignment context, the schedule synchronization constraints and the proposed solution approach are useful for other applications. Such are: job-shop scheduling [17]; vehicle routing problems with split deliveries [7,12]; periodic aircraft routing and scheduling with time windows [6]; and in scheduling situations with sliding time windows [14].

The contribution of this paper focuses on the schedule synchronization feature. First, we propose a multi-commodity flow formulation for the fleet assignment problem that includes this new type of constraints. Second, we propose an optimal branch and bound method where the bounds
are computed by using an extension of the Dantzig–Wolfe decomposition method. The subproblem resulting from this decomposition is a shortest path problem with time windows and time costs, as recently discussed in Ref. [20]. The master problem results in a set partitioning type problem with additional schedule synchronization constraints. Third, we show that we cannot impose branching decisions by fixing master problem variables to zero or to one as in many usual vehicle routing and crew scheduling problems. Fourth, in addition to the branch and bound strategy based on time synchronization variables, we develop a dedicated branch and bound commodity formulation. Finally, the computational experiments identify the main advantages of the method, which is shown to be very efficient and practical. The paper is organized as follows. Section 2 presents a multi-commodity network based mathematical formulation of the problem. Next, Section 3 proposes an optimal solution approach based on a mathematical decomposition of the problem. Section 4 extends the solution approach to the case of networks containing cycles. Computational experiments are presented in Section 5. Finally, Section 6 gives some conclusions and directions for future work.

2. Problem formulation

This section presents a multi-commodity formulation which deals with the general case of a heterogeneous aircraft fleet and includes non-linear constraints. We next discuss the special case where all aircraft are identical; such a formulation was previously used in Ref. [27].

2.1. A multi-commodity flow formulation

We begin by defining the notation associated with groups of flights that must respect the same departure time. Let $M$, indexed by $m$, denote the set of group identifiers. With each identifier $m \in M$ are associated one or several flights (with the same origin and destination) to be performed on different days. Let $N$, indexed by $i$, denote the set of all flights to be performed during a week. Since a given group identifier may be associated with more than one flight, we have that $|M| \leq |N|$. Let $m_i$, with $m_i \in M$, denote the group identifier of flight $i$. It follows that all pairs of flights $i, j \in N$ with $m_i = m_j$ must have the same departure time. We assume that the time windows of two such flights $i$ and $j$ satisfy $a_i = (a_j \mod 1440)$ and $b_i = (b_j \mod 1440)$, where 1440 minutes correspond to 24 hours. For any pair of flights to synchronize that does not satisfy this condition, we can replace both time windows with their intersection (modulo 1440) without any loss of generality. However, having identical time windows for both flights is a necessary but not a sufficient condition to obtain a synchronized schedule; we have added specific constraints in our mathematical formulation of the problem to ensure synchronization. Note that time related problem data are usually expressed in minutes in most airline, urban transit or rail applications.

The proposed problem formulation is based on an acyclic network structure. However, in Section 4 we extend our solution method for networks containing cycles.

Let $K$ denote the set of aircraft and define a network $G^k = (V^k, A^k)$ for each of them, where $V^k$ is the set of nodes and $A^k$ is the set of arcs. Let $o(k)$ and $d(k)$ denote the source and sink nodes in $G^k$, while $N^k$ are nodes representing flights compatible with aircraft $k$. Hence, $V^k = N^k \cup \{o(k), d(k)\}$ represents the node set. Therefore, the set of all weekly flights is given by $N = \bigcup_{k \in K} N^k$. Let $[a_i, b_i]$ denote the time window for the departure of flight $i \in N$, while $d_i$ and $s_i$ denote its duration and associated minimum ground time, respectively. Without loss of generality, let $a_{o(k)} = b_{o(k)} = 0$. The arc set $A^k$ of network $G^k$ contains two types of arcs, the depot arcs and the flight connection arcs. The depot arcs $(o(k), i)$, $(i, d(k))$, for $i \in N^k$, link each flight with the source and sink nodes. A flight connection arc corresponds to a feasible connection between a flight arriving at an airport and a flight departing from the same airport. The connection arc $(i, j) \in A^k$ satisfies $a_i + d_i + s_i \leq b_j$. Concerning the time windows for the sink nodes, let $a_{d(k)} = 0$ and $b_{d(k)} = \max_{i \in N^k} b_i + \max_{i \in N^k} (d_i + s_i)$. Finally, a cost $c_{ij}^k$ is associated with each
arc \((i, j) \in A^i\). The network cost structure depends on the objective of the optimization problem. The capital cost and fixed cost for aircraft \(k\) are assigned to arc \((o(k), i)\), for \(i \in N^k\). The operational cost of performing flight \(i\) by aircraft type \(k\) is assigned to all arcs with tails at node \(i\), for all \(i \in N^k\). An arc \((o(k), d(k))\) is added to set \(A^k\) to account for unused aircraft, \(k \in K\).

A multi-commodity flow problem formulation where each aircraft \(k \in K\) defines a commodity is presented next. To formulate the problem, we define the following variables:

- \(X_{ij}^k, k \in K, (i, j) \in A^k\): binary flow variable equal to 1 if arc \((i, j)\) is part of the solution route for aircraft \(k\) and 0 otherwise;
- \(T_i^k, k \in K, i \in N^k\): time variable representing the departure time of flight \(i\) if it is performed by aircraft \(k\) and 0 otherwise;
- \(T_m, m_i \in M\): non-negative time variable representing the slack between the beginning of a flight time window and the actual flight departure time, that is, \(T_m = \sum_{k \in K} T_i^k - a_i\).

With this notation, the problem formulation becomes

\[
\min \sum_{k \in K} \sum_{(i, j) \in A^k} c_{ij}^k X_{ij}^k, \tag{1}
\]

s.t.

\[
\sum_{k \in K} \sum_{j \in N^k} X_{ij}^k = 1, \quad \forall i \in N, \tag{2}
\]

\[
\sum_{k \in K} T_i^k - T_m = a_i, \quad \forall i \in N' \subseteq N, \tag{3}
\]

\[
\sum_{j \in N^k} X_{o(k), j}^k = \sum_{j \in N^k} X_{j, d(k)}^k = 1, \quad \forall k \in K, \tag{4}
\]

\[
\sum_{j \in N^k} X_{ij}^k - \sum_{j \in N^k} X_{ji}^k = 0, \quad \forall k \in K, \forall i \in N^k, \tag{5}
\]

\[
X_{ij}^k (T_i^k + d_i + s_i - T_j^k) \leq 0, \quad \forall k \in K, \forall (i, j) \in A^k, \tag{6}
\]

\[
a_i \leq T_i^k \leq b_i, \quad \forall k \in K, \forall i \in \{o(k), d(k)\}, \tag{7}
\]

\[
a_i \left( \sum_{j \in A^k} X_{ij}^k \right) \leq T_i^k \leq b_i \left( \sum_{j \in A^k} X_{ij}^k \right), \quad \forall k \in K, \forall i \in N^k, \tag{8}
\]

\[
X_{ij}^k \text{ binary}, \quad \forall k \in K, \forall (i, j) \in A^k. \tag{9}
\]

The objective (1) together with constraints (2)–(9), except for (3), form a classical multiple traveling salesman problem with time windows [11]. The time constraints of the problem are the same departure time constraints (3) and time window constraints, (7) and (8). Note that same departure time constraints (3) are only required for flight identifiers \(m \in M\) involving at least two flights during the week. Hence, there are at most \(|N|\) such constraints. Let \(N'\) denote the set of same departure time constraints. Time window constraints (7) are imposed at the source and sink nodes. Constraint set (8) imposes that \(T_i^k = 0\) whenever \(\sum_{j \in N^k} X_{ij}^k = 0\), that is, the time window is imposed only for the aircraft \(k\) that performs flight \(i\). Hence, whenever \(\sum_{j \in N^k} X_{ij}^k = 1\), variable \(T_i^k\) contributes to (3). Finally, constraints (6) model the compatibility requirements between the flow and time variables. These are the non-linear form of the Miller–Tucker–Zemlin subtour elimination constraints [24].

Note that time variables \(T_m, i \in N\), can be eliminated from the problem formulation and written as functions of variables \(T_i^k\). However, since it can be shown that for problems with integer time data, \(a_i, b_i, d_i\) and \(s_i\) for all \(i \in N\), there exist optimal solutions where the time variables are also integer, variables \(T_m\) are later used to define a branching strategy.

### 2.2. Special case: Homogeneous fleet

Defining a commodity for each aircraft yields a general formulation which implicitly accommodates for several aircraft types. A special case of the previous formulation occurs when all aircraft are identical. In this case, all aircraft can use the same network and index \(k\) is no longer useful in the variable definition. Remark that the summation on the flow variables can be eliminated from the time window constraints (8) if all aircraft are identical.

This special case is presented in Ref. [27]. The author uses a linearized version of constraints (6) and the resulting formulation involves \(|A|\) binary flow variables, \(|N|\) bounded time variables, \(|N|\) assignment constraints, \(|N'|\) synchronization con-
constraints and $|A|$ constraints in set (6). Moreover, the right hand side of constraints (4) is the number of available aircraft.

The author proposed a solution method which consists in solving this mixed integer linear program by using a linear programming optimization package and a branch and bound scheme. The main disadvantage of such a formulation is that the resulting linear program has a large number of constraints (of the order of $|A|$), and the practical application of this method is restricted to problems involving a small number of feasible flight connections. The author reported solving three weekly problems with up to 1213 constraints. Even for a small number of flight connection arcs, the case of a heterogeneous aircraft fleet cannot be efficiently solved by such a direct method. The solution process described in Section 3 permits to avoid this inconvenient by using a mathematical decomposition of the above problem formulation.

3. Solution method

An optimal integer solution for the above problem is provided by embedding a column generation approach within a branch and bound procedure [11]. First, the linear relaxation of formulation (1)–(9) is solved by Dantzig–Wolfe decomposition. Second, to find an optimal integer solution, this first step is incorporated into a branch and bound scheme, where each such linear relaxation solution gives a lower bound for the explored branch.

3.1. Mathematical decomposition

A solution, not necessarily integer, for the above problem can be found using Dantzig–Wolfe decomposition [4]. We propose a decomposition scheme in which the objective function (1) and constraints (2) and (3) belong to the master problem, while constraints (4)–(9) belong to the subproblems. Note that each $k$ is associated with a specific subproblem. The variables (columns) of the master problem correspond to feasible aircraft assignments while the role of the subproblems is to generate such feasible aircraft assignments (columns) until optimality is reached. Therefore, each subproblem is in fact a shortest path problem with side constraints and its solution is the minimal reduced cost path corresponding to a given set of dual variables for the master problem. Moreover, since time windows restrict the times at which nodes can be visited in the subproblem, the solution path describes not only the list of selected arcs, but the schedule as well. Thus, an optimal solution path in subproblem $k$, for an acyclic network structure, is denoted by

$$\{x_{ij,k}, t_{ip,k}\}, (i,j) \in A^k, i \in N^k,$$

where $p$ stands for the corresponding column index in the master problem. We now introduce some notation required for reformulating constraints (2) and (3) as they appear in the master problem. Let $\theta_{p,k}, p \in \Omega^k$ of cost $c_{p,k}$ denote the variables of the master problem generated by subproblem $k$. The variables in formulation (1)–(9), which are also used in the subproblems, are related to the variables of the master problem as follows:

$$X_{ij}^k = \sum_{p \in \Omega^k} x_{ij,p}^k \theta_{p,k}, \forall k \in K, \forall (i,j) \in A^k,$$

$$X_{ij}^k \text{ binary}, \forall k \in K, \forall (i,j) \in A^k,$$

$$T_{ip}^k = \sum_{p \in \Omega^k} t_{ip,p}^k \theta_{p,k}, \forall k \in K, \forall i \in N^k,$$

$$\sum_{p \in \Omega^k} \theta_{p,k} = 1, \forall k \in K,$$

$$\theta_{p,k} \geq 0, \forall k \in K, \forall p \in \Omega^k.$$

Let $x_{ij,p}^k$ and $t_{ip,p}^k, i \in N, p \in \Omega, k \in K$, be the coefficients corresponding to the flight covering constraints (2) and the same departure time constraints (3), respectively. Recall that each negative cost subproblem solution defines a new column in the master problem. The corresponding cost and master problem coefficients are obtained from the following calculations:
\[
\begin{align*}
  c^k_p &= \sum_{(i,j) \in A^k} c^k_{ij} x^k_{ij,p}, \quad \forall k \in K, \\
  z^k_i &= \sum_{j(i,j) \in A^k} x^k_{ij,p}, \quad \forall k \in K, \quad \forall i \in N^k, \\
  \beta^k_{i,p} &= t^k_{i,p}, \quad \forall k \in K, \quad \forall i \in N^k.
\end{align*}
\]

The master problem is next written as an integer linear program corresponding to a set partitioning constraints.

\[
\begin{align*}
  \text{min} & \sum_{k \in K} \sum_{p \in \Omega^k} c^k_p \theta^k_p \\
  \text{s.t.} & \sum_{k \in K} \sum_{p \in \Omega^k} z^k_i \theta^k_p = 1, \quad \forall i \in N, \\
  & \sum_{k \in K} \sum_{p \in \Omega^k} \beta^k_{i,p} \theta^k_p - T_{m} = a_i, \quad \forall i \in N', \subseteq N, \\
  & \sum_{p \in \Omega^k} \theta^k_p = 1, \quad \forall k \in K, \\
  & \theta^k_p \geq 0, \quad \forall k \in K, \quad \forall p \in \Omega^k, \\
  & X^k_{ij} = \sum_{p \in \Omega^k} x^k_{ij,p} \theta^k_p, \quad \forall k \in K, \quad \forall (i,j) \in A^k,
\end{align*}
\]

The time window constraints (8) impose a zero value for time variables associated with nodes not in the shortest path. Thus, all node costs present in the objective (26) are associated with nodes in the shortest path. The subproblem corresponding to a given \( k \), modeled by relations (26) and (4)–(9), is a shortest path problem with time windows and linear time costs.

An efficient dynamic programming algorithm for optimally solving a subproblem is presented in Ref. [20]. As in standard labelling shortest path algorithms for acyclic networks, the nodes are processed in topological order and paths starting with modified arc and node costs. Thus, the reduced cost of an arc is obtained by subtracting the dual variable \( \pi_j \) from \( c^k_{ij} \). Since the dual variables \( \pi_j \) are associated with time constraints, they introduce a linear reduced cost \(-\sigma_i\) associated with the time variables \( T^k_i \), for \( i \in N^k \). Subproblem \( k \) has the same variables as in the previous problem formulation, i.e., \( X^k_{ij}, (i,j) \in A^k \) equal to 1 for arcs in the shortest path and 0 otherwise, and time variables \( T^k_i \) representing the departure of flight \( i \in N^k \). The objective of the subproblem \( k \) is written as

\[
\begin{align*}
  \text{min} & \sum_{(i,j) \in A^k} (c^k_{ij} - \pi_j) x^k_{ij} - \sum_{i \in Y^k} \sigma_i T^k_i - \pi^k_0.
\end{align*}
\]

The objective function (19), covering constraints (20) and constraints (22) describes a minimum cost fleet assignment problem. Constraints (21) provide synchronization between different aircraft performing flights subject to same departure time constraints. Note that there are exactly \(|N|\) task covering constraints in set (20) and at most \(|N|\) same time constraints in set (21). Note that bound constraints can be derived for time variables \( T_m \), as from the definition of these variables we have \( 0 \leq T_m \leq b_i - a_i \), such that \( m = m_i, m \in M \). But since these bound constraints are redundant with respect to the subproblem constraint set, they are not present in the formulation. Constraints (22) are imposed to require a convex combination of the non-negative path variables (23). The only integer variables in formulation (19)–(25) are the \( X^k_{ij} \) variables. Therefore, the linear relaxation of this problem formulation is obtained by eliminating constraints (24) and (25).
at the source node are iteratively extended until all nodes are treated, i.e., the sink node is reached. The information attached to each path is a function giving the cost of the path according to the node visiting time. This function generalizes the (cost, time) type of labels used in dynamic programming algorithms for the shortest path problem with time windows (SPPTW) [8,9]. This cost function is piecewise linear, nonincreasing, continuous, with a number of linear pieces less than or equal to the number of nodes visited by the path. The minimum of these functions at each node indicates the dominance cost function, which gives the cost of the best paths within the node time window. This dominance cost function is also piecewise linear, non-increasing and may include steeps. It generalizes the steep function obtained as a result of the dominance process used in SPPTW dynamic programming approaches [8,9]. The resulting algorithm is pseudo-polynomial. Moreover, the optimal schedule associated with the shortest path is proved to be integer if all problem time related data are integer. Computational experiments prove that this pseudo-polynomial dynamic programming algorithm is more efficient than an alternative optimal approach that selectively discretizes the time windows.

In practice, the linear relaxation of the master problem is solved first. Then, this process is embedded into a branch and bound procedure to find an optimal integer solution. This procedure is described in Section 3.2.

3.2. Branch and bound procedure

Several branching strategies compatible with the column generation technique are discussed in the specialized literature. Two recent papers [19] and [23] propose branching strategies involving decisions on path variables when the subproblem is allowed to generate up to \( |N| \) best solutions. In Ref. [25], the authors devised a branching strategy on path variables which is compatible with set partitioning problems. Generalizations of this strategy can be found in Refs. [2,29], where branching decisions on certain sums of path variables are introduced. These decisions are then transferred to the subproblem using additional binary variables. To our knowledge, the only previous paper discussing branching techniques which involve time variables and which are compatible with the column generation scheme is [16].

Constraints (24) and (25), which were eliminated to obtain the linear relaxation of our problem, must be imposed in order to obtain integer flow variables \( X_{ij}^k \). To obtain an optimal integer solution for our problem, we propose a branching technique involving decisions both on the time variables \( T_m, m \in M \), and the arc variables \( X_{ij}^k, k \in K \) and \((i,j) \in A^k \). The branching decisions involving the time variables apply only when the time related problem data are integer. We use a binary branching tree, and the root node is the initial problem given by formulation (19)–(23). New nodes are created by adding branching decisions to both the master problem and the subproblem. The main steps of the procedure are: choose a fractional decision variable; define two possible branching decisions that eliminate the current fractional value; identify the most promising of the two decisions; impose the most promising decision first, both in the master problem and in the subproblem; and solve the current branch node. These steps are described next.

Choose a fractional decision variable: When a fractional solution is obtained by column generation after solving a given branching node, we first identify fractional time variables \( T_m \). Recall that each such variable appears in at least two master problem constraints and is bounded by \( b \leq T_m \leq b \). The first decision involving \( T_m \) has \( b = 0 \) and \( \bar{b} = b_i - a_i \), for \( m = m_i \). The variable chosen for the next branching decision is that appearing in the largest number of master problem constraints. Let \( T \) denote the fractional value of the selected variable.

Define two possible branching decisions: Using the previous bounds two branches are created, corresponding to new bounding decisions \( b \leq T_m \leq T \) and \( T \leq T_m \leq \bar{b} \). Such branching decisions are then imposed both in the master problem and in the subproblem to eliminate the current fractional value of \( T_m \).

Identify the most promising decision: To evaluate each of the two possible decisions, we compute
the sum of all master problem variables corresponding to paths that contain flights (nodes) to synchronize using $T_m$. This computation takes place for each of the following time intervals: $b \leq T_m \leq \lfloor T \rfloor$ and $\lfloor T \rfloor \leq T_m \leq b$. The most promising branching decision is that corresponding to the larger sum. Let $d \leq T_m \leq d$ denote the decision to be imposed first.

**Impose the selected decision:** Decision $d \leq T_m \leq d$ is imposed on the master problem by eliminating all columns with $\beta_{i,p} \not\in [d,d]$. To impose the current branching decision in the subproblem, the node time windows are modified to $[a_i + \max(b, d), a_i + \min(d, b)]$ for $k \in K$ and $\forall i \in N^k$ such that $m_i = m$. Thus, all time windows for flights (nodes) appearing together with $T_m$ in a same time constraint are modified, and the next shortest path subproblem solution will be a path respecting the current branching decision.

**Solve the current branching node:** The resulting master problem is reoptimized by column generation. Note that all columns generated at this node are consistent with the last decision imposed.

**Example:** This example is used to demonstrate how our strategy which identifies the most promising decision is applied. Suppose we have a fractional value for $T_m = 42.5$ and the time window within which all corresponding synchronized flights must depart is 6 AM to 8 AM. Furthermore, suppose that only two flights $i$ and $j (i \neq j)$ are synchronized using $T_m$ and that this is the first decision involving this variable, so we have $T_m \in [0, 120]$. The two branching decisions which are possible in this case are $T_m \in [0, 42]$ and $T_m \in [43, 120]$. The fractional value of the time variable $T_m$ was obtained as a weighted combination of the times corresponding to different paths which contain these flights. For node $i$, we consider a path that visits this node at 6:25 AM with a flow of 0.8, a path that visits at 7:50 AM with a flow of 0.1 and a path that visits the node at 7:55 AM with a flow of 0.1. For node $j$, we consider a path that visits this node at 6:15 AM with a flow of 0.5 and a path that visits at 7:10 AM with a flow of 0.5. The sum of the master problem variables for the first decision is $0.13 = 0.8 + 0.5$, since $15, 25 \in [0, 42]$. The sum of the variables for the second decision is $0.7 = 0.1 + 0.1 + 0.5$. Our strategy identifies $[0, 42]$ as the most promising branching decision and this branch is explored first.

New branching nodes are created as the process iterates, until all variables $T_m, m \in M$, are integer. The branching tree was explored using a best first strategy. Each branching node has a lower bound provided by the linear relaxation solution of its predecessor, and the best branching node is the one with the smallest lower bound.

Two additional features were added to the above branching strategy on time variables, to reduce the size of the master problem whenever possible and to re-use potentially useful columns. First, some branching decisions can reduce the number of constraints in the corresponding branch. For example, for a current branching decision on variable $T_m$ with $d = d$, all same time constraints corresponding to flights $i \in N^k$ such that $m_i = m$ can be deleted for $k \in K$, since all these flights are trivially scheduled at time $d$. Note that each such a decision eliminates at least two constraints from the master problem. Rows associated with these constraints must be added back to the master problem when backtracking in the branching tree, however, since decision $d = d$ is no longer valid. Backtracking occurs if the current branching decision leads to an infeasible problem or to an integer solution that has not (yet) been proved optimal. Second, columns eliminated at a given node may be useful when backtracking in the branching tree. Recall that when imposing a current decision on the master problem, several columns may be eliminated from the master problem because of their coefficients in constraints involving the selected decision variable $T_m$. All of these columns are retained at the node corresponding to this branching decision. Similarly, when backtracking in the branching tree and when the previous decision is no longer valid, the eliminated columns are added back into the master problem so that they are not generated again. Obviously, subproblem decisions are also modified when backtracking in the branching tree.

The solution of (19)–(23) may still be fractional, even if all time variables $T_m$ are integer, because of fractional path variables. The previous type of branching decisions can be slightly modified to
continue the search for the optimal integer solution. Let \( \theta_p \) denote a fractional path variable, \( x_p \), the visiting time of a node \( i \) in this path and \( T \) be the integer value of the time variable \( T_m \) involved with the same time constraint associated with node \( i \). From (20), it follows that there is at least one other fractional path variable, \( \theta_p \) corresponding to a path visiting node \( i \) at time \( t_p \). The branching decisions made in this case on the time variable \( T_m \) restrict the possible values of this variable either to \( b \leq T_m \leq T - 1 \) or to \( T \leq T_m \leq b \). For example, if there are only two fractional paths visiting node \( i \), each of these decisions eliminates one of the fractional paths, since the value \( T \) was obtained as an average of the corresponding node visiting times \( t_p \) and \( t_p \) (provided that \( t_p \neq t_p \)). This discussion applies, obviously, to more than two fractional variables corresponding to paths that visit a given node.

Note that this type of branching decisions can not be applied to eliminate a fractional solution if, for the above example, \( t_p = t_p \). In such a case, the search for the optimal integer solution is continued with decisions made on the arc variables involved with the path corresponding to the master problem fractional variable. This second type of decision was used in many practical applications [11], such as airline scheduling.

4. Solution method for networks with cycles

The subproblem solution is used to compute the coefficients of the corresponding master problem column. In Section 2 we introduced the assumption that networks \( G_k \), \( k \in K \), are acyclic. For this case, the subproblem solution path is also acyclic and therefore the corresponding master problem coefficients are given by the unique departure time of flight \( i \) as \( \beta_{i,p} = t_{i,p} \), as shown in Section 3. However, this assumption is not critical and the proposed solution approach also holds for networks with cycles. The solution approach corresponding to networks with cycles is discussed next.

Assume, for example, that the subproblem solution represented by master variable \( \theta_q \) is a path which visits node \( i \) twice. It follows that the corresponding coefficient in the task covering constraints (20) is given by \( x_i^k = \sum_{j(i,j) \in q} x_{i,j}^k = 2 \); to obtain a master column of the same reduced cost as the subproblem solution, the corresponding coefficient in the same departure time constraints (21) is computed as \( \beta_{i,q}^k = t_{i,q}^k + t_{i,q}^k \), where \( t_{i,q}^1 \) and \( t_{i,q}^2 \) are the two visiting times at node \( i \). Since the right hand side value is 1 for the covering constraints, the corresponding path variable satisfies \( \theta_q^i \leq 1/2 \). Note that only columns with binary coefficients \( x_{i,p}^k \) may appear in an integer solution, as the covering constraints (20) require that \( \sum_{k(p) \in q} \theta_q^k = 1 \), for all \( i \in N \). In a non-integer optimal solution to the master problem, let \( \delta \) denote the fractional value of variable \( \theta_q \) corresponding to the above considered path. Such a path variable may only be part of a non-integer solution as \( \sum_{k(p) \in q} \theta_q^k = 1 - \delta \) for node \( i \), which is visited twice. It follows that in an integer solution we have \( \delta = 0 \). Note that the time coefficients \( \beta_q \), \( k \in K \), \( i \in N \), are integer if the problem time data are integer. This discussion can be easily extended to any subproblem solution which contains cycles. In many airline scheduling applications, however, the underlying networks are often acyclic, especially for relatively narrow time windows.

5. Computational experiments

We tested the proposed approach on a data set first used in Ref. [27], arising from actual weekly fleet scheduling situations encountered at Lufthansa Airlines. These data correspond to the long range planning problem described in Section 1 and is used for fleet size forecasts. Our objective, therefore, is to find the minimum fleet size aircraft assignment. A single aircraft type is used in each test problem and no base restrictions are specified for the aircraft. Therefore, all aircraft may use the same network \( G = (V,A) \) with \( V = N \cup \{o,d\} \). The simplest cost structure to minimize the size of the fleet that covers all flights \( i \in N \), is \( c_{oi} = 1 \) and \( c_{ij} = 0 \) for \( i \in N \), \( j \in V \). The main goal of these experiments is to highlight the computational advantages of the proposed solution approach. We used twelve test problems, which are described in Table 1. The first column contains the number of
flights involved with each test problem. In long range forecasting problems, the time windows can be very wide since there is no need for tight time window specifications. Thus, in our data set the time window width varies from 2 or 3 hours to 24 hours. However, there are also a few fixed-schedule flights, i.e., such that \(a_i = b_i\). The average time window width is indicated in the second column of Table 1. The third column indicates the average flight duration plus the minimal ground time that must follow a given flight. Finally, the last column contains the number of flights that must satisfy a same time constraint. The difference between the total number of flights and the number of same time flights (same departure time constraints) is due either to fixed schedule flights or to flights flown only once a week.

5.1. Solution report – First results

The master problem is solved using the primal simplex of the CPLEX software [3], while the subproblem is solved using the dynamic programming algorithm for the shortest path problem with linear time costs presented in Ref. [20]. The first attempt to solve the master problem resulted in major numerical stability difficulties with the simplex software used for master problem optimization. Therefore, we reformulated constraints (3) of the problem as

\[
\sum_{k \in K} \left( T_i^k - a_i \sum_{j : (i,j) \in A} x_{ij}^k \right) - T_{m_i} = 0, \forall i \in N'
\]

and the coefficients of the master problem corresponding constraints are given by

\[
\beta_{i,p}^k = t_{i,p}^k - a_i \left( \sum_{j : (i,j) \in A} x_{ij}^k \right),
\]

\(\forall k \in K, \forall i \in N^k\).

This new formulation is based on the same principle as constraints (8) and transforms the departure time of a flight \(i\) into the slack obtained with respect to the beginning of the time window. The master problem coefficients \(\beta_{i,p}^k\) are reduced to values less or equal to 1440 minutes (the number of minutes in a day) instead of values of up to 10080 minutes (the number of minutes in a week). These new values for the master problem coefficients decreased by over 30\%, on the average, the master problem solution time and increased significantly the numerical stability of the simplex solution.

The tests were run on a HP9000/735 (124.0 Mips, 40.0 Mflops). Table 2 reports on the proposed solution process. We present the size of the resulting networks, that is, the number of arcs and the number of nodes, in columns one and two, respectively. Due to long flight durations and to relatively few feasible flight connections, all prob-

| Problem name | Number of flights | Avg. TW width (min) | Avg. flight duration (min) | Number of same time flights \((|N'|)\) |
|-------------|-------------------|----------------------|---------------------------|---------------------------------|
| Test 0      | 106               | 472.97               | 654.58                    | 106                             |
| Test 1      | 106               | 514.29               | 654.58                    | 106                             |
| Test 2      | 106               | 509.01               | 653.80                    | 106                             |
| Test 3      | 85                | 158.88               | 134.04                    | 78                              |
| Test 4      | 54                | 206.11               | 110.55                    | 50                              |
| Test 5      | 14                | 766.07               | 158.71                    | 12                              |
| Test 6      | 10                | 1072.50              | 213.44                    | 8                               |
| Test 7      | 18                | 1051.39              | 685.36                    | 18                              |
| Test 8      | 24                | 778.75               | 723.45                    | 24                              |
| Test 9      | 76                | 440.39               | 635.80                    | 76                              |
| Test 10     | 50                | 602.70               | 621.66                    | 50                              |
| Test 11     | 36                | 716.53               | 662.23                    | 36                              |
lem networks are acyclic despite of the very wide time windows. The third column indicates the number of same departure time (SDT) constraints. This number is no larger than the number of same departure time flights presented in Table 1. However, the same time constraints are satisfied a priori for any fixed schedule flights. For example, consider the problem Test 3, in which there are 78 same departure time flights, but only 18 of which have a flexible schedule (non-zero time window width). Mathematically speaking, the same solution is obtained whether the 60 redundant constraints are present in the master problem or not.

Computationally, however, it is very important to eliminate these redundant constraints to increase the numerical stability of the simplex software used for master problem optimization. The fourth column contains the solution to the linear relaxation of the master problem, which is in fact the root node in the branching tree. The fifth column presents the optimal integer solution found in the branching tree. The last two columns present the number of branch and bound nodes and the total CPU time required for an optimal integer solution. Additional information on backtracking in the branch and bound tree is depicted in Fig. 1 for

### Table 2

<table>
<thead>
<tr>
<th>Problem name</th>
<th>Nodes number</th>
<th>Arcs number</th>
<th>Number of SDT constraints</th>
<th>LR sol.</th>
<th>BB sol.</th>
<th>Number of BB nodes</th>
<th>CPU (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 0</td>
<td>108</td>
<td>1255</td>
<td>106</td>
<td>11</td>
<td>11</td>
<td>103</td>
<td>2646.3</td>
</tr>
<tr>
<td>Test 1</td>
<td>108</td>
<td>1255</td>
<td>106</td>
<td>10</td>
<td>10</td>
<td>83</td>
<td>4186.4</td>
</tr>
<tr>
<td>Test 2</td>
<td>108</td>
<td>1237</td>
<td>102</td>
<td>10</td>
<td>10</td>
<td>90</td>
<td>2610.7</td>
</tr>
<tr>
<td>Test 3</td>
<td>87</td>
<td>1279</td>
<td>18</td>
<td>7</td>
<td>7</td>
<td>10</td>
<td>71.8</td>
</tr>
<tr>
<td>Test 4</td>
<td>56</td>
<td>617</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>9</td>
<td>9.4</td>
</tr>
<tr>
<td>Test 5</td>
<td>16</td>
<td>63</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>Test 6</td>
<td>12</td>
<td>36</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>0.1</td>
</tr>
<tr>
<td>Test 7</td>
<td>20</td>
<td>50</td>
<td>18</td>
<td>15</td>
<td>15</td>
<td>7</td>
<td>0.6</td>
</tr>
<tr>
<td>Test 8</td>
<td>26</td>
<td>106</td>
<td>24</td>
<td>5.8</td>
<td>6</td>
<td>187</td>
<td>17.6</td>
</tr>
<tr>
<td>Test 9</td>
<td>78</td>
<td>744</td>
<td>76</td>
<td>9</td>
<td>9</td>
<td>45</td>
<td>428.3</td>
</tr>
<tr>
<td>Test 10</td>
<td>52</td>
<td>407</td>
<td>50</td>
<td>7</td>
<td>7</td>
<td>1</td>
<td>61.2</td>
</tr>
<tr>
<td>Test 11</td>
<td>38</td>
<td>202</td>
<td>36</td>
<td>6</td>
<td>6</td>
<td>19</td>
<td>5.3</td>
</tr>
</tbody>
</table>

Fig. 1. Depth of the branch and bound nodes.
three problems, Test 0, Test 1 and Test 2. The maximal depth reached during the branching process is of 102, 71 and 89, respectively. Using the best first strategy, no backtracking is performed during the solution process for problems Test 0 and Test 2, while this occurs for problem Test 1 where the total number of branch and bound nodes reported in Table 2 is slightly bigger than the number of levels explored in the branching tree. More detail on the computational times as well as other elements regarding the solution process are given in Table 3. The problem names and the number of SDT constraints were repeated in this Table to underline the fact that the difficulty of the problem increases when the number of these constraints increases. The other columns contain results concerning the solution process, that is, branch and bound and column generation. The third column indicates the number of column generation iterations required to reach the integer solution. In other words, it represents the number of times that the subproblem was solved during this process. The fourth column represents the CPU time consumed in the master problem optimization by CPLEX, while the fifth column indicates the cumulative CPU time consumed for the subproblem solution. The total number of columns generated by the subproblem is indicated in the sixth column. Columns three and four permit to estimate the average time required for a single master problem optimization. Similarly, columns three and five permit an estimation of the average time required for a subproblem solution. A very small fraction of the total CPU time is used to solve the subproblem, while the rest is mostly consumed in solving the master problem by CPLEX. Each subproblem solution takes 0.03 seconds on the average for the larger size problems, Test 0, Test 1 and Test 2, while the average master problem solution time is 0.7 seconds. The last column reports the memory requirements during the solution process. Since columns are rapidly generated by the subproblem, we limited the maximum memory size that can be used. This resulted in the frequent elimination of columns from the master problem constraint matrix and the reduction of the simplex CPU time. Obviously, some of the eliminated columns may be generated again.

The numerical stability of the simplex solution process depends on the number of same time constraints (21) present in the master problem. A fine tuning of CPLEX parameters related to numerical stability was needed in order to solve problems Test 0, Test 1 and Test 2. The default values of these parameters work very well, however, for problems containing the flight covering constraints (20) and only a few SDT constraints. Note that the flight covering constraints have binary coefficients in the master problem constraint matrix, while the same departure time constraints have coefficients corresponding to flight departure times (see constraints (27)). As the method produces weekly schedules and time-related data are expressed in minutes,

<table>
<thead>
<tr>
<th>Problem name</th>
<th>Number of SDT constraints</th>
<th>Number of C.G iterations</th>
<th>MP CPU (s)</th>
<th>SP CPU (s)</th>
<th>Number of generated columns</th>
<th>Memory (Mega)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 0</td>
<td>106</td>
<td>2939</td>
<td>2541.9</td>
<td>104.3</td>
<td>88827</td>
<td>4.99</td>
</tr>
<tr>
<td>Test 1</td>
<td>106</td>
<td>6745</td>
<td>3934.5</td>
<td>251.8</td>
<td>208548</td>
<td>5.49</td>
</tr>
<tr>
<td>Test 2</td>
<td>102</td>
<td>3706</td>
<td>2475.0</td>
<td>135.6</td>
<td>119895</td>
<td>5.40</td>
</tr>
<tr>
<td>Test 3</td>
<td>18</td>
<td>772</td>
<td>55.3</td>
<td>16.4</td>
<td>18137</td>
<td>0.87</td>
</tr>
<tr>
<td>Test 4</td>
<td>4</td>
<td>8</td>
<td>6.7</td>
<td>2.6</td>
<td>5071</td>
<td>0.53</td>
</tr>
<tr>
<td>Test 5</td>
<td>4</td>
<td>16</td>
<td>0.1</td>
<td>0.1</td>
<td>146</td>
<td>0.21</td>
</tr>
<tr>
<td>Test 6</td>
<td>4</td>
<td>8</td>
<td>0.1</td>
<td>0.0</td>
<td>60</td>
<td>0.18</td>
</tr>
<tr>
<td>Test 7</td>
<td>18</td>
<td>45</td>
<td>0.3</td>
<td>0.1</td>
<td>192</td>
<td>0.25</td>
</tr>
<tr>
<td>Test 8</td>
<td>24</td>
<td>780</td>
<td>15.4</td>
<td>2.1</td>
<td>6147</td>
<td>0.65</td>
</tr>
<tr>
<td>Test 9</td>
<td>76</td>
<td>1568</td>
<td>379.3</td>
<td>48.9</td>
<td>60791</td>
<td>2.43</td>
</tr>
<tr>
<td>Test 10</td>
<td>50</td>
<td>176</td>
<td>58.0</td>
<td>3.1</td>
<td>6062</td>
<td>0.71</td>
</tr>
<tr>
<td>Test 11</td>
<td>36</td>
<td>114</td>
<td>4.4</td>
<td>0.8</td>
<td>1888</td>
<td>0.56</td>
</tr>
</tbody>
</table>
these values are in range \([0, b_i - a_i]\), that is \([0, 1440]\).

The experiments showed that it was essential to eliminate same departure time constraints at a branch and bound node whenever a fixed time branching decision is imposed; this permitted a reduction in the constraint number in the simplex matrix and it decreased the number of column generation iterations required to solve a branching node. Fig. 2 depicts the variation of the number of column generation iterations with the number of branching nodes for problems Test 0, Test 1 and Test 2. Note that the number of column generation iterations at the root node of the branching tree is of 369, 323 and 291, respectively, for the three problems. It increases rapidly for the first branching nodes, and less rapidly toward the end of the branching process. Using the information in Table 3, the average number of column generation iterations per branching node is 28.5, 78 and 41.1, respectively.

5.2. Solution improvement – Final results

The only drawback of our current results is the slow convergence of the Dantzig–Wolfe decomposition/column generation process. Three improvements were devised to accelerate the convergence of this process for our problem.

First, note that an integer solution doesn’t necessarily require integer master problem variables \(\theta_i\). Indeed, let us consider an example of an integer solution found for problem Test 7. This test problem contains 18 flights. The solution at the root node of the branching tree gives an optimal objective value of 15 and presents only two fractional columns (all columns have a cost of 1). The first fractional column has a flow of 0.9166 (value of the associated \(\theta_i\) variable) and corresponds to the following path: source node (time 0); flight 17 (time 120); flight 15 (time 5115); flight 18 (time 5870); flight 5 (time 6555) and sink node. The second fractional column has a flow of 0.0834, and corresponds to the same path: source node (time 0); flight 17 (time 0); flight 15 (time 5115); flight 18 (time 5870); flight 5 (time 6555) and sink node. Note that only the departure time of flight 17 is different for this second fractional column. A solution is integer when each flight is covered by a single path. Since both columns correspond to the same path, we do not need to impose any branching decision; to obtain the optimal integer solution, we need only to compute the optimal departure time of flight 17 as a weighted combination of values 0 and 120. The optimal departure time for flight 17 is 110. The recognition of this type of integer solution was integrated into our branch and bound procedure.

![Fig. 2. Column generation iterations during the solution process.](image-url)
Second, we noticed that a large number of column generation iterations was used for very small variations on the synchronized departure times $T_m, m \in M$. To further reduce the number of iterations, we introduced a tolerance on the right hand side of constraints (21) in the master problem. Obviously, this tolerance allows for non synchronized flight departure times; to obtain an optimal synchronized solution, we slid the values of $T_m$ on the closest synchronized values. Since the time data in our test problems were given in multiples of 5, we used a tolerance of 4 minutes. For our test problems, a 4 minute tolerance allowed us to retrieve an optimal integer solution in all cases. However, large values of the tolerance may result in possible suboptimality. Smaller values of the tolerance, such as 0.4 for example, may require a longer CPU time, but lead to an optimal integer solution (provided that the time data in the problem are integer).

Third, we modified the subproblem to generate several disjoint columns at each iteration of the column generation process. The results obtained as a result of these modifications are given in Table 4. The corresponding details on the solution process are given in Table 5.

These improvements lead to very efficient solution times for our test problems due to a better convergence of the column generation method. For problems Test 0, Test 1 and Test 2, the average solution time was reduced by 97.6%. The same order of improvement is observed for the set of all

### Table 4
Improved results

<table>
<thead>
<tr>
<th>Problem name</th>
<th>Nodes</th>
<th>Arcs</th>
<th>Number of SDT constratints</th>
<th>LP sol.</th>
<th>IP sol.</th>
<th>Number of BB nodes</th>
<th>CPU (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 0</td>
<td>108</td>
<td>1255</td>
<td>212</td>
<td>11</td>
<td>11</td>
<td>18</td>
<td>79.1</td>
</tr>
<tr>
<td>Test 1</td>
<td>108</td>
<td>1255</td>
<td>212</td>
<td>10</td>
<td>10</td>
<td>14</td>
<td>82.3</td>
</tr>
<tr>
<td>Test 2</td>
<td>108</td>
<td>1237</td>
<td>204</td>
<td>10</td>
<td>10</td>
<td>12</td>
<td>59.4</td>
</tr>
<tr>
<td>Test 3</td>
<td>87</td>
<td>1279</td>
<td>36</td>
<td>7</td>
<td>7</td>
<td>9</td>
<td>20.5</td>
</tr>
<tr>
<td>Test 4</td>
<td>56</td>
<td>617</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2.1</td>
</tr>
<tr>
<td>Test 5</td>
<td>16</td>
<td>63</td>
<td>8</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0.1</td>
</tr>
<tr>
<td>Test 6</td>
<td>12</td>
<td>36</td>
<td>8</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0.1</td>
</tr>
<tr>
<td>Test 7</td>
<td>20</td>
<td>50</td>
<td>36</td>
<td>15</td>
<td>15</td>
<td>0</td>
<td>0.1</td>
</tr>
<tr>
<td>Test 8</td>
<td>26</td>
<td>106</td>
<td>48</td>
<td>5.8</td>
<td>6</td>
<td>2</td>
<td>0.2</td>
</tr>
<tr>
<td>Test 9</td>
<td>78</td>
<td>744</td>
<td>152</td>
<td>9</td>
<td>9</td>
<td>8</td>
<td>8.1</td>
</tr>
<tr>
<td>Test 10</td>
<td>52</td>
<td>407</td>
<td>100</td>
<td>7</td>
<td>7</td>
<td>6</td>
<td>2.7</td>
</tr>
<tr>
<td>Test 11</td>
<td>38</td>
<td>202</td>
<td>72</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>0.2</td>
</tr>
</tbody>
</table>

### Table 5
Analysis of the improved solution process

<table>
<thead>
<tr>
<th>Problem name</th>
<th>Number of SDT constraints</th>
<th>Number of C.G. iterations</th>
<th>MP CPU (s)</th>
<th>SP CPU (s)</th>
<th>Number of generated columns</th>
<th>Memory (Megs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 0</td>
<td>212</td>
<td>204</td>
<td>73.6</td>
<td>3.6</td>
<td>3127</td>
<td>1.56</td>
</tr>
<tr>
<td>Test 1</td>
<td>212</td>
<td>186</td>
<td>76.9</td>
<td>3.5</td>
<td>3225</td>
<td>1.62</td>
</tr>
<tr>
<td>Test 2</td>
<td>204</td>
<td>153</td>
<td>54.9</td>
<td>2.9</td>
<td>2762</td>
<td>0.40</td>
</tr>
<tr>
<td>Test 3</td>
<td>36</td>
<td>164</td>
<td>17.0</td>
<td>2.6</td>
<td>2108</td>
<td>0.90</td>
</tr>
<tr>
<td>Test 4</td>
<td>8</td>
<td>54</td>
<td>1.5</td>
<td>0.4</td>
<td>567</td>
<td>0.40</td>
</tr>
<tr>
<td>Test 5</td>
<td>8</td>
<td>11</td>
<td>0.1</td>
<td>0.0</td>
<td>41</td>
<td>0.16</td>
</tr>
<tr>
<td>Test 6</td>
<td>8</td>
<td>5</td>
<td>0.1</td>
<td>0.0</td>
<td>13</td>
<td>0.16</td>
</tr>
<tr>
<td>Test 7</td>
<td>36</td>
<td>5</td>
<td>0.1</td>
<td>0.0</td>
<td>12</td>
<td>0.16</td>
</tr>
<tr>
<td>Test 8</td>
<td>48</td>
<td>15</td>
<td>0.1</td>
<td>0.0</td>
<td>93</td>
<td>0.22</td>
</tr>
<tr>
<td>Test 9</td>
<td>152</td>
<td>129</td>
<td>6.0</td>
<td>1.5</td>
<td>1258</td>
<td>0.72</td>
</tr>
<tr>
<td>Test 10</td>
<td>100</td>
<td>90</td>
<td>1.9</td>
<td>0.6</td>
<td>827</td>
<td>0.47</td>
</tr>
<tr>
<td>Test 11</td>
<td>72</td>
<td>9</td>
<td>0.1</td>
<td>0.0</td>
<td>79</td>
<td>0.22</td>
</tr>
</tbody>
</table>
12 test problems, as the solution time was reduced on the average by 97.4%. This dramatic improvement is performed mainly on the master problem solution time. As the final results show, each subproblem solution takes 0.02 seconds on the average for the larger size problems, Test 0, Test 1 and Test 2, while the average master problem solution time is 0.37 seconds. The last column shows that the memory requirements are also reduced. While the first and third improvements also helped reduce the solution time, the second improvement strategy proved to be the most efficient for our problem.

6. Conclusions

The proposed approach has several advantages over an alternative approach proposed in Ref. [27]. First, a small number of constraints is present in the master problem formulation, that is, at most $2|N|$ constraints. Second, the subproblem solution time required throughout the resolution was very small for test problems involving up to 1279 arcs, less than 0.02 seconds at each column generation iteration. This shows that our approach is very promising for treating weekly problems with a large number of feasible connection arcs. In fact, the subproblem algorithm was independently tested in Ref. [20] and it solved problem samples with up to 80000 arcs in less than 21 seconds. Our approach can be, therefore, very efficient for applications presenting a large number of arcs and a large number of schedule synchronization constraints. Finally, a good convergence of the column generation method was obtained as a result of extensive improvements on the solution process. These improvements resulted in a dramatic reduction of the overall solution time.

References

[19] P. Hansen, B. Jaumard, M. Poggi De Aragao, Un Algorithme Primal de Programmation Linéaire


