Nationaal Lucht- en Ruimtevaartlaboratorium
National Aerospace Laboratory NLR

NLR TP 97286

Slot Allocation by Column Generation

J.M. van den Akker and K. Nachtigall

Summary

A slot allocation policy assigns a time slot for departure to each of a number of flights in order to avoid overload in control sectors and on runways. We present a new solution method for a basic version of the European slot allocation problem. The approach is based on an integer linear programming model, where for each flight there is a set of 0-1 variables each associated with one of its possible departure slots. First, column generation is used to solve the LP-relaxation. This results in the construction of a reasonable set of departure slots for each of the flights. Then, the resulting integer linear program is solved. We also present a rounding heuristic to derive feasible integral solutions from fractional solutions after each iteration of the column generation algorithm. Computational results for real-world problems are encouraging, because they indicate that compared to existing exact optimisation algorithms our method generates solutions of approximately the same quality but requires only a fraction of the computation time. Moreover, they indicate that the rounding heuristic performs well.

Keywords: Air Traffic Flow Management, Slot Allocation, Integer Linear Programming, Column Generation.

Contents

5 Tables
4 Figures

(29 pages in total)

List of tables

This page is intentionally left blank.

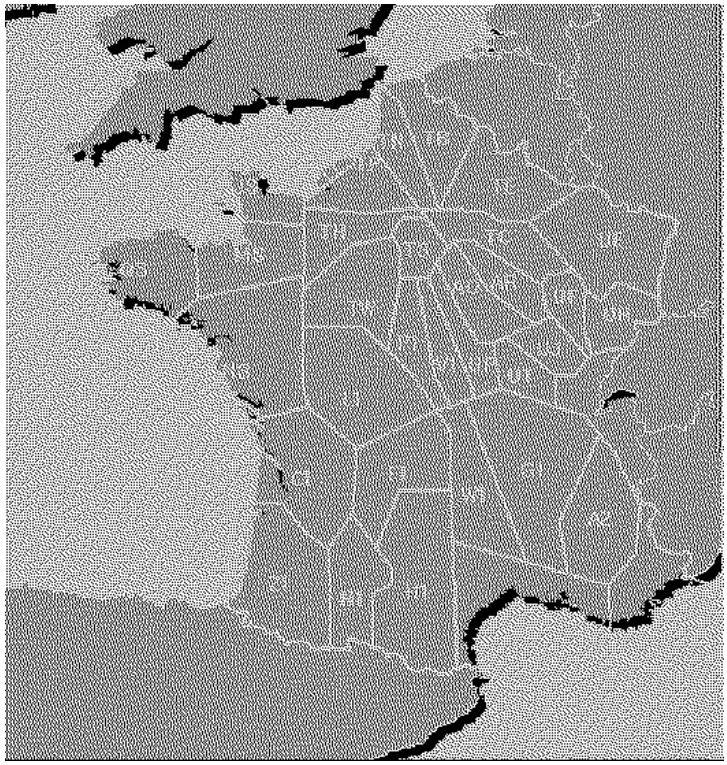

Fig. 1 French Airspace

1 Introduction

In the last decade, air traffic has increased extensively and has exceeded all forecasts. The result has been a considerable increase of delays. For example, Duytschaever [9] mentions that on Fridays in July 1992, 57 % of the flights in Western Europe were delayed and that the average delay of these flights was 25 minutes. The situation is expected to get worse; an ATAG study [4] expects the number of flights in Western Europe to increase by 54% between 1990 and 2000 and by 110% between 2000 and 2010, which will amount to more than 11 million flights per year.

On the short-term, the only way to deal with the increasing amount of traffic is to use the existing airspace capacity more efficiently. This capacity is characterized as follows. The European airspace is divided into different control sectors; Figure 1 depicts the division of the French airspace. In each sector a certain number of controllers are responsible for maintaining safety. To guarantee that the controllers are able to do this, overload of the sector should be avoided. This is reflected in the so-called sector capacity. In literature the sector capacity is either defined by the maximum number of aircraft which during a given time period are allowed to enter the sector, or by the maximum number of aircraft which during a given time period are allowed to be within

the sector. Airports have an arrival and a departure capacity which indicate the maximum number of aircraft which during a given time period are allowed to land at and to depart from the airport, respectively.

A way to make more efficient use of the existing airspace capacity is slot allocation. A slot allocation policy assigns a time slot for departure to each of a number of flights. The idea is as follows. Suppose that, if an aircraft departs at a certain time, it encounters overloaded sectors or cannot land immediately at its arrival airport because of congestion. This leads to airborne delay, i.e., the aircraft has to fly in circles in a holding area or has to reduce its speed. Since airborne delays are more expensive and less safe than delays which are incurred on the ground, it can be more beneficial to delay the departure of the flight, i.e., select another time slot for departure of the flight.

Summarizing, the slot allocation problem is to assign a time slot for departure to each of a number of flights in such a way that for each sector and each airport the capacity is not exceeded and that the costs are minimal.

If the departure slot of a flight falls later than its scheduled departure time, then allocating the slot amounts to imposing a ground-holding delay. Therefore, slot allocation is also known as ground-holding.

Currently, slot allocation for flights in Western-Europe is performed by the CASA system of Eurocontrol (see Philipp and Gainche [15]). CASA is based on a first-come-first-served heuristic, i.e., departure slots are assigned to flights in the order of their scheduled departure time.

In the literature different models for the slot allocation or ground-holding problem have been presented. The first were models for the so-called Single Airport Ground-Holding problem, in which a number of flights heading for one given congested airport are considered. We refer to Andreatta, Odoni, and Richetta [3] for an overview. For the problem involving a complete set of airports with limited departure and arrival capacities, but no restrictions on the capacities of the control sectors, different optimisation methods have been proposed. Integer linear programming formulations have been studied by Vranas [17], Vranas, Bertsimas, and Odoni [6], and Bertsimas and Stock [7], and have been compared by Andreatta and Brunetta [1]. Heuristics based on priority rules have been proposed by Andreatta, Brunetta, and Guastella [2] and Navazio and Romanin-Jacur [14].

The assumption that capacities of control sectors do not pose a restriction, may hold for the situation

in the USA, but certainly not for that in Europe. The models including sector capacities are special cases of models for the Generalized Traffic Flow Management Problem (TFMP), in which optimal ground-holding as well as airborne delays are computed. Integer linear programming models for the TFMP have been proposed by Helme [11], Lindsay, Boyd, and Burlingame [12], Bertsimas and Stock [7], and Maugis [13]. Vranas and Psaraftis [18] performed computational experiments for the slot allocation problem with the model in [7] and proposed another model for this problem. The common characteristic of the models in [12], [7], and [13] is that they use time-indexed variables, such as for example binary variables u_{ft} indicating that flight f departs at time t. Such models are often very strong in the sense that the LP-relaxation provides a very good bound [a]. An important disadvantage is the large number of decision variables [b], which leads to large computation times. Experiments in [18] show that for real-world examples with at least 6000 flights solving the LP-relaxation requires already an hour on a SGI Power Challenge and finding an integral solution turns out to be impractical. On the other hand, they indicate that for certain instances the solution provided by the first-come-first-served heuristic, i.e., the type of heuristic that is used in the CASA system, is 40 % above the optimum.

In this paper, we consider the integer linear programming model which was studied by Maugis [13]. We present an LP-based solution method in which we solve the LP-relaxation by column generation to deal with the large number of variables. It consists of the following steps:

(1) Use a heuristic to find a feasible solution of the problem.
(2) Solve the LP-relaxation by column generation.
(3) Solve the resulting integer linear program, i.e., the integer linear program consisting of the variables generated in the previous step.

Since we solve the integer linear program restricted to the set of variables which are generated by the column generation algorithm, our method is a heuristic. The idea is that the column generation algorithm will generate for each flight a set of good departure slots. Our computational experiments indicate that for real-world examples the LP-based method provides nearly optimal solutions, while requiring significantly less computation time than finding the optimal solution by solving the complete integer linear programming problem.

In each iteration of the column generation algorithm a fractional solution is found. We have implemented a rounding heuristic to derive feasible integral solutions from these fractional solutions. The same type of heuristic has shown to work very well for single-machine scheduling problems

[a] For example, Maugis [13] reports that for all tested real-world examples the integrality gap did not exceed 0.07 %.
[b] For a problem instance with approximately 4500 flights the model contains about 1.000.000 binary variables (see Section 4).

(see Van den Akker [16] and Hall, Schulz, Shmoys, and Wein [10]). It turns out that this heuristic enables us to find rather good solutions quickly, which is interesting from a practical point of view.

Moreover, we extend our model to handle connections between flights. Two flights can be connected because they have to be performed by the same aircraft, or because an airline wants to guarantee that passengers can change from one flight to the other. Connected flights are also studied in [6], [7], [13], and [18]. We show that our LP-based method can be applied in this situation, too.

This paper is organized as follows. In Section 2, we present the integer linear programming formulation. In Section 3, we discuss our LP-based method, especially the column generation algorithm and the rounding heuristic. Moreover, we treat connected flights. Section 4, we report on computational results for real-world examples. Finally, in Section 5 we give our conclusion.

2 The basic model and general assumptions

Notation	Symbol
Decision Variables	
x_{fd}	indicates if flight f departs at time d
Sets	
Set of sectors	$\mathcal{S}$
Set of time periods for capacity definition	$\mathcal{T}$
Set of flights	$\mathcal{F}$
Set of flights crossing sector s	$\mathcal{F}(s)$
Set of sectors crossed by flight f	$\mathcal{S}(f)$
Set of all possible departure slots of flight f	$\mathcal{D}^*(f)$
Set of generated departure slots of flight f	$\mathcal{D}(f)$
Times	
Length of period for sector capacity constraints	τ
Time period induced by number t for capacity definition	$T := [t * \tau, (t+1) * \tau)$
Length of period between two consecutive departure slots	δ
Maximum delay for one flight	G
Scheduled departure time of flight f	d_0^f
Duration after departure at which f enters sector s	$d_{s,\text{in}}^f$
flight time of flight f	D_f
Other symbols	
Costs if flight f has departure time d	ω_{fd}
Capacity of sector s during period T	$C_s(T)$

Table 1 Notation

We consider a set $\mathcal{F}$ of n flights. For each flight $f \in \mathcal{F}$ are given: a departure and an arrival airport, and a scheduled departure time d_0^f. Our task is to find for each flight a departure time $d \geq d_0^f$ such that at any time the sector and airport capacities are not exceeded, and the total costs are minimal. The notation that is used throughout the paper is summarized in Table 1.

To define sector capacity we apply time discretization, i.e., time is divided into periods of a given length τ. Time period T, induced by t, is given by the interval $T := [t * \tau, (t+1) * \tau)$. Recall that in the literature two definitions of sector capacities occur. The capacity $C_s(T)$ of sector s in period T can either be

(1) the maximum number of flights which are allowed to enter s during T, or
(2) the maximum number of flights which are allowed to be present within s during (a part of) T.

We consider the first definition. The main reason is that this definition is used in current practice by

Eurocontrol. Moreover, when the second definition is used, the LP-relaxation is very weak, which makes the problem much harder to solve. In our model we will consider arrival and departure capacities of airports as a special type of sector capacities, so it is not necessary to introduce separate notation for these capacities, here.

Slot allocation only deals with fixing departure times of flights, and not with routing flights. Therefore, we assume that for each flight the route is fixed. This means, that for each flight f we have a list of sectors $\mathcal{S}(f)$ that are crossed by f and for each sector we know the time interval during which flight f crosses sector s if it departs at time d, especially we know the time $d + d^f_{s,\text{in}}$ at which flight f enters sector s if it departs at time d.

To keep our model tractable, we also discretize the set of possible departure times: time is divided into periods of a given length δ and aircraft are assumed to depart at the beginning of such time periods. We assume that all data d^f_0 and $d^f_{s,\text{in}}$ are integral multiples of δ. Furthermore, like most of the existing models, our model includes an upper bound G for the delay of an individual flight. It follows that the set $\mathcal{D}^*(f)$ of possible departure times for flight f is given by $\mathcal{D}^*(f) = \left\{d \,:\, d = d^f_0 + j * \delta \ (j = 0, 1, ..., N_f)\right\}$, where $N_f = \lfloor \frac{G}{\delta} \rfloor$. Note that in the definition of sector capacities, we divided time into periods of length τ. We assume that the division into periods of length δ is a refinement of the division into periods of length τ, i.e., each 'capacity' period of length τ contains an integral number of 'departure' periods of length δ, which implies that τ is a multiple of δ. The above type of discretization of departure times is used in most models (see e.g. [13, 18]).

We obtain the following integer linear programming model (GHP) for the slot allocation problem. For each flight f we introduce a set of decision variables x_{fd} ($d \in \mathcal{D}^*(f)$), where x_{fd} equals 1 if flight f has departure time d, and 0 otherwise.

$$
\begin{aligned}
(\text{GHP}) \qquad \text{minimize} \quad & \sum_{f \in \mathcal{F}} \sum_{d \in \mathcal{D}^*(f)} \omega_{fd} x_{fd} \\
\text{s.t.} \quad & \\
(\text{a}) \qquad & \sum_{d \in \mathcal{D}^*(f)} x_{fd} = 1 \qquad \forall\, f \in \mathcal{F} \\
(\text{b}) \qquad & \sum_{f \in \mathcal{F}(s)} \sum_{d \in \mathcal{D}^*(f) : d + d^f_{s,\text{in}} \in T} x_{fd} \le C_s(T) \qquad \forall\, s \in \mathcal{S},\ T \in \mathcal{T} \\
(\text{c}) \qquad & x_{fd} \in \{0, 1\} \qquad \forall f \in \mathcal{F}, d \in \mathcal{D}^*(f).
\end{aligned}
$$

The coefficient ω_{fd} is defined as the cost incurred if flight f has departure time d. Constraints (a) guarantee that exactly one departure time is assigned to each flight and constraints (b) are the capacity constraints. Arrival and departure capacities of airports can be included in the constraints (b) in the following way. We consider each airport as two sectors. The first sector is used by flights departing from the airport and the second sector by flights arriving at the airport. For the departure sector we have $d^f_{s,\text{in}} = 0$ for all flights f departing from the airport, and for the arrival sector we have $d^f_{s,\text{in}} = D_f$ for all flights f arriving at the airport, where D_f is the flight time of flight f. We assume that D_f is a multiple of δ. Note that it is possible to base the airport capacity constraints on periods of length different from the length τ used to define sector capacities. These periods should however contain an integral number of periods of length δ.

Observe that our model assigns departure times, instead of departure slots, which are defined as intervals, to flights. Let $d \in \mathcal{D}^*(f)$; d is hence at the beginning of a period of length δ. Since all problem data d^f_0, $d^f_{s,\text{in}}$, and D_f are multiples of δ it follows that, if a flight f departs during the interval $[d, d+\delta)$, then it enters each of the sectors it crosses in exactly the same period of length τ as it does when it departs exactly at time d. For this reason, feasible solutions of our model (GHP) are still feasible if we allow a flight with departure time d to depart during the interval $[d, d+\delta)$. Hence, in our model, assigning departure time d to a flight is equivalent to assigning departure interval or slot $[d, d+\delta)$ to the flight. We prefer to use departure times, since this facilitates the analysis of our LP-based method.

Observe that the number of decision variables is approximately $\frac{G*|\mathcal{F}|}{\delta}$ which is rather large for real-world examples. To handle this large number of variables, we propose a column generation scheme to solve the LP-relaxation of (GHP) which will be discussed in detail in the next section.

3 Column Generation

3.1 The LP-based method

In this section we describe our LP-based method in detail. Recall that the method consists of the following steps:

(1) Use a heuristic to find a feasible solution of the problem.
(2) Solve the LP-relaxation by column generation.
(3) Solve the resulting integer linear program, i.e., the integer linear program consisting of the variables generated in the previous step.

Moreover, after each iteration of the column generation algorithm we use a rounding heuristic to obtain a feasible integral solution from the current fractional solution. This heuristic will be discussed in Section 3.3

Step (1): The FCFS heuristic

Before starting the column generation algorithm, it is necessary to have a feasible solution. In the first step of the method such a solution is derived by the following first-come-first-served heuristic (FCFS). Recall that the CASA system of Eurocontrol also uses a FCFS heuristic. Our heuristic proceeds as follows:

(1) Order the flights on their scheduled departure times.
(2) Handle the flights in this order: take for flight f the earliest possible departure time $d \geq d_0^f$ such that no sector capacity overload occurs.

Step (2): Solving the LP-relaxation by column generation

In the second step of the method, we use column generation to solve the LP-relaxation of (GHP), i.e., the linear programming problem obtained from (GHP) by replacing the integrality constraints $x_{fd} \in \{0,1\}$ by the simple sign constraints $0 \leq x_{fd}$. Note, that this and constraints (a) imply $0 \leq x_{fd} \leq 1$. Clearly, the optimal value of the LP-relaxation is a lower bound on the optimal value of the integer programming problem. A column generation algorithm always works with a restricted linear programming problem in the sense that only a subset of the variables is taken into account; other variables are generated only when they are necessary to improve the current solution. We refer to Barnhart et al. [5] for a description of column generation and its applications.

Let $\mathcal{D}(f)$ be a subset of the set of possible departure times of flight f which contains for each flight f the departure time generated by the above FCFS heuristic. Let (GHP2) be the restriction of the LP-relaxation of (GHP) to the variables x_{fd} with $d \in \mathcal{D}(f)$. Then (GHP2) has a feasible solution. (GHP2) is given by:

$$
\begin{aligned}
(\text{GHP2}) \qquad & \text{minimize} \quad \sum_{f \in \mathcal{F}} \sum_{d \in \mathcal{D}(f)} \omega_{fd} x_{fd} \\
& \text{s.t.} \\
& (\text{a}') \qquad \sum_{d \in \mathcal{D}(f)} x_{fd} = 1 \qquad \forall\, f \in \mathcal{F} \\
& (\text{b}') \qquad \sum_{f \in \mathcal{F}(s)} \sum_{d \in \mathcal{D}(f): d + d^f_{s,\text{in}} \in T} x_{fd} \le C_s(T) \qquad \forall\, s \in \mathcal{S},\ T \in \mathcal{T} \\
& (\text{c}') \qquad 0 \le x_{fd} \qquad \forall f \in \mathcal{F},\ d \in \mathcal{D}(f).
\end{aligned}
$$

From the theory of linear programming it is known that after solving the restricted problem to optimality, each included variable has nonnegative reduced cost. The reduced cost of variable x_{fd} is defined in the following way. Let $\mathcal{S}(f)$ be the set of all sectors which are crossed by flight f. Let u_j be the dual variables corresponding to constraints (a'), and v_{sT} those corresponding to the constraints (b'). Then the reduced cost $\hat{\omega}_{fd}$ of the variable x_{fd} is given by

$$
\hat{\omega}_{fd} := \omega_{fd} - u_f - \sum_{(s,T): s \in \mathcal{S}(f), d + d^s_{f,\text{in}} \in T} v_{sT}. \tag{1}
$$

If each variable outside the restricted problem also has nonnegative reduced cost, then we have found an optimal solution of the original problem, i.e., of the LP-relaxation of (GHP). On the other hand, if there exist variables with negative reduced cost, then one or more of these variables have to be added to the restricted problem (GHP2) and the linear program has to be solved again. After that the existence of variables with negative reduced cost is checked again, etc. The process of identifying variables with negative reduced cost and re-optimizing the linear program is called an iteration of the column generation algorithm.

The main property of column generation is that the existence of variables with negative reduced cost is not checked by enumerating all variables, but in a faster way by solving an optimization problem, which is called the pricing problem. To apply column generation successfully, it is very important to be able to solve the pricing problem efficiently.

3.2 The Pricing Problem

Solving the pricing problem amounts to finding variables $x_{fd} \in \mathcal{D}^*(f) \setminus \mathcal{D}(f)$, i.e., variables that are not in the restricted problem (GHP2), with reduced cost $\hat{\omega}_{fd} < 0$. This can be done by minimizing for each flight f the reduced cost $\hat{\omega}_{fd}$ over all possible departure times d.

The indicator function

$$\rho^f_{sT}(d) = \begin{cases} 1 & \text{if } d + d^s_{f,\text{in}} \in T, \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

indicates whether flight f enters sector s with $s \in \mathcal{S}(f)$ during time period T if has departure time d. Obviously, ρ^f_{sT} is a jump function with two jumps. Note that we assume that all problem data d^f_0 and $d^s_{f,\text{in}}$ are multiples of δ, and that a period T of length τ contains an integral number of periods of length δ. As a result, the jumps are at points which are multiples of δ. Now the function

$$\rho^f(d) = \sum_{\{(s,T)\,:\, s \in \mathcal{S}(f), T \in \mathcal{T}\}} v_{sT} \cdot \rho^f_{sT}(d) \tag{3}$$

is also a jump function. We assume that the cost ω_{fd} are increasing with respect to the associated departure time d, i.e., for $d \leq d'$ we have $\omega_{fd} \leq \omega_{fd'}$. For example the cost function representing the total amount delay has this property. Then the function $g(d) := \omega_{fd} - \rho^f(d)$ takes its minimum at one of the finite jump points of ρ^f. This is shown in Figures 2 and 3 for the case that the cost function equals the total delay, i.e., $\omega_{fd} = d - d^f_0$.

It now follows that for a given flight f, the minimum reduced costs over all possible departure times, i.e.,

$$\min_{d \geq d^f_0} \hat{\omega}_{fd} = -u_f + \min_{d \geq d^f_0} \left(\omega_{fd} - \rho^f(d) \right) \tag{4}$$

can be calculated by enumerating all jump positions of ρ^f.

If d' is such that $\hat{\omega}_{fd'} = \min_{d \geq d^f_0} \hat{\omega}_{fd} < 0$, then $x_{fd'}$ will be added to (GHP2) and after that (GHP2) has to be solved again. For computational efficiency it is beneficial to look first for all flights f for variables $x_{fd'}$ with $\hat{\omega}_{fd'} < 0$ and then put them all into the LP-relaxation (GHP2),

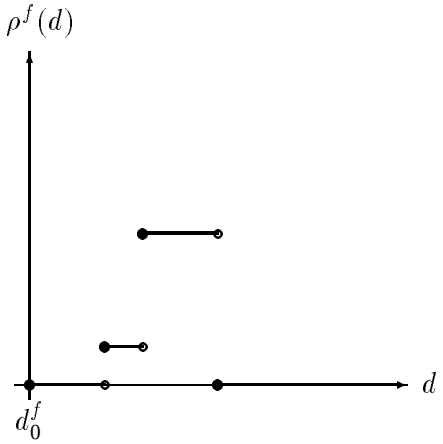

Fig. 2 Jump function $\rho^f(d)$

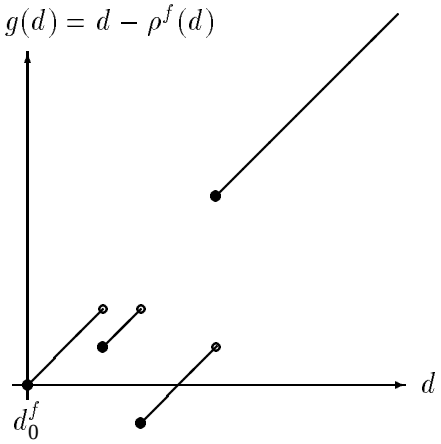

Fig. 3 Resulting reduced cost $g(d) = d - \rho^f(d)$ *with delay cost* $\omega_{fd} = d - d_0^f$

before solving it again.

The computational effort required to calculate the minimum of $\hat{\omega}_{fd}$ depends on the number of jumps of ρ^f. Since every elementary jump function ρ^f_{sT} has exactly 2 jumps, an upper bound on this number is given by $2 \cdot |\{(s,T) : s \in \mathcal{S}(f), T \in \mathcal{T},\ v_{sT} \neq 0\}|$. From the theory of linear programming, it follows that from among the capacity constraints (b') only non-trivial constraints $\sum_{(f,d)\in Q} x_{fd} \leq C_s(T)$ with $|Q| > C_s(T)$ can have dual variables $v_{sT} \neq 0$. The computational results show that the number of such constraints is relatively small. [a)] This explains why the pricing problem can be solved very quickly and shows why column generation is much faster than the simplex method.

Step (3): Solving the resulting integer linear program

The resulting integer linear programming problem, i.e., the integer linear program consisting of the variables generated by the column generation algorithm is solved by the commercial software package CPLEX4.0 [8]. CPLEX applies a branch-and-bound algorithm in which lower bounds are obtained by solving linear relaxations.

[a)] For one of the instances with about 4500 flights the column generation generates approximately 10,000 decision variables and between 600 and 750 capacity constraints (s,T) for which it is possible that $v_{sT} \neq 0$.

3.3 A Rounding Heuristic

The following heuristic can be used to obtain good integral solutions while solving the LP-relaxation. After each iteration of the column generation algorithm we have a fractional solution. We can use such a fractional solution in a heuristic to find a feasible integral solution of the slot allocation problem (GHP) in the following way. Let $\tilde{x}$ be a fractional solution. Define the average departure time $\bar{d}_f$ of flight f by

$$\bar{d}_f = \sum_{d \in \mathcal{D}(f)} \tilde{x}_{fd} * d.$$

For example, if $x_{f3} = \frac{1}{2}$ and $x_{f5} = \frac{1}{2}$, then $\bar{d}_f = 4$. The heuristic consists of the following steps:

(1) Order the flights on their average departure times $\bar{d}_f$.

(2) Apply a FCFS heuristic based on this order.

The same type of heuristic was implemented for a time-indexed formulation for single-machine scheduling problems by Van den Akker [16]. Her computational results showed that the heuristic provided very good solutions. Moreover, Hall, Schmoys, Schulz, and Wein [10] derived a worst case-ratio of 2 of the rounding heuristic for a slightly modified formulation. Our computational results will show that this type of heuristic performs well for slot allocation, too.

3.4 Connected Flights

In this section we consider connected flights. Connected flights were also studied in [6], [7], [13], and [18]. Assume that two flights f_1, f_2 have to be performed by the same aircraft, or that an airline wants to guarantee that passengers can change from f_1 to f_2, for example in a hub-and-spoke system. In such situations we say that the two flights are connected. If f_1 and f_2 are connected flights, then the arrival airport of flight f_1 is the departure airport of flight f_2. Moreover, between the arrival of flight f_1 and departure of flight f_2 there has to be a so-called turn around time q. If the flights have to be performed by the same aircraft, this turn around time is required to unload, clean, and then reload the aircraft. If passengers have to change from one flight to the other, the turn around time is required to give the passengers the time get off the first aircraft, move to another gate and get on the second aircraft.

We consider the situation where flights can only be connected in pairs, i.e., each flight can only be connected to one other flight. We say that two connected flights f_1 and f_2 are consecutive flight legs of one flight f. Now, the departure times d_1 of flight f_1 and d_2 of flight f_2 have to fulfill $d_2 \geq d_1 + D_{f_1} + q$, where D_{f_1} denotes the flight time of f_1 and q the turn around time. We assume that, like the other problem data, q is a multiple of δ. We do not handle the consecutive flight legs by independent decision variables $x_{f_1 d_1}$ and $x_{f_2 d_2}$, but we associate a decision variable x_{fd} with each pair of departure times $d = (d_1, d_2)$ such that $d_1 \geq d_0^{f_1}$, $d_2 \geq d_0^{f_2}$, and $d_2 \geq d_1 + D_{f_1} + q$.

If f is a flight consisting of two consecutive flight leg, then $\mathcal{D}^*(f)$ denotes the set of all feasible departure time pairs $d = (d_1, d_2)$. Let $\mathcal{F}'$ be the subset of such flights and $\mathcal{F}'(s)$ the subset of flights in $\mathcal{F}'$ for which at least one flight leg crosses sectors s. Then the capacity constraint (b) for sector s during period T has to be replaced by

$$(\tilde{b}) \qquad \sum_{f \in \mathcal{F}(s) \setminus \mathcal{F}'(s)} \left(\sum_{d \in \mathcal{D}^*(f) \,:\, d + d^s_{f,\text{in}} \in T} x_{fd} \right) + \sum_{f \in \mathcal{F}'(s)} \left(\sum_{d=(d_1,d_2) \in \mathcal{D}^*(f) \,:\, d_1 + d^s_{f_1,\text{in}} \in T} x_{fd} + \sum_{d=(d_1,d_2) \in \mathcal{D}^*(f) \,:\, d_2 + d^s_{f_2,\text{in}} \in T} x_{fd} \right) \leq C_s(T).$$

Recall that the pricing problem is solved by determining for each flight the variable x_{fd} with minimal reduced cost. Clearly, for flights containing a single flight leg this can be done exactly as explained in Section 3.1. For flights consisting of two consecutive flight legs the minimal reduced cost are given by the following minimum over the departure times d_1 and d_2:

$$-u_f + \min_{d_2 - D_{f_1} - q \geq d_1 \geq d_0^{f_1} \wedge d_2 \geq d_0^{f_2}} \left(\omega_{fd} - \rho^{f_1}(d_1) - \rho^{f_2}(d_2) \right), \tag{5}$$

where u_f is the dual variable corresponding to constraint (a'), $\omega_{fd} = \omega_{f_1 d_1} + \omega_{f_2 d_2}$ is given by the sum of costs for both flight legs, and the functions $\rho^{f_1}(d_1)$ and $\rho^{f_2}(d_2)$ are as defined in (3). Again, we assume that ω_{fd} is a monotonically increasing function of d,, i.e., if for $d = (d_1, d_2)$ and $d' = (d_1', d_2')$ we have that $d_1 \leq d_1'$ and $d_2 \leq d_2'$, then $\omega_{fd} \leq \omega_{fd'}$. Recall that the functions ρ^{f_1} and ρ^{f_2} are jump functions. Let $\mu_0^k, ..., \mu_{m_k}^k$ denote the jumps of the function ρ^{f_k} (for $k = 1, 2$). Then

$$\begin{aligned}
\xi^* &= \min_{d_2 - D_{f_1} - q \geq d_1 \geq d_0^{f_1} \wedge d_2 \geq d_0^{f_2}} \left(\omega_{fd} - \rho^{f_1}(d_1) - \rho^{f_2}(d_2) \right) \\
&= \min_{d_2 - D_{f_1} - q \geq d_1 \geq d_0^{f_1} \wedge d_2 \geq d_0^{f_2}} \left(\omega_{f_1 d_1} + \omega_{f_2 d_2} - \rho^{f_1}(d_1) - \rho^{f_2}(d_2) \right) \\
&= \min_{d_1 \geq d_0^{f_1}} \left(\omega_{f_1 d_1} - \rho^{f_1}(d_1) + \min_{d_2 \geq d_1 + D_{f_1} + q \wedge d_2 \geq d_0^{f_2}} \left(\omega_{f_2 d_2} - \rho^{f_2}(d_2) \right) \right)
\end{aligned}$$

We consider the function

$$\tilde{\rho}(d_1) = \min_{d_2 \geq d_1 + D_{f_1} + q \wedge d_2 \geq d_0^{f_2}} \left(\omega_{f_2 d_2} - \rho^{f_2}(d_2) \right) .$$

Recall that $\omega_{f_2 d_2} - \rho^{f_2}(d_2)$ can be minimized over d_2 by enumerating the jump points $\mu_0^2, \mu_1^2 \ldots, \mu_{m_2}^2$ of $\rho^{f_2}(d_2)$. If d_1 increases, then the number of jump points μ_k^2 satisfying $\mu_k^2 \geq d_1 + D_{f_1} + q$ decreases and hence $\tilde{\rho}(d_1)$ increases. More precisely, for any $d_1 = \mu_k^2 - D_{f_1} - q + \delta$ the function value $\tilde{\rho}(d_1)$ can be larger than for the previous departure time $d_1 = \mu_k^2 - D_{f_1} - q$. We conclude that $\tilde{\rho}(d_1)$ is an increasing jump function whose value can increase just after the points $\mu_k^2 - D_{f_1} - q$. Since d_1 only takes values that are a multiple of δ, the minimum of $\tilde{\rho}(d_1)$ is achieved in one of the points in the set $\{\tilde{\mu}_0, \ldots, \tilde{\mu}_{m_2}\}$, where $\tilde{\mu}_k := \mu_0^2 - D_{f_1} - q + \delta$. It now follows that also the function $\xi(d_1) := \tilde{\rho}(d_1) - \rho^{f_1}(d_1)$ is a jump function that can be minimized by evaluating the function in the $m_1 + m_2$ points in the set $\{\mu_0^1, ..., \mu_{m_1}^1, \tilde{\mu}_0, ..., \tilde{\mu}_{m_2}\}$. Hence

$$\xi^* = \min_{d_1 \geq d_0^{f_1}} \left(\omega_{f_1 d_1} + \xi(d_1) \right)$$

can be computed by evaluating $\omega_{f_1 d_1} + \xi(d_1)$ for all $m \leq m_1 + m_2$ jump positions of $\xi(d_1)$. Since ξ^* can be calculated in at most m steps, the pricing problem can be solved within $O(m_1 + m_2)$ time.

In a similar way, we can handle flights with more than two consecutive flight legs.

4 Computational results

For our computational experiments we used the real-world data which were also used by Maugis [13]. They are available on the Internet:

'http://www.cenaath.cena.dgac.fr/∼ maugis/ghp/readme.html',

and contain data representing the traffic in the French airspace during three days in 1994: Sunday May 8, Wednesday June 1, and Thursday June 2. Table 2 reports the number of flights $|\mathcal{F}|$ and the number of capacity constraints $|\mathcal{C}|$. For more details on the instances we refer to [13]. The experiments in [13] consisted of solving the complete integer linear programming problem (GHP) for each of these instances.

Our computer runs have been performed on a SUN Sparc 5 workstation with 196 MB core memory. The linear programs have been solved using the commercial software package CPLEX 4.0 [8]. Since the experiments of Maugis [13] were also performed on a SUN Sparc 5, our computation times can directly be compared to Maugis' computation times.

Like in [13], we set the length of the 'capacity' period τ equal to 30 minutes. As cost function we consider the total amount of delay, i.e., we define $\omega_{fd} = d - d_0^f$. Furthermore, in all instances the maximum delay G for one flight has been set to 240 minutes.

We performed two series of experiments. In the first series we set δ equal to the values that were used by Maugis, i.e., δ is equal to 5 for Wednesday June 1, and δ is equal to 10 for the other two days. In the second series we set δ equal to 1.

It turns out that in the given instances, not all of the data are multiples of δ for δ equal to 5 or 10. Since our solution method is based on the fact that all problem data are multiples of δ, we rounded all scheduled departure times to the nearest multiple of δ. This is one of the reasons why the total amount of delay in our integral solutions differs from the total amount of delay found by Maugis. For the experiments with δ equal to 1, we used the original data. The results for $\delta = 5, 10$ are

Day	$\|\mathcal{F}\|$	$\|\mathcal{C}\|$
Wednesday June 1	4551	1147
Thursday June 2	4753	1153
Sunday May 8	4335	1134

Table 2 Traffic scenarios used for the experiments.

Scenario			Results of Maugis		Column Generation					
Date	δ	Z_{FCFS}	Z_I^*	time*	Z_{LP}	Z_I	time	n_{col}	n_{var}	n_{cons}
Wed	5	57920	47895	5003	47622.5	47655	490	8443	218448	614
Thu	10	59690	33580	7672	33630	33630	670	9826	114072	739
Sun	10	83920	45810	6183	46940	46940	630	10120	104040	620

Table 3 Computational results of Maugis compared to column generation.

presented in Table 3 and the results for $\delta = 1$ are presented in Table 4. In the tables we give the following quantities:

Date: traffic scenario
Z_{FCFS}: objective value of the solution found by the FCFS heuristic
Z_{LP}: optimal value of the LP-relaxation
Z_I: objective value of an integer solution with relative objective difference 0.05. [a)]
time: computation time in seconds
n_{col}: number of variables generated by column generation
n_{var}: total number of variables of the model (GHP)
n_{cons}: number of non-trivial capacity constraints $\sum_{(f,d)\in Q} x_{fd} \leq c$ with $|Q| > c$.

Additionally, in Table 3 we give:

Z_I^*: objective value of an integer solution given in [13]
time*: computation time in seconds reported in [13]

Table 4 shows that the objective value Z_I of the best integral solution that we have found is very close to the optimal value Z_{LP} of the LP-relaxation. This indicates that the solutions found by our method are nearly optimal, and hence practically as good as the solutions found by Maugis. For Sunday May 8 our solution even is optimal.

[a)] This is a parameter value used by CPLEX which forces the mixed integer optimization to ignore integer solutions that do not improve the best integer solution found so far by at least the percentage that is given by the parameter, i.e., by at least 0.05 %. This limits the number of nodes in the branch-and-bound tree but has the disadvantage that the best integer solution could be ignored (see [8]).

Scenario			Column Generation					
Date	δ	Z_{FCFS}	Z_{LP}	Z_I	time	n_{col}	n_{var}	n_{cons}
Wed	1	55622	45968,75	45983	440	9278	1092240	634
Thu	1	52934	28486,00	28508	568	10303	1140720	763
Sun	1	76468	40522,50	40523	640	10620	1040400	671

Table 4 Computational results of column generation for $\delta = 1$.

Table 3 shows that our solution method is much faster than solving the complete integer linear program as was done by Maugis. For the given examples our method requires about 10% of the computation time required for solving the complete integer linear program.

Observe that the simplex method finds non-basic variables with negative reduced cost by enumerating all variables, whereas column generation finds these variables by solving the pricing problem. In our algorithm the pricing problem is solved by investigating for every flight the jumps of a specific cost function. The experiments have shown, that the average number of jumps per cost function is approximately 30, which is relatively small. As a result, the computation time needed to solve the pricing problem is only a fraction of the time required to enumerate all variables, which is an important reason why our method is faster than the method of Maugis.

Moreover, the tables show that the number of variables generated by the column generation algorithm, especially for $\delta = 1$, is relatively small compared to the number of variables of the complete integer program (GHP). Also the number of non-trivial capacity constraints is relatively small compared to the number of constraints in (GHP), which is about 1150 (see Table 2). Hence, the integer linear program that is solved by CPLEX in the third step of our method is considerably smaller than (GHP), which is another reason why our method is faster.

A comparison of the results in Tables 3 and 4 demonstrates that the use of $\delta = 1$ will significantly improve the solution, i.e., reduce the total amount of delay in the solution. However, it does not increase the number of generated variables by column generation nor the computation time. This suggests that column generation can handle small numbers δ which leads to good solutions.

A very interesting result is that the solutions of the FCFS algorithm are improved considerably by the linear programming approach. The tables show that the delays of the FCFS solutions are reduced by 17% up to 48% by using our LP-based method or by computing the optimal solution by solving the complete integer linear programming problem. Similar results were obtained from the experiments described in [18].

Iteration	time	Z_{heur}	Z_{LP}
1	7.68	52934	52934
2	30.82	52934	52934
3	220.56	34975	33384
4	340.86	32216	29561.4
5	388.90	30545	28643.8
6	417.96	30315	28510
7	442.11	30080	28488.5
8	464.63	29912	28487
9	486.72	30058	28486
10	508.40	30049	28486

Table 5 Results of rounding heuristic for Thu-2-June-1994 and $\delta = 1$.

Recall that after each iteration of the column generation algorithm the rounding heuristic is applied. For Thursday June 2 and $\delta = 1$, we present the results of this heuristic in Table 5. In this table we give the following quantities:

time: computation time until the end of the iteration
Z_{heur}: objective value of the solution found by the rounding heuristic
Z_{LP}: objective value of the LP-relaxation after this iteration.

Details of the computer run for Thursday, June 2 and $\delta = 1$ are depicted in Figures 4 and 5. The left part of Figure 4 shows the objective value of the relaxation (GHP2) and the objective value of the solution found by the rounding heuristic after each iteration of the column generation algorithm. The right part shows the objective value of the integer solution found by our method, i.e., the optimal solution of the integer linear program resulting from the column generation algorithm. Figure 5 contains the number of variables in the relaxation (GHP2) after each iteration of the column generation algorithm. The data in both figures are presented as a function of the computation time. For the other instances a similar table and similar figures can be given. Since they show a similar behaviour, these are omitted.

Table 5 and Figure 4 show that the objective value of the LP-relaxation and the objective value of the solution found by the rounding heuristic differ by only a few percent. This indicates that

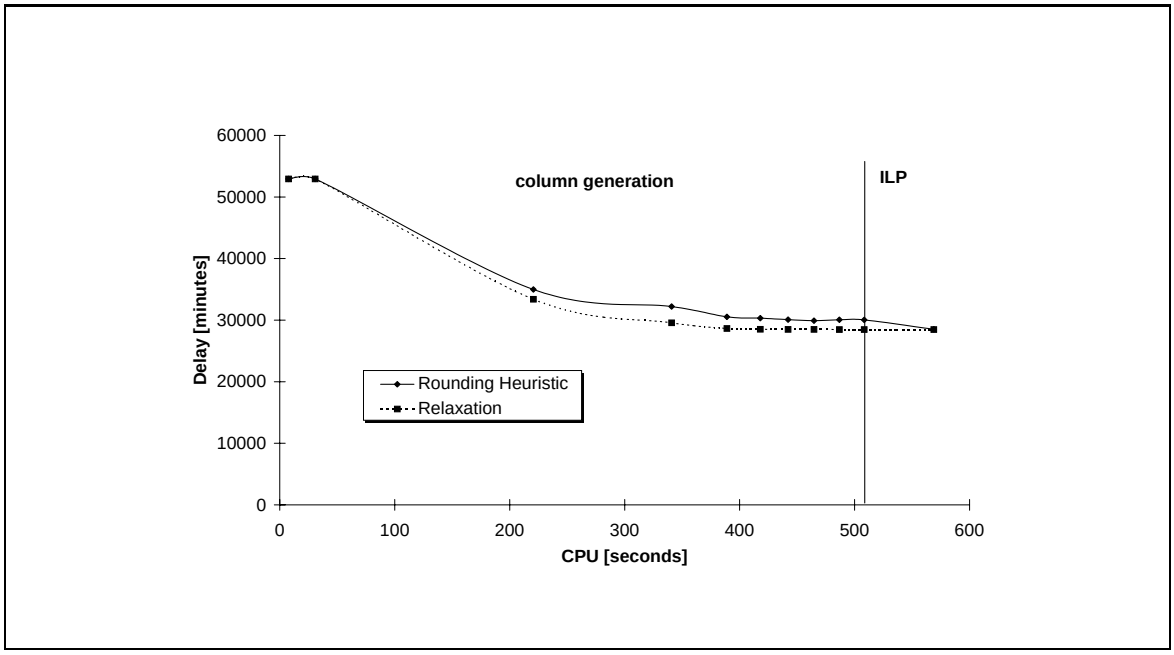

Fig. 4 Results of rounding heuristic and column generation for Thu-2-June-1994.

the rounding heuristic provides very good solutions. Moreover, the results show that most of the reduction of delay in the solution found by the rounding heuristic is achieved during the first two or three iterations which only require approximately half of the computation time.

A very important advantage of combining column generation with the rounding heuristic is that after each iteration of the column generation a feasible solution is available. The results show that after a few iterations the computation can be stopped with as result a rather good solution being available. This is a quite important from a practical point of view. To take modified flight data into account, slot allocation is repeated very often. Currently, Eurocontrol runs the CASA algorithm approximately every 15 minutes. This limits the computation time available for a slot allocation algorithm. Since the rounding heuristic often finds a good solution rather quickly from the fractional solutions provided by the column generation algorithm, the heuristic can be very useful in practice.

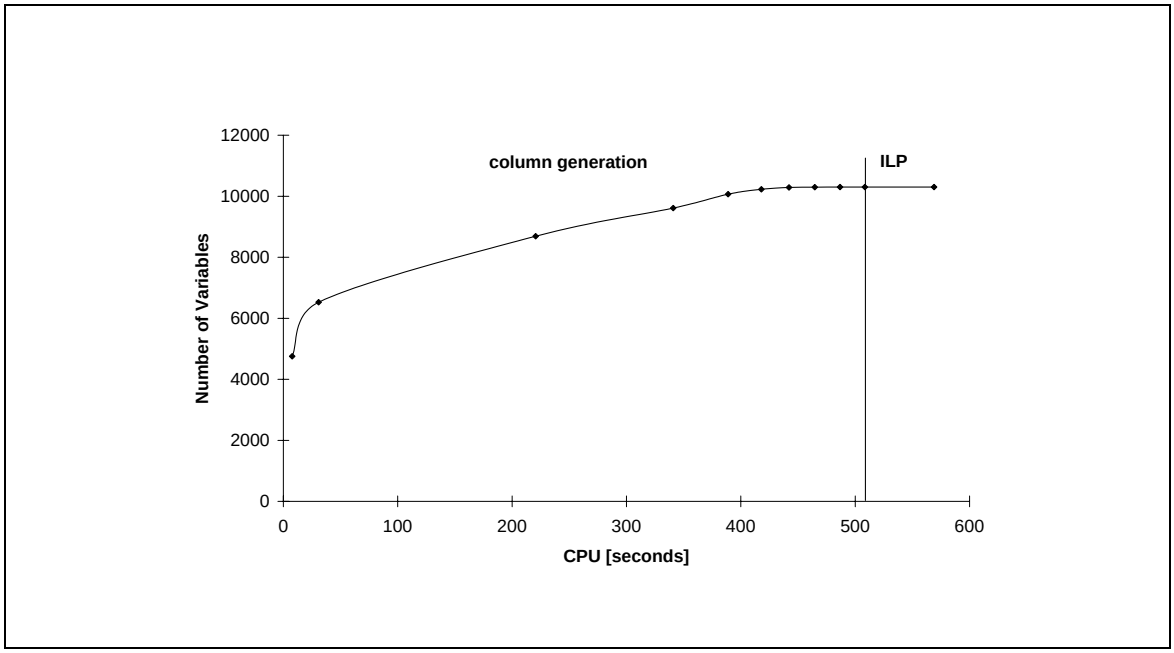

Fig. 5 Number of generated variables for Thu-2-June-1994 and $\delta = 1$.

5 Conclusions

In this paper we presented a heuristic LP-based solution method for a basic version of the European slot allocation problem. The most important step of the method is solving the LP-relaxation by column generation.

We performed computational experiments with real-world examples. They showed that the method provides solutions which are very close to the optimum while requiring only a fraction of the computation time of an exact optimisation algorithm. Moreover, they showed that the LP-relaxation of the integer linear programming model is very strong, which was also observed in [13] and [18]. For the tested instances, the solutions found by a first-come-first-served heuristic, which is the type of heuristic which is used in current practice, can be reduced by 17 % up to 48 % by using our method.

After each iteration of the column generation algorithm we applied a rounding heuristic to derive a feasible solution from the current fractional solution. Our experiments indicate that the rounding heuristic provides rather good solutions quickly, i.e., already after a few iterations of the column generation algorithm. This is very interesting from a practical point of view, since in practice the slot allocation algorithm has to be run quite often to deal with modified flight data.

We conclude that column generation in combination with the rounding heuristic is a very promising approach for solving the European slot allocation problem.

References

1. G. Andreatta and L. Brunetta. *Multi-Airport Ground Holding Problem: A Computational Evaluation of Exact Algorithms*. Technical report, Department of Pure and Applied mathematics, University of Padova, Padova, Italy, 1995.
2. G. Andreatta, L. Brunetta, and G. Guastella. *Multi-Airport Ground Holding Problem: A heuristic Approach Based on Priority Rules*. Technical report, Department of Pure and Applied mathematics, University of Padova, Padova, Italy, 1995.
3. G. Andreatta, A.R. Odoni, and O. Richetta. Models for the Ground-Holding Problem. In L. Bianco and A.R. Odoni, editors, *Computation and Information Processing in Air Traffic Control*. Springer-Verlag, Berlin, 1993.
4. ATAG. *European Traffic Forecasts*. Geneva, August, 1992.
5. C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelbergh, and P.H. Vance. Branch-and-price: Column Generation for Solving Huge Integer Programs. *Operations Research*, to appear.
6. D. Bertsimas, A.R. Odoni, and P.B. Vranas. The Multi-Airport Grounnd-Holding Problem in Air Traffic Control. *Operations Research*, 42(2):249–261, 1994.
7. D. J. Bertsimas and S. Stock. *The Air Traffic Flow Management Problem with En-Route Capacities*. Technical report, Alfred P. Shool for Management, Massachusetts Institute of Technology, 1994.
8. CPLEX Opimization, Inc. *Using the CPLEX Callabale Library*. Manual, 1995.
9. D. Duytschaever. *The Development and Implementation of the EUROCONTROL Central Flow Management Unit (CFMU)*. Technical report, Eurocontrol, 1993.
10. L.A. Hall, A.S. Schulz, D.B. Shmoys, and J. Wein. *Scheduling to Minimize Average Completion Time: Off-line and On-line Approximation Algorithms*. Technical Report Preprint 516/1996, Department of Mathematics, University of Technology, Berlin, Germany, 1996.
11. M.P. Helme. Reducing Air Traffic Delay in a Space-Time Network. In *Proceedings of the 1992 IEEE International Conference on Systems, Man and Cybernatics*, pages 236–242, Chicago, 1992.
12. K.S. Lindsay, E.A. Boyd, and R. Burlingame. Traffic Flow Management Modeling with the Time Assignment Model. *Air Traffic Control Quarterly*, 3(1):255–276, 1993.
13. L. Maugis. *Mathematical Programming for the Air Traffic Management Problem with En-Route Capacities*. Technical Report CENA/R95-022, CENA Orly Sud 205, 94542 Orly Aerograre Cedex, France, June, 1995.
14. L. Navazio and G. Romanin-Jacur. *Multi Connections Multi-Airport Ground Holding Problem: Models and Algorithms*. Technical report, Department of Electronics and Informatics, 1995.

15. W. Philipp and F. Gainche. Air Traffic Flow Management in Europe, pages 64–106. in H. Winter and H.G. Nűsser: *Advanced Technologies for Air Traffic Flow Management*, Springer-Verlag, Berlin, 1994.
16. J.M. Van den Akker. *LP-based Solution Methods for Single-Machine Scheduling Problems*. PhD thesis, Eindhoven University of Technology, 1994.
17. P.B. Vranas. *The Multi-Airport Ground Holding Problem in Air Traffic Control*. PhD thesis, Operations Research Center, MIT, Cambridge, MA, 1992.
18. P.B. Vranas and H.N. Psaraftis. *Work package 2: Evaluation of Tactical En-route Strategies, Workstream 1: Define ATFM Optimisation Criteria, Models and Trial Scenarios*. Technical report, Final report of the NOAA project (European Union), 1996.