Table of Contents

II

Rotation Planning for the Continental Service of a European Airline

M. Elf[1], M. Jünger[1], and V. Kaibel[2]

[1] Universität zu Köln, Institut für Informatik, Pohligstraße 1, 50169 Köln, Germany
[2] Technische Universität Berlin, Fachbereich Mathematik, Straße des 17. Juni 136, 10623 Berlin, Germany

Abstract. We consider a version of the *aircraft rotation problem* where the objective is to minimize delay risks. Given a set of flights to be flown by a subfleet the rotation problem is to find a specific route for each aircraft of the subfleet such that each flight is flown by exactly one aircraft. Additionally, the sequence of flights defining a route must satisfy certain requirements mainly to avoid delays. We present a mathematical model for the problem of minimizing the delay risk according to special requirements of a major airline. An efficient Lagrangian heuristic is proposed that uses subgradient optimization and linear assignments as subproblems. Computational results on real data are given and compared to actual aircraft rotations of that airline.

1 Introduction

The *aircraft rotation problem* we consider in this article is one of the first steps in the planning process of an airline. Usually, two other problems have to be solved before. First, the airline must decide which flights should be offered. This depends, for instance, on the expected numbers of passengers and profits of the flights. Because an airline has different types of aircraft (*subfleets*) each with different characteristics like seating capacities, crew size, and operational costs it must also decide which aircraft type should be used to establish a connection. The result of this *fleet assignment* is a flight schedule for each subfleet which is the input for the rotation planning problem.

A flight schedule for a given subfleet consists of different *legs* (i.e., nonstop flights) each defined by a departure and arrival airport, a date, and departure and arrival times. An example for a leg between Frankfurt and Amsterdam could be

$$(\text{FRA / 15. Aug / 19:20} \longrightarrow \text{AMS / 15. Aug / 20:25}) \ .$$

The *aircraft rotation problem* is to determine *routes*, i.e., sequences of consecutive legs flown by the same aircraft.

In contrast to other problems arising within the planning process of an airline (like the fleet-assignment problem mentioned above or the *crew-scheduling problem*) the aircraft rotation problem has seldomly been a subject

in Combinatorial Optimization. Some models, solution methods, and results for the aircraft rotation problem of major U.S. airlines can be found, e.g, in the papers of Clarke et al. (1997) and Bohland et al. (1998). The main objective of their models is the maximization of the *through value*. Because airline networks in northern America have a distinctive *hub and spoke* structure, passengers often have to change airplanes at the hubs during their trip. Airlines try to make their connections attractive by choosing rotations which allow passengers to travel without changing airplanes too often. The trough value is the revenue the airline expects to gain by additional passengers attracted to a service because of the possibility to stay on the same airplane rather than to change it during a stop-over. Besides the through value maximization these models include maintenance constraints which guarantee regular visits of sufficient lengths at maintenance stations for every aircraft.

The rotation problem we consider is of different structure because the objective is not to optimize through values and maintenance conditions are negligible. This is due to a different handling of maintenance issues and due to a different weighing of the goals "maximizing through value" and "minimizing delay risk" by the our industrial partner *Lufthansa Systems GmbH*.

The rest of this article is organized as follows. In the next parts of this section we define our problem in detail and describe special requirements and properties. Section 2 describes a mathematical model based on a graph formulation. This is used to formulate an algorithm for the solution of the rotation problem in Section 3. Computational results are presented in Section 4.

1.1 The Rotation Problem

The input for our rotation problem is a feasible schedule. This schedule is the result of a fleet assignment and consists of all legs to be flown by a subfleet within a period of about six months. Feasibility means that there exist routes covering all legs with at most as many routes as there are airplanes available. Each leg is flown by exactly one airplane. This rotation planning problem can also be interpreted as the problem of assigning each incoming leg at an airport to one outgoing leg. By defining the successor of each leg routes are determined.

In contrast to U.S. airlines continental services of European airlines show a different structure. There is no hub and spoke network but a small number of airports in the home country and many destinations in other parts of Europe. Figure 1 shows an example of all connections within one day of a large subfleet of a German airline. We can see that the arrival airports, resp. the departure airports, of most connections are located in the home country. That means, most of the connections affect a relatively small number of airports. Later we describe how this can be used to reduce the problem size.

Because usually there are only few stop-overs, the through value is of minor interest. The objective of our optimization is the reduction of *delay risks*. In order to avoid delays we are given several rules for connecting legs.

Fig. 1. Connections of a large subfleet during one day. The size of the airports are drawn in logarithmic scale according to the number of arrivals and departures. Each edge represents at least one connection per day.

These rules had been developed by our industrial partner and represent its criteria for building rotations. Some of the rules forbid connections, others penalize them because of an increasing delay risk. There are two kinds of penalizing rules which we describe below. Each violation of a rule results in a penalty. We are interested in rotations with a minimum sum of penalties which means small delay risks.

One set of rules (*local rules*) are easy to handle. They affect the connection of exactly two legs. Later we will see how to model them with *linear assignment problems*. An example of a local rule is that between two connected legs there must be a certain positive *minimum ground time* between the arrival time of the first and the departure time of the second leg. This time is needed, e.g., for filling-up fuel, cleaning-up, and inspecting the aircraft. It depends on the airport of the connection, the departure airport of the first, and the arrival airport of the second leg. Sufficiently large ground times should furthermore guarantee that airplanes can start at their scheduled departure times even if they arrive late. Under certain conditions the actual ground time is allowed to be slightly less than the minimum ground time, although

this is not appreciated due to an increasing delay risk. The number of such *minimum ground time (MGT)* violations should be kept small.

Some airports in Europe have a very large volume of traffic. They are called *air traffic critical (ATC)* airports. Routes not visiting such airports too often should be preferred because large traffic may cause delay. This is taken into account by another local rule saying that the departure airport and the arrival airport of two connected legs must not be air traffic critical. A violation is called *direct ATC violation*. Overall there are six local rules.

The second class of rules affect the consecutive connection of more than two legs, i.e, subpaths of routes. They are called *non-local rules* and are much more difficult to handle. One rule restricts the number of ATC airports an airplane visits within one day. Another rule demands not to connect a certain number of consecutive legs with only the minimum ground time. The most general rule is the following. For each subpath of a route beginning in the morning a *propagated minimum ground time (PMGT)* is defined which depends on the structure of the subpath and the possibility of delay accumulation. The ground time of the last connection in such a subpath should not exceed the propagated minimum groundtime.

1.2 Special Properties

In our problem we do not need to worry about maintenance requirements because maintenance events are implicitly modeled in the schedule. The given schedule contains *maintenance reserves (operational legs)* which are virtual flights with the same arrival and departure airport like

$$(\text{FRA / 04. Aug / 00:00} \longrightarrow \text{FRA / 04. Aug / 23:59}) \ .$$

This represents an 24 hours maintenance check at Frankfurt. Operational legs are treated like usual legs in the rotation planning process. The strategy of the airline we have cooperated with is to exchange an aircraft that requires a certain maintenance service with an aircraft scheduled for a maintenance reserve when necessary.

The European air traffic has a special structure. There are very few overnight flights. Figure 2 shows the average number of airborne machines for several subfleets of a German airline over the period of one day. We can see that there is only one subfleet with an average of one airborne airplane during the night. For all other subfleets there is a period of about three hours with no flights. It follows that delays cannot accumulate overnight. This means we can decompose the whole problem into optimization for each day.

2 Modeling the Problem

In this section we formulate the rotation planning problem as a graph problem and derive an *integer linear program* (ILP) for which we will develop an algorithm in Section 3.

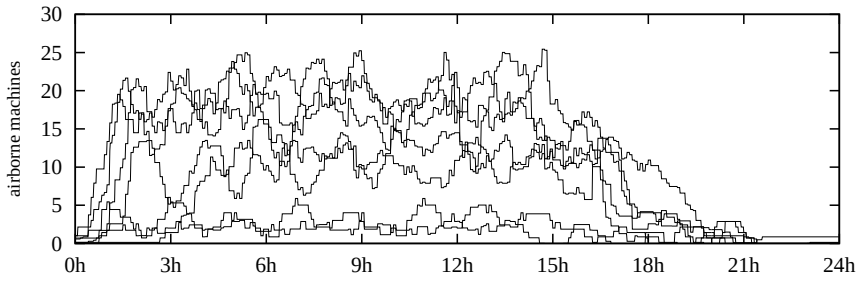

Fig. 2. Average number of airborne machines over the period of one day. Each graph corresponds to a subfleet. Time is given in hours.

2.1 A Graph Problem

Let $n \in \mathbb{N}$ be the number of airports and $\mathcal{L}$ be the set of legs for which arrival or departure time lie within the optimization period (usually one day). For each airport $i \in \{1, \ldots, n\}$ we define a set of *arrival nodes* A_i and *departure nodes* D_i. For each leg arriving at an airport i within the period there is exactly on arrival node $a \in A_i$ and for each leg leaving an airport i within the period there is exactly one departure node $d \in D_i$. Thus, usual legs (i.e., those completely lying within the period) yield two nodes in the model, while (the rare) "overnight-legs" give rise to only one node. Now we define edges representing feasible connections of two legs. Connectable legs correspond to a pair of arrival and departure nodes at the same airport. We define an edge set E_i at each airport i. If it is possible to connect the legs corresponding to the arrival node $a \in A_i$ and the departure node $d \in D_i$ we introduce an edge $e_{ad} \in E_i$. This yields a *bipartite* graph $G_i = (A_i \cup D_i, E_i)$ for each airport $i \in \{1, \ldots, n\}$. Let $G = (A \cup D, E) = \bigcup_{i=1}^{n} G_i$ with $A = \bigcup_{i=1}^{n} A_i$, $D = \bigcup_{i=1}^{n} D_i$, and $E = \bigcup_{i=1}^{n} E_i$ be the collection of these graphs.

Figure 3 shows an example with three airports. Nodes are drawn top down according to the arrival resp. departure times of their legs. Solid edges represent possible connections. Arrival and departure nodes belonging to the same leg are connected with a dotted arc. They are not edges of G but represent the flight defined by their end nodes. Some of the arrival nodes without a corresponding departure node are drawn red. These nodes are called *initial nodes*. They represent the aircraft being located at the respective airport at the start of the optimization period (e.g., those that have spent the night at that airport). The initial nodes mark the beginning of the different routes in the rotation schedule.

In order to find a rotation we have to assign an arriving leg to each departure leg. In the graph model this results in *D-perfect matchings*, i.e., subsets of the edges E such that no two edges in the subset share a common endnode, but every node in D is covered. Conversely, each D-perfect matching

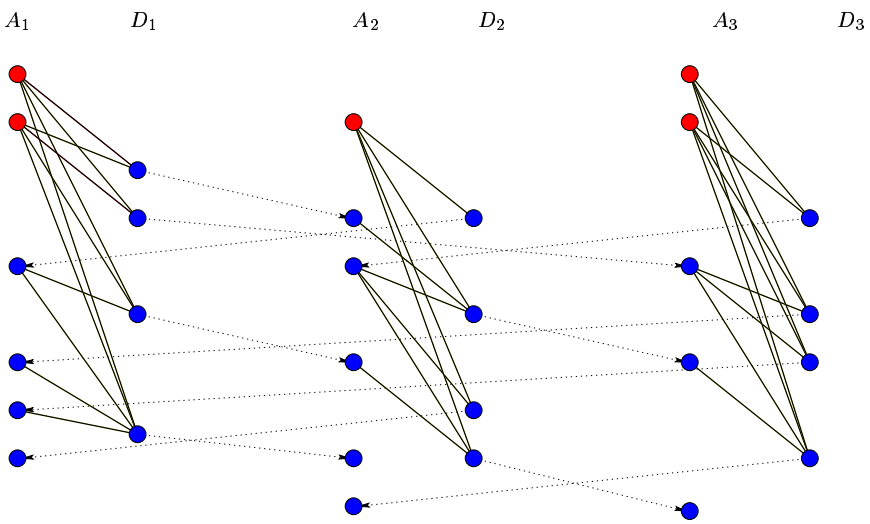

Fig. 3. The graph model with three airports. Solid edges represent possible connections. Dotted edges correspond to flights. Initial nodes are drawn red.

defines a rotation by assigning an arrival leg to each departure leg. Figure 4 shows a D-perfect matching for the graph model of Figure 3. Bold edges are part of the D-perfect matching. The marked path starting from the first arrival node at airport A_1 represents one of five routes in the solution.

As mentioned above, local rules only affect two connected legs. If we weight the edges $(a, d) \in E$ of our graph G by $c_{(a,d)} \in \mathbb{R}^{\geq 0}$ according to the penalty caused by the violation of local rules and neglect non-local rules, the rotation problem can be solved by computing a minimum weighted D-perfect matching of G. This problem can be solved efficiently (see, e.g., Ahuja et al. (1993)). Furthermore, the problem decomposes into D_i-perfect matching problems for each airport. In order to speed up the computation they should be processed separately.

2.2 An ILP-Formulation

Non-local rules affect more than one consecutive connection. They implicitly define certain possible paths in the route of an aircraft that should be avoided. These rules cannot be expressed in the matching formulation. Based on the graph model, we formulate the problem with non-local rules as an integer linear program.

Let $\mathcal{P}$ be the set of paths defined by the non-local rules. Every $P \in \mathcal{P}$ consists of some edges $P = e_1 e_2 \dots e_k$ in E. Let $|P| := k$ be the length of the path P. For every path $P \in \mathcal{P}$ we define a weight $\gamma_P \in \mathbb{R}^{\geq 0}$ according to its penalty. For each edge $(a, d) \in E$ we define a binary variable $x_{(ad)} \in \{0, 1\}$

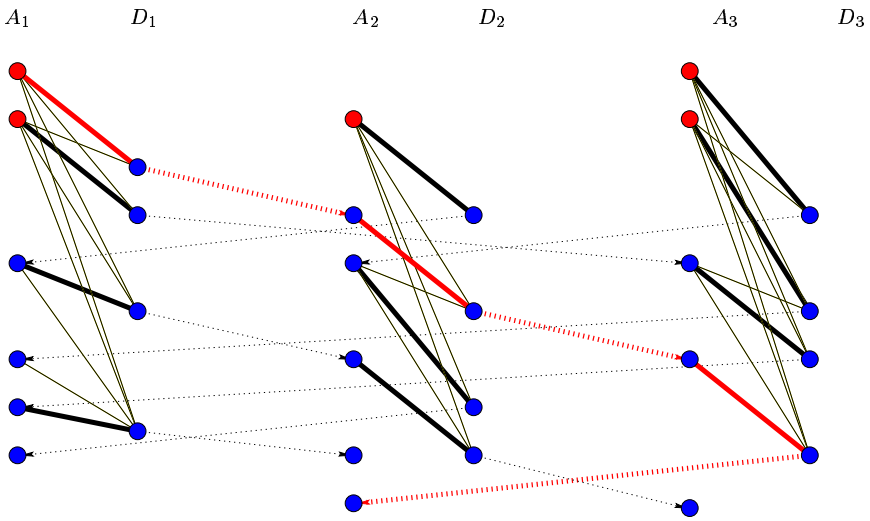

Fig. 4. One possible solution of the rotation problem. Bold edges represent chosen connections. Red edges show a resulting route.

with the meaning that $x_{(a,b)} = 1$ holds if the edge $(a,b) \in E$ belongs to the solution and $x_{(a,b)} = 0$ otherwise. For each path $P \in \mathcal{P}$ we introduce a binary variable $y_P \in \{0,1\}$ with $y_P = 1$ if and only if each edge e in the path $P = e_1 e_2 \dots e_k$ is part of the solution. The optimum solution of the integer linear program

$$Z = \min \sum_{(a,d)\in E} c_{(a,d)} x_{(a,d)} + \sum_{P\in\mathcal{P}} \gamma_P y_P$$

$$\sum_{d\in D_i} x_{(a,d)} \le 1 \qquad (a \in A_i, 1 \le i \le n) \tag{1}$$

$$\sum_{a\in A_i} x_{(a,d)} = 1 \qquad (d \in D_i, 1 \le i \le n) \tag{2}$$

$$\sum_{(a,d)\in P} x_{(a,d)} - y_P \le |P| - 1 \qquad (P \in \mathcal{P}) \tag{3}$$

$$x_{(a,d)} \in \{0,1\} \qquad ((a,d) \in E)$$

$$y_P \in \{0,1\} \qquad (P \in \mathcal{P})$$

is the optimum solution of the rotation problem. The constraints (1) and (2) together with the binary condition for the x-variables guarantee that the x-part of the solution always defines a D-perfect matching in G because (1) implies that each arrival node is covered by at most one edge and (2) implies that each departure node is covered exactly once. The x-part in the objective function sums up the penalties for local rules (matching edges). By setting

$y_P = 1$ for all $P \in \mathcal{P}$ the constraints (3) are always feasible and do not influence the x-part of a solution. Thus, for every rotation there is a feasible solution of the ILP and vice versa. Because the sense of the optimization is to minimize and the objective function coefficients γ_P are non-negative a variable y_P has value zero unless

$$\sum_{(a,d)\in P} x_{(a,d)} = |P| \ ,$$

i.e., unless the x-part of the solution defines a route containing the path P. It follows that the objective function value is the sum of all penalties.

3 Solving the Problem

Using the ILP formulation of the previous section we developed a Lagrangean heuristic which is able to find good solutions in reasonable time. Running time plays an important role since most of the schedule and rotation planning is still an interactive process. Schedules are manually fine-tuned by forbidding existing connections, creating new connections, and by changing legs. Afterwards new rotations are determined by the computer and again modified by a schedule designer. This process is repeated several times.

Another possibility for solving the ILP formulation is a branch & cut approach using the LP-bound of the ILP-formulation. The LP-bound can be computed with a cutting plane procedure. Starting with the linear program containing only the constraint sets (1) and (2), constraints of set (3) are added if they are violated. After that the linear program is reoptimized and the process is iterated until no constraint is violated. We tried this approach but it turned out not to be competitive with the approach we describe in the following section. This has basically two reasons. First, in general this method does not yield feasible solutions for the rotation problem and it is not easy to construct rotations from the fractional LP-solutions. The second reason is that in each iteration new violated constraints must be detected. This *separation problem* is a shortest path problem with difficult side constraints. Its solution is very time consuming.

3.1 Lagrangean Relaxation

Besides the constraints (1) and (2) for the x-part of the solution our ILP-formulation contains a constraint and an additional variable y_P for each subpath P associated with a non-local rule. The flight schedules we are dealing with contain thousands of such subpaths which result in integer linear programs with a very large number of constraints and variables. They are too large for state of the art mixed integer solvers. In order to solve the rotation problem, we apply a Lagrangean relaxation to our ILP-formulation and use subgradient optimization to solve the dual problem. In general, this approach

does not yield optimal solutions. But compared to actually flown rotations we can substantially decrease the number of violations. The lower bound provided by the Lagrangean relaxation is of minor quality.

By relaxing the constraint set (3) for non-local rules with Lagrangean multipliers $\lambda_P \in \mathbb{R}^{\geq 0}$ for each subpath $P \in \mathcal{P}$ we obtain a Lagrangean relaxation

$$
\begin{aligned}
Z_D(\lambda) = \min \sum_{(a,d)\in E} c_{(a,d)} x_{(a,d)} + \sum_{P\in\mathcal{P}} \gamma_P y_P \\
+ \sum_{P\in\mathcal{P}} \lambda_P \left(\sum_{(a,d)\in P} x_{(a,d)} - y_P - |P| + 1 \right) \\
\sum_{d\in D_i} x_{(a,d)} \leq 1 \qquad (a \in A_i, 1 \leq i \leq n) \\
\sum_{a\in A_i} x_{(a,d)} = 1 \qquad (d \in D_i, 1 \leq i \leq n) \\
x_{(a,d)} \in \{0,1\} \qquad ((a,d) \in E) \\
y_P \in \{0,1\} \qquad (P \in \mathcal{P}) \ .
\end{aligned}
$$

The Lagrangean dual is

$$Z_D = \max_{\lambda\in\mathbb{R}^{\geq 0}} Z_D(\lambda)$$

which yields a lower bound on the optimum objective function value of the ILP-formulation. Our Lagrangean relaxation has the "integrality property" which means that there exists alway an integer optimum solution even if the integrality requirements are ignored. Obviously the y-part of any optimum solution is integer and does not influence the x-part. By a classical theorem due to Birkhoff (1946) (see also von Neumann, 1953) the x-part of an optimum solution is also integer since the feasible set is an integer polyhedron. The integrality property implies that the bound obtained by the Lagrangean dual equals the LP-bound of the ILP-formulation (see, e.g., Schrijver (1993)). The independence of the x-part and the y-part also guarantees that the Lagrangean relaxation with a fixed λ can be solved by determining D-perfect matchings in the model graph like in the case without non-local rules. The optimum values of y_P can be obtained by setting

$$y_P = \begin{cases} 0 & \text{if } \gamma_P \geq \lambda_P \\ 1 & \text{if } \gamma_P < \lambda_P \end{cases} \qquad \text{for all } P \in \mathcal{P} \ .$$

3.2 Subgradient Optimization

Because the Lagrangean dual is a piecewise linear convex optimization problem it can be solved by a subgradient method as described in Fischer (1985).

Given an initial value $\lambda^{(0)}$, a sequence $\{\lambda^{(k)}\}$ of Lagrangean multipliers is iteratively generated by the rule

$$\lambda_P^{(k+1)} = \max\left\{\lambda_P^{(k)} + t_k \xi_P^{(k)}, 0\right\} \qquad (P \in \mathcal{P}, k \geq 1) \tag{4}$$

where $\xi^{(k)}$ is a subgradient of the dual function $Z_D(\lambda)$ and t_k is a positive scalar step size. The subgradient of the k-th iteration is obtained as

$$\xi_P^{(k)} = \sum_{(a,d)\in P} x_{(a,d)}^{(k)} - y_P^{(k)} - |P| + 1 \qquad (P \in \mathcal{P}, k \geq 1)$$

with the optimum solution $(x^{(k)}, y^{(k)})$ of the Lagrangean relaxation $Z_D(\lambda^{(k)})$.

Theoretically, the sequence $\{Z_D(\lambda^{(k)})\}$ converges to Z_D if the step length t_k is chosen such that $t_k \to 0$ and $\sum_{i=0}^{k} t_i \to \infty$. In practice it is more suitable to choose the step length according to

$$t_k = \alpha \, \frac{\hat{Z} - Z_D(\lambda^{(k)})}{\|\xi^{(k)}\|^2}$$

with an upper bound $\hat{Z} \geq Z_D$ and a periodically decreasing $0 < \alpha < 2$.

In our application the initial values are $\lambda_P = 0$ for the Lagrangean multipliers and $\alpha = 1.9$. If the lower bound cannot be improved within 5 iterations of the subgradient method α, is divided by 2. In each iteration of the subgradient optimization, the Lagrangean subproblem is solved and yields a feasible solution of the rotation problem, since the x-part of the solution defines D-perfect matchings in the model graph G. The best solution is stored and the upper bound $\hat{Z}$ is set to the corresponding objective function value. If $\hat{Z}$ is updated, α is set to 1.9 again. Because of the large number of Lagrangean multipliers we generate them only if they become positive, i.e influence the relaxation. This happens only if the corresponding subpath is contained in a previously computed D-perfect matching. Multipliers are removed as soon as their value becomes 0. By this strategy, the relaxation can be kept small. In our experiments with real data the number of active multipliers never exceeded 200. We run the subgradient procedure for at most 250 iterations but stop if the length of the search direction becomes too small.

Usually subgradient optimization does not produce sequences of monotonously decreasing lower bounds. Often a phenomenon called "zig-zagging" occurs which has very bad influence on the convergence and the quality of the solution. This can be avoided by using an improved version of the subgradient method proposed by Crowder (1976). In this variant the search strategy (4) is replaced by

$$\lambda_P^{(k+1)} = \max\left\{\lambda_P^{(k)} + t_k d_P^{(k)}, 0\right\} \qquad (P \in \mathcal{P}, k \geq 1)$$

and the search direction

$$d^{(k)} = \xi^{(k)} + \theta d^{(k-1)} \qquad (k \geq 1)$$

is chosen as a linear combination of the current subgradient and the previous direction. In our application a damping factor of $\theta = 0.7$ turned out to be suitable.

3.3 Solving the Subproblems

During the subgradient procedure most of the work is spent in solving the Lagrangean subproblems, i.e, for the solution of the bipartite matching problems in the model graph G. In our software the time for solving the matching problems is about 98% of the overall running time. Thus, these problems should be solved as efficiently as possible. One possibility to gain a faster algorithm is to reduce the problem size. In Section 1.1 we already mentioned the special structure of our flight schedules. There are many airports with a very small number (e.g, ≤ 4) of connections during the day. This often results in bipartite matchings with exactly one solution or in problems where most of the edges cannot be part of any solution. Due to structural reasons, edges that are not part of any solution even occur at airports with larger number of connections. We can prune all these edges from the model graph in a preprocessing step. This is done as follows.

Consider a bipartite graph $G_i = (A_i \cup D_i, E_i)$ of our model and let M be a D_i-perfect matching. An M-alternating path is a path such that each non-matching edge is followed by a matching edge and vice versa. Alternating cycles are defined similarly. Since the symmetric difference of two matchings is a disjoint union of alternating paths and cycles, it follows that an edge belongs to no D_i-perfect matching if it is not contained in any M-alternating cycle and not contained in an alternating path with one end covered by M and one not. If we direct all edges in M from D_i to A_i and all other edges from A_i to D_i, the first kind of edges are edges between two strongly connected components of the directed graph. The second kind of edges are edges which are not part of a directed path starting at a free node in A_i. Both can be computed in linear time.

Another crucial point for keeping the running time as small as possible is the method for solving the bipartite matching problems. Usually this is done by the *Hungarian method* (see Kuhn (1955)), but for our problem primal methods are more suitable. During the subgradient optimization we have to solve a sequence of matching problems at the same graph but with different objective functions. Especially in later iterations of the subgradient method when the step size has become small there are only little changes of edge weights. It follows that the optimum objective function values of the matching problems change only slightly and that the solution of an iteration is near to the optimum of the next iteration.

This is taken into account by primal methods. They are able to perform a "warm start" at a feasible solution, e.g the near optimum solution of the previous iteration of the subgradient method. Often it is possible to compute the optimum matching with only few iterations. In our application we

formulated the bipartite matching problems as *uncapacitated transportation problems* and solved them with a specialized variant of the *network simplex method* (see Ahuja et al. (1993)).

4 Results

We tested our algorithm on three different data sets of real flight schedules. For each of these schedules we performed experiments with six subfleets of different sizes and characteristics. It turned out that the schedules do not differ in terms of the complexity of the rotation problems. Table 1 shows the problem characteristics of all subfleets for a typical day. There are four large subfleets (321, 320, 733, and 735) and three smaller ones (314, 319, and AB6). For each subfleet the table contains a row with the number of edges in the model graph G, the number of chosen edges (connections) in the solution, the number of dualized constraints for non-local rules, the maximum number of non-zero Lagrangean multipliers during the subgradient method, and the time for the Lagrangean heuristic.

Table 1. Typical problem characteristics for all subfleets

	subfleets						
	314	319	320	321	733	735	AB6
edges in G	150	518	3064	1729	2334	1507	373
connections	33	108	207	236	207	129	48
dualized constraints	85	142	1035	1132	1265	800	27
non-zero multipliers	34	57	104	175	130	113	19
time in seconds	0.2	0.7	4.9	3.0	4.1	2.4	0.1

Because our data shows a distinctive weekly periodicity we do not present results of the whole period of about six month but concentrate on a small number of weeks. The presented results are generalizable. Figure 5 shows the ratios of the number of violated rules in solutions found with the Lagrangean heuristic and the number of violations in actually flown routes. Each graph corresponds to one of the subfleets under consideration. Except for the subfleets 314, 319, and AB6 the ratio is between 0.4 and 0.7 which means that the computed routes contain 30%–60% less violations than the flown routes. The number of violations for the subfleets 314 and AB6 is at least 20% better than in the flown routes. The ratio for subfleet AB6 is often less than 0.4. For the subfleet 319 it is sometimes possible to obtain reductions of violations better than 50% but on some days we only get the same number of violations. The time used to compute the solutions is always less than 5 seconds. Table 1

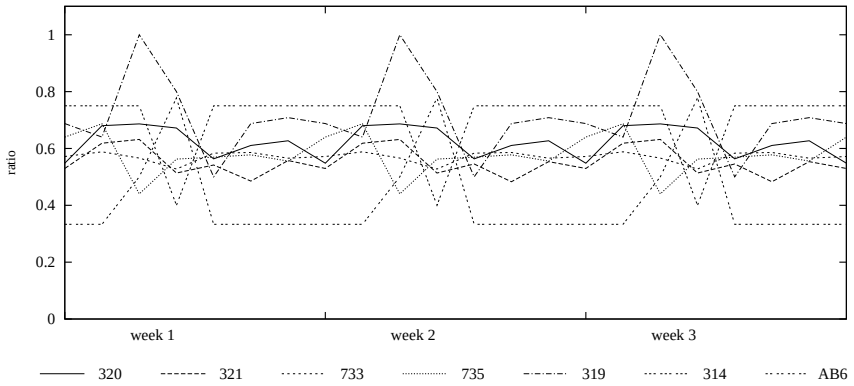

Fig. 5. Ratios of the total number of violated rules for computed and actually flown rotations. Each of the six graphs corresponds to a subfleet.

contains typical running times for all subfleets at a SUN Ultra 4 workstation with 300 MHz.

Until now we only considered the total amount of violations. Table 2 shows a detailed comparison of all violations in computed and flown routes for one week of subfleet 733. Also these results are similar for different weeks and subfleets. We can see that in both cases the number of violations is dominated by the number of violations for the rule "I" which is the PMGT rule. With the Lagrangean heuristic it is possible to reduce these violations drastically. Although we minimize the total amount of violations, the overall decrease of violations is obtained by reducing violations for all but one rule. Only the number of violations of rule "E" (MGT violation) in the computed routes marginally exceeds the violations in the flown routes three times.

The last column of Table 2 contains the lower bound obtained by optimizing the Lagrangean dual. There is a large gap between the best solution and this bound. This is not caused by an insufficient convergence of the subgradient method but the poor LP-bound of our ILP-formulation. Often a Lagrangean heuristic is embedded in an branch and bound scheme to find optimum solutions. Because such a scheme stronly depends on the quality of the bounding procedure this is not reasonable in our case.

In order to make our algorithms and tools for the rotation problem suitable for practical use we bundled them in the software package *RPF*. This package contains a C++ class library, a command line tool, and a graphical user interface. The library can be used to integrate our methods in existing software packages. With the command line tool it is possible to read/write the schedule and routing data in the standard *SSIM* format used by many airlines. With this format we can easily communicate with existing software without changing it. In addition to the computation of rotations, our tool provides a lot of functionality to analyze schedule data and routing informa-

Table 2. Detailed comparison of computed and actual routes for the subfleet 733. The left value in a column is the number of violations for the computed route the right value is the number of violations in the actually flown routes.

	local rules						non-local rules				
Day	A	B	C	D	E	F	G	H	I	$\sum$	bound
1.	0/0	0/0	0/0	0/1	6/4	0/2	0/1	0/9	36/55	42/72	12
2.	0/0	0/0	0/0	0/0	5/6	0/2	0/2	2/9	34/51	41/70	12
3.	0/0	0/0	0/0	0/1	4/5	0/2	0/1	2/9	37/58	43/76	13
4.	0/0	0/0	0/0	0/2	5/4	0/1	0/1	2/8	32/54	40/70	11
5.	0/0	0/0	0/0	0/2	6/4	0/1	0/1	2/7	32/53	40/68	12
6.	11/11	2/3	0/0	0/3	4/7	1/1	1/1	0/2	11/25	30/53	23
7.	7/7	1/2	0/0	1/2	3/6	0/0	0/0	0/2	7/17	19/36	13

tion. With the graphical user interface the schedule and routing information can be displayed in a user-friendly way. Figure 6 shows a screenshot of our interface displaying a solution for subfleet 314. It is also possible to show differences of two rotations, e.g., rotations found by optimizing with different weighting of violations. This can help to analyse different rotations and to find an appropriate weighting strategy.

References

Ahuya, R. K., Magnanti, T. L., and Orlin, J. B., Network Flows: Theory, Algorithms, and Applications, Prentice Hall, Englewood Cliffs, NJ, 1993.

Boland, N. L., Clarke, L. W., Johnson, E. L., Nemhauser, G., and Shenoi, R. G.: Flight String Models for Aircraft, Fleeting and Routing, Transportation Science. Focused Issue on Airline Optimization, Vol. 32, No. 3, pp. 208-220, 1998.

Birkhoff, G.: Tres observationes sobre el algebra lineal. Revista Facultad de Ciencias Exactas, Puras y Applicadas Universidad Nacional de Tucuman, Serie A (Matematicas y Fisica Teoretica) **5**, pp. 147–151, 1946.

Clarke, L., Johnson, E. , Nehmhauser, G., and Zhu, Z.: The Aircraft Rotation Problem. Annals of Operations Research **69**, pp. 33–46, 1997.

Crowder, H.: Computational Improvements for Subgradient Optimization. Symposia Mathematica, Vol XIX., Academic Press, London, 1976.

Fisher, M. L.: An Application Oriented Guide to Lagrangian Relaxation. Interfaces **15**, pp. 10–21, 1985.

Kuhn, H. W.: The Hungarian Method for the Assignment Problem. Naval Research Logistics Quaterly **2**, pp. 83–97, 1955.

Schrijver, A.: Theory of Linear and Integer Programming, John Wiley & Sons, Inc., New York, 1986.

von Neumann, J.: A certain zero-sum two-person game equivalent to the optimal assignment problem. In: Contributions to the Theory of Games, Vol. II, Kuhn, H. W. and Tucker, A. W. (eds.), Princeton University Press, pp. 5–12, 1953.

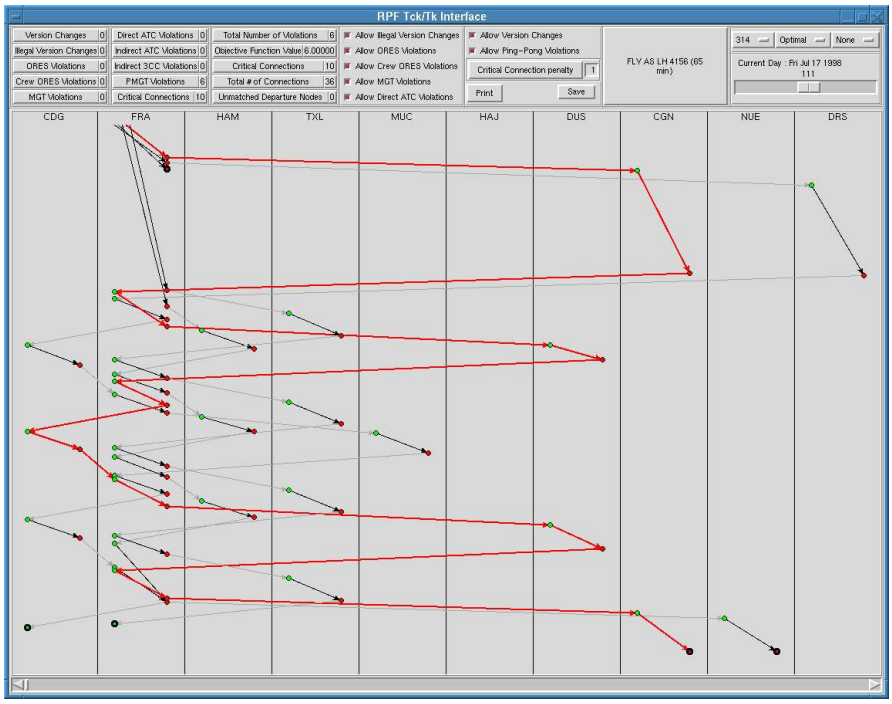

Fig. 6. Graphical user interface showing a solution for subleet 314