Alcom-FT Technical Report Series
ALCOMFT-TR-03-173

Fleet Assignment with Connection Dependent Ground Times

Sven Grothklags*

University of Paderborn, Department of Computer Science
Fürstenallee 11, 33102 Paderborn, Germany
sven@uni-paderborn.de

Abstract. Given a flight schedule, which consists of a set of flights with specified departure and arrival times, a set of aircraft types and a set of restrictions, the airline fleet assignment problem (FAP) is to determine which aircraft type should fly each flight. As the FAP is only one step in a sequence of several optimization problems, important restrictions of later steps should also be considered in the FAP. This paper shows how one type of these restrictions, connection dependent ground times, can be added to the fleet assignment problem and presents three optimization methods that can solve real-world problem instances with more than 6000 legs within minutes.

1 Introduction

For operating an airline, several optimization problems have to be solved. These include network planning, aircraft and crew scheduling. In this paper we address the fleet assignment problem (FAP) in which connection dependent ground times are taken into account. Briefly, in the FAP a flight schedule is given, consisting of a set of flights without stopover, called *legs*, with departure and arrival airport, called *stations*, and departure and arrival time for every aircraft type. An aircraft type, also called *subfleet*, has to be assigned to every leg while maximizing the profit and not exceeding the given number of aircraft per subfleet.

The FAP is only one, though important, optimization problem in the planning process of an airline and these stages have impact on each other. Therefore the main operational constraints of the following stages should also be adhered by the FAP. One operational restriction, that arises in these later stages, is considering minimum turn times of aircraft that depend on the arriving *and* departing leg. We call such turn times *connection dependent ground times*. There are many situations, where connection dependent ground times are needed. A common situation occurs at stations that have distinct terminals for domestic and international flights. If an aircraft, arriving with a domestic flight, wants to proceed with an international flight, it must be towed to a different terminal, which can last more than an hour. But if it proceeds with another domestic flight, it can stay at the terminal and the minimum ground time is much shorter.

Surprisingly, the integration of connection dependent ground times into the FAP has hardly been addressed in literature so far. In this paper we therefore present three methods, one MIP based and two Local Search based approaches, to solve the FAP

* Partially supported by the Future and Emerging Technologies programme of the EU under contract number IST-1999-14186 (ALCOM-FT).

with connection dependent ground times. The MIP approach combines ideas of two well-known MIP models for the FAP to get an improved model that is able to solve real-world problem instances with connection dependent ground times in reasonable time. The Local Search approach is an extension of a heuristic solver that is part of a commercial airline tool of our industrial partner Lufthansa Systems.

The FAP is of considerable importance to airlines and has therefore attracted researchers for many years, theoretically [9, 11, 13] and practically [1, 4, 6, 10, 14, 17]. A common approach for solving the FAP is to model the problem as a mixed integer program (MIP) like in [10]. But there also exist a number of heuristic approaches [4, 8, 15, 16, 18], mainly based on Local Search. Recently, many researchers started to extend the basic FAP by adding additional restrictions [3, 5] or incorporating additional optimization problems [2, 15].

The paper is organized as follows. In Section 2 the FAP with connection dependent ground times is defined as a mathematical model. Section 3 introduces a new MIP model for the FAP and in Section 4 a Local Search based heuristic is presented. In Section 5 a generalized heuristic preprocessing technique is described and Section 6 reports on the experiments made. The paper ends with concluding remarks in Section 7.

2 Problem Definition

The FAP occurs in two flavors during the planning process of an airline. In the midterm strategic planning the FAP is solved for a typical cyclic time period, typically one day or one week. During short-term tactical planning the FAP must be solved for a concrete fully-dated time interval of up to six weeks.

2.1 Input Data and Notations

For simplicity and ease of notation we restrict ourselves to the non-cyclic FAP for the rest of the paper. Nevertheless note that all presented models, algorithms and preprocessing techniques can be easily adopted for cyclic FAPs. An instance of an FAP consists of the following input data:

$\mathcal{F}$	set of available subfleets
N_f	number of aircraft available for subfleet $f \in \mathcal{F}$
$\mathcal{L}$	set of legs (schedule)
$\mathcal{F}_l$	set of subfleets that can be assigned to leg $l \in \mathcal{L}$, $\mathcal{F}_l \subseteq \mathcal{F}$
$p_{l,f}$	profit of leg $l \in \mathcal{L}$ when flown by subfleet $f \in \mathcal{F}_l$
s_l^{dep}	departure station (airport) of leg $l \in \mathcal{L}$
s_l^{arr}	arrival station of leg $l \in \mathcal{L}$
$t_{l,f}^{dep}$	departure time of leg $l \in \mathcal{L}$ when flown by subfleet $f \in \mathcal{F}_l$
$t_{l,f}^{arr}$	arrival time of leg $l \in \mathcal{L}$ when flown by subfleet $f \in \mathcal{F}_l$
$g(k,l,f)$	minimum ground time needed at station s_k^{arr} when an aircraft of subfleet $f \in \mathcal{F}$ operates legs $k \in \mathcal{L}$ and $l \in \mathcal{L}$ successively; $s_k^{arr} = s_l^{dep}$ must hold

The only difference of the FAP with connection dependent ground times compared to the classical FAP is the minimum ground time function $g(k,l,f)$, which depends on the arriving *and* departing leg. In the classical FAP the minimum ground time only depends on the arriving flight k and the subfleet f and can therefore directly be incorporated into the arrival time $t_{k,f}^{arr}$.

2.2 Connection Network

The FAP with connection dependent ground times described here can be defined by a well-known IP based on a flow network, the connection network ([1]). The IP consists of $|\mathcal{F}|$ flow networks, one for each subfleet. The legs are the nodes of the network and arcs represent possible connections between legs. We define the set of valid successors $C_{l,f}$ (predecessors $C_{l,f}^{-1}$) of a leg l when flown by subfleet f:

$$C_{l,f} = \left\{ k \in \mathcal{L} | f \in \mathcal{F}_k, s_l^{arr} = s_k^{dep}, t_{l,f}^{arr} + g(l,k,f) \leq t_{k,f}^{dep} \right\} \cup \{*\}$$

$$C_{l,f}^{-1} = \left\{ k \in \mathcal{L} | f \in \mathcal{F}_k, s_k^{arr} = s_l^{dep}, t_{k,f}^{arr} + g(k,l,f) \leq t_{l,f}^{dep} \right\} \cup \{*\}$$

$C_{l,f}$ contains all possible legs, that an aircraft of subfleet f can proceed with *after* operating leg l. Therefore a leg $k \in C_{l,f}$ must be compatible with subfleet f, must depart at the arrival station of leg l and the ground time between the arrival time of leg l and the departure time of leg k must not be less than the minimum ground time of the legs l and k. The additional element $*$ is used if the aircraft does not fly any further leg until the end of the planning period. $C_{l,f}^{-1}$ contains all possible legs, that an aircraft can fly *before* operating leg l, and $*$ means that l is the first leg of the aircraft.

The connection network IP uses binary variables $x_{k,l,f}$. $x_{k,l,f}$ is one iff legs k and l are flown successively by an aircraft of subfleet f. $x_{*,l,f}$ ($x_{l,*,f}$) is used for "connections" without predecessor (successor), that is, l is the first (last) leg flown by an aircraft of subfleet f during the planning period. With these notations we can write the FAP as follows:

$$\text{maximize} \quad \sum_{k \in \mathcal{L}} \sum_{f \in \mathcal{F}_k} p_{k,f} \left(\sum_{l \in C_{k,f}} x_{k,l,f} \right) \tag{1}$$

$$\text{subject to} \quad \sum_{f \in \mathcal{F}_k} \sum_{l \in C_{k,f}} x_{k,l,f} = 1 \qquad \forall k \in \mathcal{L} \tag{2}$$

$$\sum_{k \in C_{l,f}^{-1}} x_{k,l,f} - \sum_{m \in C_{l,f}} x_{l,m,f} = 0 \qquad \forall l \in \mathcal{L}, f \in \mathcal{F}_l \tag{3}$$

$$\sum_{l \in \mathcal{L}} x_{*,l,f} \leq N_f \qquad \forall f \in \mathcal{F} \tag{4}$$

$$x_{k,l,f} \in \{0,1\} \qquad \forall k \in \mathcal{L}, f \in \mathcal{F}_k, l \in C_{k,f} \tag{5}$$

Condition (2) and (5) ensure that every leg is flown by exactly one subfleet. Equation (3) models the flow conservation and condition (4) limits the number of aircraft of each subfleet by limiting the *total inflow* of each subfleet. The objective function (1) maximizes the total profit.

Unfortunately, the connection network is impractical for real-world instances, because the number of binary variables grows quadratically with the number of legs. Even when using sophisticated preprocessing techniques, only small FAPs (a few hundred legs) can be solved by this model.

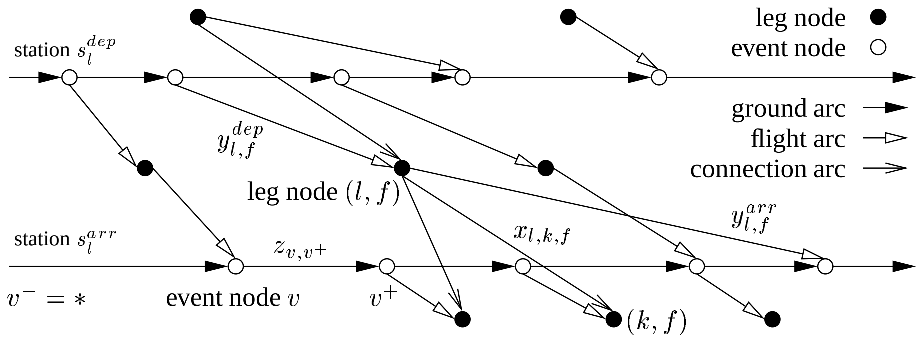

Fig. 1. Part of the flow network of subfleet f used by the new MIP model.

3 New MIP Model

In this section we present a new MIP model for the FAP with connection dependent ground times. The new model is a hybrid of the connection network introduced in Section 2.2 and the so called time space network ([10]). The time space network is probably the most popular method to solve the FAP, but it cannot handle connection dependent ground times.

The nodes in the time space network are the *flight events* on the stations (*possible* arrivals and departures of aircraft). The edges of the network are comprised of flight arcs and ground arcs. A flight arc connects a departure and an arrival event of one leg flown by a specific subfleet. The ground arcs connect two subsequent flight events on a station. A flight event can be identified by a triple (t, s, f), where t is the time of arrival (or departure), s is the station of arrival (or departure) and f is the subfleet the event belongs to.

What makes the time space network unsuitable for dealing with connection dependent ground times is the fact, that it allows *all* connections (k, l) to be established if $t_{k,f}^{arr} \leq t_{l,f}^{dep}$ holds. The main idea for our new model is to increase $t_{k,f}^{arr}$ in such a way that *all* later departing legs form valid connections. By doing so, we may miss some valid successors of leg k, that depart earlier than the adjusted arrival time of leg k, and these connections are handled explicitly like in the connection network.

The adjusted arrival time of a leg l when flown by subfleet f can be computed by:

$$t'_{l,f} = \max\left\{t_{k,f}^{dep} + 1 | f \in \mathcal{F}_k, s_l^{arr} = s_k^{dep}, t_{k,f}^{dep} < t_{l,f}^{arr} + g(l,k,f)\right\} \cup \left\{t_{l,f}^{arr}\right\}$$

The expression simply determines the latest ”compatible” departing leg, which does not form a valid connection with leg l, and sets the adjusted arrival time $t'_{l,f}$ of leg l one time unit above. If no such invalid departing leg exists, the arrival time is left unchanged. Knowing the adjusted arrival time, we can define the set of ”missed” valid successors $C_{l,f}$ (predecessors $C_{l,f}^{-1}$):

$$C_{l,f} = \left\{k \in \mathcal{L} | f \in \mathcal{F}_k, s_l^{arr} = s_k^{dep}, t_{l,f}^{arr} + g(l,k,f) \leq t_{k,f}^{dep}, t_{k,f}^{dep} < t'_{l,f}\right\}$$

$$C^{-1}_{l,f} = \left\{k \in \mathcal{L} | f \in \mathcal{F}_k, s^{arr}_k = s^{dep}_l, t^{arr}_{k,f} + g(k,l,f) \leq t^{dep}_{l,f}, t^{dep}_{l,f} < t'_{k,f}\right\}$$

Like the connection and time space network, our new MIP model consists of $|\mathcal{F}|$ flow networks. Each flow network consists of two different kind of nodes: leg nodes, that correspond to the nodes of the connection network, and event nodes, that correspond to the nodes of the time space network. The edges of the network are comprised of the flight and ground arcs of the time space network and the connection arcs of the connection network. Like in the time space network, ground arcs connect successive event nodes of a station. The flight arcs connect the departure and arrival event of a leg with its leg node. Finally, the connection arcs, that run between leg nodes, establish the "missed" valid connections. See Figure 1 for an example.

We need the following notations to define the new MIP model:

- $\mathcal{V}$ set of all flight events
- $v^{dep}_{l,f} = (t^{dep}_{l,f}, s^{dep}_l, f)$; flight event that corresponds to the departure of leg l when flown by subfleet f
- $v^{arr}_{l,f} = (t'_{l,f}, s^{arr}_l, f)$; flight event that corresponds to the arrival of leg l when flown by subfleet f
- v^+ subsequent flight event of $v \in \mathcal{V}$ on the same station of the same subfleet; $*$ if v is the last flight event
- v^- preceding flight event of $v \in \mathcal{V}$ on the same station of the same subfleet; $*$ if v is the first flight event
- $\mathcal{V}^*_f$ set of flight events of subfleet f that have no predecessor

The new MIP model uses binary variables $x_{k,l,f}$, $y^{dep}_{l,f}$ and $y^{arr}_{l,f}$. $x_{k,l,f}$ directly corresponds to the x-variables in the connection network. $y^{dep}_{l,f}$ ($y^{arr}_{l,f}$) is the flight arc that connects the departure event $v^{dep}_{l,f}$ (arrival event $v^{arr}_{l,f}$) with the corresponding leg node. Finally, we use non-negative variables z_{v,v^+} to represent the flow on the ground arcs. With these notations the new MIP model can be written as follows:

$$\text{maximize} \qquad \sum_{k \in \mathcal{L}} \sum_{f \in \mathcal{F}_k} p_{k,f} \left(y^{arr}_{k,f} + \sum_{l \in C_{k,f}} x_{k,l,f} \right) \tag{6}$$

$$\text{subject to} \qquad \sum_{f \in \mathcal{F}_k} \left(y^{arr}_{k,f} + \sum_{l \in C_{k,f}} x_{k,l,f} \right) = 1 \qquad \forall k \in \mathcal{L} \tag{7}$$

$$y^{dep}_{l,f} + \sum_{k \in C^{-1}_{l,f}} x_{k,l,f} - y^{arr}_{l,f} - \sum_{m \in C_{l,f}} x_{l,m,f} = 0 \qquad \forall l \in \mathcal{L}, f \in \mathcal{F}_l \tag{8}$$

$$\sum_{v^{arr}_{l,f} = v} y^{arr}_{l,f} - \sum_{v^{dep}_{l,f} = v} y^{dep}_{l,f} + z_{v^-,v} - z_{v,v^+} = 0 \qquad \forall v \in \mathcal{V} \tag{9}$$

$$\sum_{v \in \mathcal{V}^*_f} z_{*,v} \leq N_f \qquad \forall f \in \mathcal{F} \tag{10}$$

$$x_{k,l,f} \in \{0,1\} \qquad \forall k \in \mathcal{L}, f \in \mathcal{F}_k, l \in C_{k,f} \tag{11}$$

$$y^{dep}_{l,f}, y^{arr}_{l,f} \in \{0,1\} \qquad \forall l \in \mathcal{L}, f \in \mathcal{F}_l \tag{12}$$

$$z_{v,v^+} \geq 0 \qquad \forall v \in \mathcal{V} \tag{13}$$

$$z_{*,v} \geq 0 \qquad \forall v \in \mathcal{V}^* \tag{14}$$

Whether a leg k is flown by subfleet f or not, can be identified by the outflow of the corresponding leg node, that is $y^{arr}_{k,f} + \sum_{l \in C_{k,f}} x_{k,l,f}$. Therefore the objective function (6) maximizes the total profit. Equation (7) ensures that every leg is operated by exactly one subfleet. Equation (8) ensures the flow conservation condition for leg nodes and equation (9) for event nodes. Condition (10) limits the number of used aircraft per subfleet by limiting the total inflow for each subfleet. Conditions (11)–(14) define the domains of the used variables.

The number of "missed" valid connections (and therefore x-variables) is quite low in practice and there are also many legs, for which $C_{l,f}$ (or $C^{-1}_{l,f}$) is empty. For these legs we can substitute $y^{arr}_{l,f}$ for $y^{dep}_{l,f} + \sum_{k \in C^{-1}_{l,f}} x_{k,l,f}$ ($y^{dep}_{l,f}$ for $y^{arr}_{l,f} + \sum_{m \in C_{l,f}} x_{l,m,f}$) and remove the corresponding leg node equation from the model. So normally, we will end up with a model that is not much larger than a corresponding classical time space network MIP. The model will collapse into a classical time space network if all sets $C_{l,f}$ and $C^{-1}_{l,f}$ are empty, which is the case for FAPs *without* connection dependent ground times.

4 Local Search Heuristics

We enhanced our previously developed Local Search based heuristics ([8]) to be able to deal with connection dependent ground times. The specialized neighborhood can be used in a Hill Climbing or simulated annealing framework to produce high quality solutions in short times. The Simulated Annealing heuristic uses an adaptive cooling schedule described in [12].

Since flight schedules evolve over time, solving the FAP does not have to start from scratch. Changes to a flight schedule are integrated into the old schedule by airline experts, so that for the real-world fleet assignment instances an initial solution is given which can then be used by our Local Search algorithms.

So clearly, the most challenging part for using Local Search to solve the FAP is how to define the neighborhood. Therefore we will restrict to this topic for the rest of the section. We allow two transitions which we call *change* and *swap*. In a nutshell, change and swap look for leg sequences, that do not use more aircraft than the current solution when moved to a different subfleet.

We initially developed our neighborhood for the FAP without considering connection dependent ground times. The following two sections shortly present the basic ideas of the original neighborhood. In Section 4.3 we describe the necessary extensions to be able to deal with connection dependent ground times.

4.1 Waiting Function

Crucial for the efficient computation of our neighborhood transitions is the *waiting function* ([7]). Given a solution to the FAP, the waiting function $W_{s,f}(t)$ counts the number of aircraft of subfleet f available on station s at time t, such that $\forall t : W_{s,f}(t) \geq 0$ and $\exists t' : W_{s,f}(t') = 0$. An island of $W_{s,f}(t)$ is an interval (t_1, t_2), where $W_{s,f}(t)$ is

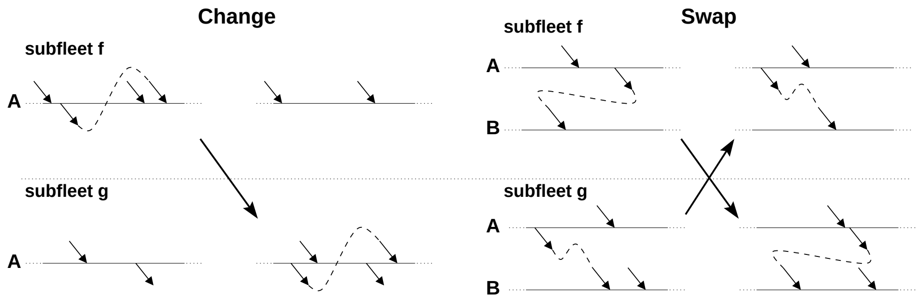

Fig. 2. Neighborhood operations: change and swap. Arcs represent incoming and outgoing legs on a station.

strictly positive $\forall t \in (t_1, t_2)$ and $W_{s,f}(t_1) = 0 = W_{s,f}(t_2)$. At the beginning (end) of the planning interval $W_{s,f}(0)$ ($W_{s,f}(T)$) does not need to be zero. Every flight event (departure or arrival) of a given solution belongs exactly to one of these islands.

The waiting functions can be used in a number of ways. The value $W_{s,f}(0)$ at the beginning of the planning interval tells us the number of aircraft of subfleet f, that must be available at station s at the beginning of the planning interval to be able to operate the schedule. These values are used to determine the number of aircraft needed by a solution. Furthermore it is known, that all connections between arriving and departing legs must lie *within* the islands of the waiting function, and, more importantly, that it is always *possible* to build these connections within the islands. And finally, the waiting function is an important tool to efficiently construct leg sequences for our change and swap transitions, that do not increase the number of aircraft used.

4.2 Change and Swap

We call $(l_0, \dots, l_n)$ a *leg sequence*, if the arrival airport of l_i is the departure airport of leg l_{i+1} for all $i = 0, \dots, n-1$ and all legs $l_0, \dots, l_n$ are assigned to the same subfleet. Leg sequences are generated by a depth first search with limited degree at the search nodes. The possible successors in a leg sequence are not allowed to increase the number of aircraft used, when the leg sequence is changed or swapped to another subfleet. For the computation of a successor, the islands of the waiting function are used. The exact generation process of leg sequences is quite complex. In this paper we can only give a sketch of the two transitions, which define the neighborhood and use leg sequences as building blocks.

The Change. The change transition alters the assigned subfleet for a leg sequence which starts and ends at the same airport A. On the input of a randomly chosen leg l, with f being the subfleet currently assigned to l, and a subfleet g, the change generates a leg sequence S_f that starts with leg l and to which subfleet g can be assigned without using more aircraft than the current solution. For this, during the whole time of S_f subfleet g has to provide an aircraft waiting on the ground on A. Therefore l and the last leg of S_f must lie within one island of $W_{A,g}$.

The Swap. The swap transition exchanges the assigned subfleets f and g among two leg sequences S_f and S_g. To maintain the current distribution of aircraft at the beginning and end of the planning interval, the two leg sequences have to start at the same airport A and end at the same airport B. Let f be the subfleet currently assigned to a randomly chosen leg l. Then, on input of leg l and a subfleet g, the swap transition computes first a leg sequence S'_f starting with l, whose legs are currently assigned to subfleet f and that can also be flown by subfleet g. The depth first search construction of leg sequence S_g also starts at station A and searches for a compatible leg sequence of legs flown by subfleet g, that ends at the same station as a *subsequence* S_f of S'_f. Then S_f and S_g form a valid swap transition.

4.3 Extensions for Connection Dependent Ground Times

Our original neighborhood for the FAP has the same difficulties with connection dependent ground times as the time space network. By using the waiting function, we can assure that a solution of the FAP (without connection dependent ground times) does not exceed the available number of aircraft, but we do not know the concrete connections between arriving and departing legs. We only know, that there *exists* at least one valid set of connections and that these connections must lie within the islands of the waiting function. But this is only true, if *all* legs, that depart later than an arriving leg l within an island, are valid successors of l, and this does not need to hold if we have to consider connection dependent ground times.

Because the waiting function has proven to be a fast and powerful tool to construct transitions for the FAP, we adhered to this concept for the FAP with connection dependent ground times and extended it by additionally storing a *valid* successor for each leg. The generation procedure of leg sequences still uses the waiting function as major tool, but it is modified to ensure, that there still is a valid successor for each leg after a transition.

The computation of successors can be solved independently for each station s and subfleet f. This can be done by calculating a complete matching on the bipartite graph of arriving and departing legs of subfleet f on station s. The nodes of this bipartite graph are the arriving and departing legs and an arriving leg k is connected with a departing leg l iff they can form a valid connection: $t^{arr}_{k,f} + g(k,l,f) \leq t^{dep}_{l,f}$. The waiting function further helps to reduce the size of the bipartite graphs, for which matchings have to be computed. As all connections must lie within islands, it is sufficient to calculate the matchings for each island separately.

The matchings are calculated by a well-known maximum-flow algorithm for the maximum matching problem for bipartite graphs. Note that these matchings only have to be computed from scratch once, namely at the beginning of the Local Search algorithm. Afterwards, we only have to deal with small updates to the islands, when single legs are added or removed. These updates happen obviously when we *switch* to a neighbor of the current solution but they also occur during the construction of leg sequences to *test* whether or not a leg sequence can be moved to another subfleet without destroying the complete matchings. In both cases, we only need to perform very few (one or two) augmentation steps of the maximum-flow algorithm to restore the complete matching or to discover that no complete matching exists anymore.

5 Preprocessing

Hane et al. ([10]) introduced a heuristic preprocessing technique, that can reduce the number of legs of an FAP instance and eliminate some of the ground arcs of the time space network. Here we present a generalization of this preprocessing technique and show how it can be applied to our new MIP model and Local Search heuristics.

Airline experts prefer schedules that use as few aircraft as possible on *small* stations. Spare aircraft should wait at large stations (hubs) where they can be used more easily in the case of schedule disruptions. A lower bound on the number of aircraft needed at a station can be determined by looking at a schedule, when it is operated by *one* artificial subfleet f^*. The subfleet f^* can fly every leg l ”faster” than any real subfleet in $\mathcal{F}$, e.g. $[t^{dep}_{l,f^*}, t^{arr}_{l,f^*}] = \bigcap_{f \in \mathcal{F}_l} [t^{dep}_{l,f}, t^{arr}_{l,f}]$. The value of the waiting function $W_{s,f^*}(0)$ at the beginning of the planning period gives us the minimal number of aircraft needed at station s and, more importantly, the islands of W_{s,f^*} define intervals, in which *all* connections of a real schedule must lie if it wants to use as few aircraft at station s as the artificial subfleet f^*.

Hane et al. suggested two methods, how to reduce the size of a time space network. First of all, two legs k and l should be combined if they are the only legs in an island of W_{s,f^*}. If we want to respect the island structure of W_{s,f^*} for our schedules, k and l must always be flown by one aircraft successively and so they can be joined reducing the number of legs by one. Secondly, we can delete all ground arcs for all subfleets in the time space network that correspond to ground arcs that would run *between* islands of W_{s,f^*}. The waiting function tells us, that the flow of these ground arcs must be zero in order to achieve the minimal number of aircraft, so they can be deleted.

We propose a third reduction method that allows us to join legs that belong to islands with more than two legs. It is a generalization of the first method described above. For each island of W_{s,f^*} we compute a complete matching between arriving and departing legs. Two legs k and l are connected by an arc, if there exists at least one *real* subfleet $f \in \mathcal{F}$ that can fly k and l successively. Then we test each arc of the complete matching, if it is part of *every* possible complete matching by removing it from the bipartite graph and trying to find an alternative complete matching. If an arc is contained in every complete matching, its corresponding legs can be combined. This preprocessing step can be done in $O(n^3)$, where n is the number of flight events of the island.

The idea of joining legs can be applied to any solution method for the FAP. The idea of deleting ground arcs can also be naturally applied to our new MIP model that deals with connection dependent ground times. Besides deleting ground arcs, we can additionally remove all connection arcs that cross islands of W_{s,f^*}.

The preprocessing methods described here are heuristic ones, as they restrict the solution space and therefore may cause a loss of profit. But as they enforce a desired feature on an FAP solution and the loss of profit generally is quite low, it rather adds to the non-monetary quality of the produced assignments.

6 Experimental Evaluation

We investigated the performance (running time and solution quality) of our algorithms on several sets of real-world data. The instances are problems of recent summer and winter schedules from major airlines provided to us by our industrial partner Lufthansa

Table 1. Properties of the problem instances tested and runtime and solution quality of Hill Climbing (HC), Simulated Annealing (SA) and MIP model with full preprocessing.

FAP	instance properties				HC		SA		MIP		
	legs	subfleets	stations	altern.	time	quality	time	quality	total time	root time	quality
A	6287	8	96	2.9	16	97.99%	302	98.81%	752	54	99.90%
B	5243	23	76	1.7	1	99.13%	13	99.84%	4	3	optimal
C	5306	21	76	4.8	5	97.82%	84	99.40%	314	125	99.96%
D	5186	20	71	4.6	5	98.12%	69	99.51%	229	124	99.98%

Systems. Due to confidentiality, only percentage results on the objective values (profit) can be shown. Our experiments include a comparison of the effects of the preprocessing techniques presented in Section 5 and a comparison between the performance of FAPs with and without connection dependent ground times.

Table 1 lists the sizes of the four problem instances used in this evaluation. The column "altern." contains the average number of different subfleets a leg can be assigned to. Instance B is special because its alternative-value is very low. Many of the legs are fixed to one subfleet. This instance emerged from a scenario where only the legs arriving or departing at one specific station should be optimized.

All tests were executed on a PC workstation, Pentium III 933 MHz, with 512 MByte RAM running under RedHat Linux 7.3. We used two randomized heuristics using the neighborhood presented in Section 4, a Hill Climbing (HC) and a Simulated Annealing (SA) algorithm, implemented in C. The HC and SA results show the average value of 5 runs. Our new MIP model was solved by the branch and bound IP-solver of CPLEX 7.5. The root node was computed using the interior-point barrier-method and the sub nodes were processed by the dual simplex algorithm. The IP-solver was used in a heuristic configuration, terminating as soon as the first valid (integral) solution was found.[1] Therefore also for the MIP approach a solution quality is given which corresponds to the IP-gap reported by CPLEX. The solution quality of the Local Search heuristics is calculated relatively to the upper bound generated by the corresponding MIP run. For the CPLEX runs both, the root relaxation time and the total run time, are given. All run times are given in seconds.

Table 1 shows the run times and solution qualities of our algorithms on the benchmark set. All preprocessing techniques described in Section 5 were applied. HC is fastest, but has the worst solution quality. MIP computes near optimal solutions in reasonable time, but normally is the slowest approach. Only on instance B it is able to outperform SA due to the very special structure of instance B mentioned above. Finally, SA is a compromise between speed and quality. In general, it is not easy to decide, if a solution quality of 99% is sufficiently good. On the one hand, we are dealing with huge numbers and 1% can be hundred thousands of Euro. On the other hand, we are only given *estimations* of the profit that can be off by more than 10%, especially during strategic planning.

In Table 2 we present the effects of the preprocessing techniques of Section 5. It contains results of four different preprocessing settings that we applied to (the representative) instance A. Preprocessing setting "no" stands for no preprocessing, "J" joins

[1] The computation of an *optimal* solution can last days despite the small IP-gaps.

Table 2. The influence of preprocessing on our FAP algorithms.

Prepro	profit loss	legs	conn. arcs	ground arcs	flight arcs	leg equ.	rows	columns	MIP time	root time	SA time
no	–	6287	8050	12054	18168	94	14544	34349	5153	179	383
J	0.39%	5102	6143	8545	14660	731	11238	26181	910	77	336
J*	0.56%	4549	6005	7163	13028	802	9693	23355	1001	59	302
GJ*	0.61%	4549	5984	6386	12953	802	9545	22611	752	54	302

Table 3. Comparison between FAPs with and without connection dependent ground times (CDGT).

	Instance A with CDGT					Instance A without CDGT				
Prepro	rows	columns	MIP time	root time	SA time	rows	columns	MIP time	root time	SA time
no	14544	34349	5153	179	383	14155	25930	3199	114	123
J	11238	26181	910	77	336	10100	18915	415	47	103
J*	9693	23355	1001	59	302	8480	16145	336	31	97
GJ*	9545	22611	752	54	302	8313	15403	360	27	95

legs of islands that only consist of two legs, ”J*” additionally joins legs in bigger islands and ”GJ*” uses all techniques described in Section 5. As mentioned earlier, these preprocessing techniques are heuristic ones and therefore the first column lists the loss in profit due to the preprocessing. The following columns show the effect of preprocessing on the LP-size of our MIP model. The number of legs, number of connection, ground and flight arcs, the number of leg node equations and the total size of the LP are given. The last columns contain the run times of our MIP and SA approach. As can be seen, preprocessing is crucial for the MIP approach, as it is able to significantly reduce the run times. SA does not take that much of an advantage from preprocessing.

In Table 3 we compare the run times of our FAP solvers on instances with and without connection dependent ground times to see the impact of adding them to the FAP. We took instance A and replaced its connection dependent ground times $g(k, l, f)$ by classical leg dependent ground times $g(k, f) = \min_{l \in \mathcal{L}} g(k, l, f)$. Thereby we built an instance without connection dependent ground times, similar in size and structure to instance A. Test runs were executed using the four different preprocessing settings from above. Note that our MIP approach transforms into a regular time space network for FAPs without connection dependent ground times. The table shows that the solution times increase only by a factor of 2 to 3 by introducing connection dependent ground times, both for the MIP and SA approach.

7 Conclusions

The main contribution of this paper shows, how an important operational restriction, connection dependent ground times, can be incorporated into the fleet assignment problem and that this extended FAP can be solved for real-world problem instances. We presented and evaluated three different optimization methods, two Local Search based (HC and SA) and one MIP based approach, that have different characteristics concerning run time and solution quality. HC is fast but produces low quality solutions, MIP calculates near optimal solutions but needs longer and SA is a compromise between speed and quality.

We are currently trying to incorporate additional restrictions and extensions into the FAP. Now that we have developed a framework that can explicitly model connections for a limited time horizon after the arrival of a leg, it should be possible to include the through-assignment problem, which is part of the aircraft rotation building optimization step. Furthermore, certain punctuality restrictions, like forbidding successive connections without time buffers, can be integrated.

References

1. J. Abara. Applying integer linear programming to the fleet assignment problem. *Interfaces*, 19(4):20–28, 1989.
2. Cynthia Barnhart, Natashia L. Boland, Lloyd W. Clarke, and Rajesh G. Shenoi. Flight strings models for aircraft fleeting and routing. Technical report, MIT Cambridge, 1997.
3. N. Belanger, G. Desaulniers, F. Soumis, J. Desrosiers, and J. Lavigne. Airline fleet assignment with homogeneity. Technical report, GERAD, Montréal, 2002.
4. M.A. Berge and C.A. Hopperstad. Demand driven dispatch: A method for dynamic aircraft capacity assignment, models and algorithms. *Operations Research*, 41(1):153–168, 1993.
5. L. W. Clarke, C.A. Hane, E.L. Johnson, and G.L. Nemhauser. Maintenance and crew considerations in fleet assignment. Technical report, 1994.
6. G. Desaulniers, J. Desrosiers, Y. Dumas, M.M. Solomon, and F. Soumis. Daily aircraft routing and scheduling. *Management Science*, 43(6):841–855, 1997.
7. I. Gertsbach and Yu. Gurevich. Constructing an optimal fleet for a transportation schedule. *Transportation Science*, 11(1):20–36, 1977.
8. S. Götz, S. Grothklags, G. Kliewer, and S. Tschöke. Solving the weekly fleet assignment problem for large airlines. In *MIC'99*, pages 241 – 246, 1999.
9. Z. Gu, E.L. Johnson, G.L. Nemhauser, and Y. Wang. Some properties of the fleet assignment problem. *Operations Research Letters*, 15:59–71, 1994.
10. C.A. Hane, C. Barnhart, E.L. Johnson, R.E. Marsten, G.L. Nemhauser, and G. Sigismondi. The fleet assignment problem: solving a large-scale integer program. *Mathematical Programming*, 70:211–232, 1995.
11. Tomothy S. Kniker and Cynthia Barnhart. Shortcomings of the conventional fleet assignment model. Technical report, MIT Cambridge, 1998.
12. I.H. Osman. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, 41:421–451, 1993.
13. Ulf-Dietmar Radicke. *Algorithmen für das Fleet Assignment von Flugplänen*. Verlag Shaker, 1994.
14. R.A. Rushmeier and S.A. Kontogiorgis. Advances in the optimization of airline fleet assignment. *Transportation Science*, 31(2):159–169, 1997.
15. D. Sharma, R. K. Ahuja, and J. B. Orlin. Neighborhood search algorithms for the combined through-fleet assignment model. Talk at ISMP'00, 2000.
16. D. Sosnowska. Optimization of a simplified fleet assignment problem with metaheuristics: Simulated annealing and GRASP. In P. M. Pardalos, editor, *Approximation and Complexity in Numerical Optimization*. Kluwer Academic Publisher, 2000.
17. R. Subramanian, R.P. Sheff, J.D. Quillinan, D.S. Wiper, and R.E. Marsten. Coldstart: Fleet assignment at Delta Air Lines. *Interfaces*, 24(1):104–120, 1994.
18. K. T. Talluri. Swapping applications in a daily airline fleet assignment. *Transportation Science*, 30(3):237–248, 1996.