Nationaal Lucht- en Ruimtevaartlaboratorium
National Aerospace Laboratory NLR

NLR-TP-2005-416

Scheduling aircraft landing using airlines' preferences

M.J. Soomer[1,2] and G.J. Franx[1]

1, 2 Vrije Universiteit (Amsterdam, the Netherlands)

2 National Aerospace Laboratory NLR (the Netherlands)

This report has been based on an article submitted for publication in Transportation Science by INFORMS.

Customer:	National Aerospace Laboratory NLR
Working Plan number:	AT.1.D
Owner:	National Aerospace Laboratory NLR
Division:	Air Transport
Distribution:	Unlimited
Classification title:	Unclassified
	September 2005

Approved by:

Author	Reviewer	Managing department
		4-10-05

Summary

Airlines currently have little influence on tactical arrival planning. However, airline operations and cost depend heavily on arrival delays (which occur during airport arrival capacity deficiency). A method will be presented to schedule aircraft landings, taking airlines' cost into account. Airlines supply a cost function for each arriving flight. Using these, the decision on landing times will reflect an economic trade-off for the airlines concerned. The optimisation provides a safe and efficient arrival schedule, which takes into account equity between airlines.

This problem can be formulated as a mixed integer program (MIP). An optimal schedule can be obtained using a MIP-solver. However, computation times become very large. Therefore, also a heuristic using local search (to obtain a landing sequence) and linear programming (to obtain optimal landing times) is introduced. This heuristic can solve large instances within reasonable time.

Experiments using data from a week at a major European hub show considerable cost reductions (compared to the current First-Come-First-Served schedule), especially under low-visibility conditions. Cost savings occur for all airlines, which may be helpful for the acceptance of the method.

Contents

6 Tables

5 Figures

(35 pages in total)

1 Introduction

Over the past few decades, air traffic has experienced a tremendous growth. Significant improvements have already been achieved in enlarging the en-route traffic capacity. As result, the air traffic bottleneck is shifting from the en-route segment to the airports. Foreseen developments, such as free flight [1, 14], will reinforce this. The capacity of airports mainly depends on the runway capacity (maximum number of landings and take-offs). In this paper the landing process is considered.

Air traffic controllers are responsible for landing aircraft in a safe and efficient manner. Currently, they decide on the landing sequence (and runway) of the aircraft. This is done roughly in first-come-first-served order and airlines have no influence on these decisions. However, airline operations and cost depend on these decisions.

When the airport arrival capacity temporarily decreases and the number of approaching flights exceeds this capacity, some of these aircraft must be delayed. Airlines experience different cost for delays of different flights. Depending on the amount of delay, there might be a number of transfer passengers that miss their connecting flight. The crew or aircraft might also be needed to perform a next flight, that now has to be rescheduled. This might propagate delays to departing flights. There are a lot of other possible costs resulting from delays, such as ground crew rescheduling, crew-overtime payments etc.

This paper presents a model and a heuristic to schedule aircraft landings, taking airlines' cost into account. For this approach, airlines have to supply a cost function for each arriving flight. Using these cost functions, the decision on landing times will reflect an economic trade-off for the airlines concerned.

Air traffic controllers are independent service-providers. They provide service to all airlines using an airport and attempt to provide a schedule that will be accepted by all airlines. Because costs of different airlines will vary and we do not wish to favour airlines providing large average cost functions, the cost functions will be "scaled". In section 2 these (scaled) cost functions are discussed in more detail.

A mixed integer programming formulation for the aircraft landing problem, using the sum of these scaled cost functions as objective, is given in section 3. A local-search heuristic to efficiently obtain reasonable schedules for problem instances is described in section 4. In section 5, a small example shows the effect of different (shaped) costs functions. The arrivals during a week at a major European hub were used as input for our method and the results are shown in section 6. These results include an analysis of the costs and fairness of the obtained schedules as well as the performance of the heuristic (both the quality of the solutions and computation times).

1.1 Arrival Process

The description of the arrival process is compiled from the books of Ashford [2] and Nolan and Wright [13].

Runways can be used in single or mixed mode. In mixed mode the runway is used for both landings and take-offs. Single mode means that for a certain time period the runway is used for either arrivals or departures. Our model applies to runways used in single mode (for landings).

Which runway(s) at an airport are used for landings at a certain time depends on wind and visibility conditions and the demanded capacity.

Aircraft cannot land immediately behind one another on the same runway. Aircraft wings generate wake vortices. An aircraft flying too close behind another could lose aerodynamic stability. To avoid this there must be a minimum separation between aircraft, which depends on their weight category (Light, Medium or Heavy). These separations are usually given in nautical miles. Not only the aircraft directly behind an aircraft suffers from the wake vortex, but also aircraft further behind. Therefore, these separation distances should hold for all aircraft flying behind an aircraft.

The landing process depends significantly on visibility. Air Traffic Controllers usually use radar and this requires a minimum separation of 2.5 nautical miles under strict conditions. In low-visibility conditions, the separation distances must be larger. The radar separation is independent of the aircraft types and consequently the actual required separation is the maximum of the wake-vortex and radar separations.

The throughput of a runway (number of landings in a certain time interval) depends on the visibility conditions and the categories and sequence of the arriving aircraft. Our approach is not to determine the sequence that requires the least separation (and maximises the runway capacity), because such a sequence may be unbeneficial with respect to punctuality and airline cost.

For every aircraft there is a finite time interval in which its landing must take place. Aircraft that still have to depart cannot arrive earlier than their departure time plus their shortest possible flight time. Aircraft already on their way to the airport cannot land earlier than their remaining flight time (at their highest speed) and should of course be on the ground before all fuel is consumed.

In our model we will assign landing times to every aircraft within its landing time interval, complying to the applicable separation rules, taking airlines' cost into account.

The assignment of an arriving aircraft to a runway (in use), depends mainly on the origin and route of the aircraft, therefore we will assume a fixed assignment of arriving aircraft to runways. If runways are

independent (the flight paths of the runways are separated sufficiently) those runways can be treated separately. This will be assumed.

An arrival schedule needs to be constantly updated, because circumstances change: e.g. because a scheduled aircraft has a (departure) delay.

A first schedule can be made about 12 hours in advance, for this period a reliable weather forecast is available and some intercontinental flights are already on their way to the airport. Suppose low visibility conditions and capacity shortage are expected. Now the model can be used to assign delays to the departure of scheduled arrivals that have not departed yet. According to Inniss and Ball [11] it is about twice as cheap to delay an aircraft on the ground than in the air. The assignment of ground delays in case of arrival capacity shortage is often referred to as the single airport ground holding problem.

On a shorter time horizon the model can be used to determine a (preferred) landing sequence, which might result in requests to slow down or speed up for some aircraft.

1.2 Previous Work

Beasley et al. [4] give an extensive literature overview on the aircraft landing problem. Here some highlights and more recent papers are listed.

Beasley et al.[4,5 and 6] are considering both single and multiple runway formulations. Their objective is to minimise the (weighted) deviation from the preferred arrival time or the total timespan. Equity among airlines is not (explicitly) considered. They give a mixed integer programming formulation and use several heuristics to solve the problem, such as using a heuristic upperbound for a tree search and restarting the branch and bound algorithm [4] and genetic algorithms [6]. Another article [5] presents a dynamic formulation that includes additional cost for perturbing earlier schedules.

Carr, Erzberger and Neumann analyse the effect of using sequence preferences from airlines for their own aircraft [7] and of exchanging delays [8], using fast-time simulations.

The single airport ground holding problem is considered in various articles. Hoffman and Ball [10] compare different formulations of the problem, minimising the weighted delay. Richetta [16] considers different cost rates for airborne and ground delays. These papers consider the problem on a flight-basis, and equity between airlines is not explicitly considered.

Vossen et. al. consider the ground holding problem by assigning arrival slots (time intervals) to flights. Flights are first assigned to slots based on scheduled arrival times. The airlines are now considered as the owner of these slots. In case of delays (or cancellations) the slots can be redistributed, by

minimising the deviation (per airline) between the actual and the first allocation [3] or by mediated slot trading among airlines [17].

Gilbo [9] considers collaborative allocation of airport arrival and departure capacity. Arrival capacity can be traded for departure capacity, to maximise the airport throughput and minimize delays. Furthermore flight priorities from airlines and other users can be used in the optimisation model.

Rassenti et al. [15] consider a combinatorial auction mechanism for the allocation of airport landing and take-off slots, maximising the total system sur-plus. Loan et al. [12] consider a combination of simultaneous multiple round and package auctions. Here the determining factor is not simply the bid price, but includes other factors related to the performance of the overall system, like safety, capacity and equity.

2 Airline Preferences and Equity

As mentioned in the previous section, the cost incurred by an airline for an arriving flight depends on its landing time. In case of an (air) delay, the fuel costs will be larger and some transfer passengers might miss their connecting flights and have to be compensated. The crew or aircraft might be needed for a next flight, which has to be rescheduled, resulting in larger cost.

Therefore, it seems natural to incorporate airline preferences into an arrival schedule, based on these costs. Since the number of transfer passengers will vary over different flights and days, every single flight is allowed to have its own cost function. Each function relates the arrival time of the flight to cost.

In general it is safe to assume this cost function is convex. Between the earliest possible landing time and the estimated landing time, the costs will probably decrease (because fuel is saved by flying more efficiently). After the scheduled landing times cost will increase, because of delays. Later in time more passengers will miss their transfers or crew will work overtime so the increase will be even larger. This kind of functions will be (or can be approximated by) convex piecewise linear functions, which will be convenient to use in a linear or mixed integer programming formulation, as in section 3. An example of such a function is depicted in figure 1.

In our method convex piecewise linear cost functions are required. Even if the cost-structure of the airline does not exactly fit this form, airlines will still be able to adequately represent their cost, by choosing any number of breakpoints, their locations and the exact slopes of the line pieces. The breakpoints should represent times where cost will make a big step, or after which cost will increase much faster than before. These breakpoints will behave as thresholds when the cost functions are used as objective in a mathematical programming formulation (as we will do). An additional advantage of all cost functions having the same properties is, that it will be easier to compare them.

As mentioned in the previous section, air traffic controllers are independent service-providers. They attempt to provide a schedule that will be accepted by all airlines using an airport. In our model aircraft are scheduled without considering to which airline they belong. However, the considered costs can vary a lot between different airlines. Now, minimising the sum of all cost functions, will favour airlines that define relatively large cost for their flight delays. At first sight this seems reasonable, since this will lead to minimal total cost. However, we have to bear in mind that all airlines are allowed to define their own cost functions. Therefore, they will be able to obtain higher priorities for their flights, by (falsely) representing very large cost for delays. This leaves room for manipulation. This troublesome aspect can be overcome by rescaling all cost functions, such that the average assigned cost per time unit are equal. However, the cost-ratio between flights of a single airline should be preserved, to reflect

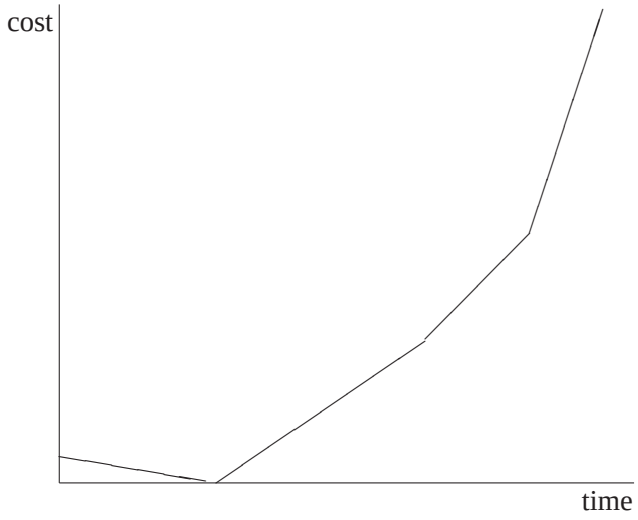

Fig. 1 Example of a convex piecewise linear cost function

the economic trade-off for this airline. Therefore, the same scaling factor will apply to all flights of the same airline.

Let us make this more precise. Consider an airline j with N_j arriving flights, with convex piecewise linear cost functions $\kappa_1(t), \ldots, \kappa_{N_j}(t)$. Let E_i and L_i be the earliest and latest possible landing times of aircraft i, respectively. So, $\kappa_i(t)$ is defined on the interval $[E_i, L_i]$.

To obtain equity, these cost functions will be scaled to new cost functions $f_i(t) = \alpha_j \kappa_i(t)$ ($i = 1, \ldots, N_j$). The scaling factors α_j are determined per airline. This ensures that the ratio between costs of their own flights are preserved in the scaled objective functions. α_j is defined such that:

$$\frac{1}{N_j} \sum_{i=1}^{N_j} \frac{\int_{E_i}^{L_i} \alpha_j \kappa_i(t) dt}{(L_i - E_i)^p} = 1.$$

So,

$$\alpha_j = N_j \left(\sum_{i=1}^{N_j} \frac{\int_{E_i}^{L_i} \kappa_i(t) dt}{(L_i - E_i)^p} \right)^{-1},$$

where p is a parameter to minimise the effect of differences in the length of the landing intervals. It is preferable to choose $p \geq 2$.

Let us explain this. Consider two airlines with only one flight and identical cost functions $\kappa_1(t) = \kappa_2(t) = t$. The possible landing interval of flight 1 is $[0, T_1]$ and for flight 2: $[0, T_2]$, with $T_2 > T_1$ Now

$$\begin{aligned} \alpha_1 &= 2T_1^{(p-2)} \\ \alpha_2 &= 2T_2^{(p-2)} \end{aligned}$$

and the scaled objective functions are:

$$\begin{aligned} f_1(t) &= 2T_1^{(p-2)}t \\ f_2(t) &= 2T_2^{(p-2)}t \end{aligned}$$

If $p < 2$ then $f_1(t) \geq f_2(t)$ for $0 \leq t \leq T_1$. This cannot be considered fair: Flight 2 has a larger interval and therefore more possible landing times. If it would be impossible to land both aircraft before time T_1, aircraft 2 is able to land between time T_1 and T_2, while aircraft 1 is not. When choosing $p < 2$ it is always cheaper to land aircraft 1 before aircraft 2 (even if both aircraft can be scheduled before time T_1) and airline 2 costs ($\kappa_2(t_2^*)$) will be larger than airline 1 costs ($\kappa_1(t_1^*)$). So when $p < 2$, airline 1 is better off, while airline 2 provides more flexibility.

If $p = 2$ then $f_1(t) = f_2(t)$ for $0 \leq t \leq T_1$, which can be considered fair. Another choice is to reward aircraft 2 for providing more flexibility (by a larger interval) by choosing $p > 2$.

The above shows that the length of the interval has influence on the scaled cost function. Basically intervals are based on practical constraints, especially when aircraft are already performing their flight. However, when an aircraft has not departed yet, the latest possible arrival time cannot be determined from operational constraints (e.g. amount of fuel left). In this case it is preferable to use a standard interval length (e.g. 3 hours). The earliest possible arrival time for this kind of flights, follows from the scheduled or expected departure time plus the shortest possible flight time. The latest possible landing time now is the earliest possible landing time plus the standard interval length. This approach minimises differences between flights (and thus airlines) caused by different interval lengths.

At most airports, there are peak periods during the day. If the α_j's were determined for a day or longer, an airline with flights scheduled in peak and non-peak periods has an advantage over an airline with flights only scheduled in peak periods. The first airline could assign large cost to the aircraft in the peak periods, compared to their other flights. If the other airline also assigns large cost for their flights (all in the peak period), their α_j will be low compared to the first airline. Delays are much more likely to happen in peak periods, because in these periods the demand is close to (or even temporarily exceeds) capacity. In these periods the first airline will receive less delay because the scaled costs of delaying flights from airline 2 are lower. Outside peak periods there is only a very small chance on delays. So, the aircraft of the first airline, scheduled in these periods, are not very likely to be delayed, because it is probably not necessary to delay any flight at all. More fairness can be achieved by determining new scaling factors for separate time periods (such as peak and non-peak periods).

This approach also offers the possibilities of trading of preference levels between airlines. Consider airlines j and k, having scaling factors α_j and α_k, respectively. Now airline j could pay airline k to receive a scaling factor $1.1\alpha_j$, and the new scaling factor of airline k would be $0.9\alpha_k$.

3 MIP Formulation

In this section a Mixed Integer Programming (MIP) formulation of the model is given. The basic notation and constraints are similar to those of Beasley et al. [4]. The objective function is as described in the previous section.

3.1 Basic Notation and Constraints

Let

N :	Number of arriving flights to schedule	
E_i :	Earliest possible landing time for flight i	$i = 1, \ldots, N$
L_i :	Latest possible landing time for flight i	$i = 1, \ldots, N$
S_{ij} :	Required separation time when flight i lands before flight j	$i, j = 1, \ldots, N; i \neq j$

and decision variables

t_i : landing time for flight i $\quad i = 1, \ldots, N$

$$\delta_{ij} = \begin{cases} 1 & \text{if flight } i \text{ lands earlier than flight } j \\ 0 & \text{otherwise} \end{cases} \qquad i, j = 1, \ldots N; i \neq j$$

We will now introduce the basic constraints, ensuring a feasible and safe schedule. A more extensive description of those can be found in Beasley et al. [4].

$$E_i \leq t_i \leq L_i \qquad i = 1, \ldots, N \tag{1}$$

This defines the possible landing time. Considering pairs of flights (i, j):

$$\delta_{ij} + \delta_{ji} = 1 \qquad i = 1, \ldots, N-1; j > i \tag{2}$$

This set of constraints states that either flight i must land before flight j ($\delta_{ij} = 1$) or flight j must land before flight i ($\delta_{ji} = 1$).

Using the separation times and (overlap of) possible landing time intervals we can define three sets of pairs of flights:

U : the set of pairs (i, j) of flights for which it is uncertain whether flight i lands before flight j

V : the set of pairs (i, j) of flights for which flight i definitely lands before flight j, but for which the separation is not automatically satisfied

W : the set of pairs (i, j) of flights for which aircraft i definitely lands before flight j, and the separation is automatically satisfied

More formally:

$$\begin{aligned} U &= \{(i,j)|E_j \leq E_i \leq L_j \text{ or } E_j \leq L_i \leq L_j \text{ or } E_i \leq E_j \leq L_i \text{ or } E_i \leq L_j \leq L_i; i,j = 1,\ldots,N; i \neq j;\} \\ V &= \{(i,j)|L_i < E_j \text{ and } L_i + S_{ij} > E_j; i,j = 1,\ldots,N; i \neq j\} \\ W &= \{(i,j)|L_i < E_j \text{ and } L_i + S_{ij} \leq E_j; i,j = 1,\ldots,N; i \neq j\} \end{aligned}$$

For elements in the set V and W the order of the flights is known. So trivially the following constraints hold:

$$\delta_{ij} = 1 \qquad \forall (i,j) \in W \cup V \tag{3}$$

Note that for flights in set V we still need to ensure the proper separation:

$$t_j \geq t_i + S_{ij} \qquad \forall (i,j) \in V \tag{4}$$

Finally we also need to ensure separation if the order of the flights is not known (pairs of flights in set U):

$$t_j \geq t_i + S_{ij}\delta_{ij} - (L_i - E_j)\delta_{ji} \qquad \forall (i,j) \in U \tag{5}$$

If $\delta_{ij} = 1$ (flight i lands before flight j) then it states: $t_j \geq t_i + S_{ij}$, which is the actual separation constraint. If $\delta_{ij} = 0$ (flight j lands before flight i) then it states: $t_j \geq t_i - (L_i - E_j)$. Since $t_i - L_i \leq 0$ and $t_j \geq E_j$ this is always satisfied. Note that the separation in this case is ensured by the constraint for the pair (j, i).

3.2 Objective and Airline Preferences

As explained in section 2 each airline provides a convex piecewise linear cost function for each of its flights, that is then scaled to obtain equity among airlines. The scaled convex piecewise linear function $f_i(x)$ for flight i can be written as a set of linear functions on a number of (disjunct) intervals:

$$f_i(x) = \begin{cases} A_{i0}x + B_{i0} & 0 \leq x \leq X_{i1} \\ A_{i1}x + B_{i1} & X_{i1} \leq x \leq X_{i2} \\ \vdots & \vdots \\ A_{iK_i}x + B_{iK_i} & X_{iK_i} \leq x \end{cases}$$

where

K_i : Number of breakpoints of $f_i(x)$ $\qquad i = 1, \ldots, N$

Now:

$$f_i(x) = \max_{k=0,\ldots,K_i} \{A_{ik}x + B_{ik}\}$$

This can be seen using that $f_i(x)$ is continuous, meaning

$$f_i(X_{i,k}) = A_{i,k-1}X_{i,k} + B_{i,k-1} = A_{i,k}X_{i,k} + B_{i,k} \qquad k = 1, \ldots, K_i$$

and $f_i(x)$ is convex, meaning

$$A_{i0} < A_{i1} < \ldots < A_{iK_i}.$$

The objective of the MIP will be to minimise the sum of these cost functions. However, these function are not linear in the current decision variables t_i, and therefore new decision variables c_i are introduced:

c_i : cost for landing flight i $\quad i = 1, \ldots, N$

The c_i will represent the cost function $f_i(t_i)$. To ensure this the following constraints are introduced:

$$c_i \geq A_{ik}t_i + B_{ik} \qquad i = 1, \ldots, N; k = 0, \ldots, K_i \tag{6}$$

These ensure for flight i that $c_i \geq \max_{k=0,\ldots,K_i}\{A_{ik}t_i + B_{ik}\} = f_i(t_i)$.

Next we will introduce our objective function:

$$z = \min \sum_{i=1}^{N} c_i \tag{7}$$

Equations 6 and 7 ensure that $c_i = f_i(t_i)$:

Suppose $t_1^*, t_2^*, \ldots, t_n^*, c_1^*, c_2^*, \ldots, c_n^*$, is an optimal solution of our MIP, where: $c_i^* > f_i(t_i^*)$ for some i. Let $c_i' = f_i(t_i^*)$. Replacing c_i^* with c_i' will decrease the objective by $c_i^* - f_i(t_i^*)$ without violating any of the constraints 6 ($c_i' = f_i(t_i^*) = \max_{k'=0,\ldots,K_i}\{A_{ik'}t_i^* + B_{ik'}\} \geq \{A_{ik}t_i^* + B_{ik}\}$ for $k = 0, \ldots, K_i$). So $c_i^* > f_i(t_i^*)$ cannot be optimal (and $c_i' = f_i(t_i^*)$ is).

The use of non-convex (linear) cost functions is possible, but will introduce additional decision variables to the model.

4 Local Search

Solving the MIP-formulation will provide an optimal solution, but it can be shown that the decision problem is NP-complete. This can be done by reduction from a jobshop scheduling problem with sequence dependent-setup times and zero processing times (or zero setup times and sequence dependent processing times) (see [4]). This means no efficient (polynomial) solution algorithm is known.

Therefore, we want an algorithm that provides reasonable solutions very fast. To achieve this, the following observation is used: If the landing sequence of the flights is given, the MIP-formulation becomes a LP formulation (since the values of all the binary variables are known). The solution of this LP provides the optimal landing times, given this sequence.

The LP-formulation given a sequence, consists of constraints 1, 4 and 6 and objective 7. Assuming $\max_i K_i < N$ this LP-formulation contains $2N$ variables and at most $2N^2 + N$ constraints. It is known that LP-formulations of this size can be solved quite efficiently for quite large N.

Now the idea is to use local search, to improve the sequence of the flights repeatedly. Since successive sequences will be similar, the corresponding LP formulations will also be. LP solvers are able to solve such a formulation very efficient by using the solution of the previous LP.

The general local search algorithm is given below.

LOCAL SEARCH()
1 $S =$ initial feasible solution
2 **while** there is a neighbour of S of better quality
3 **do** $S =$ neighbour of S of better quality

4.1 Finding a feasible initial solution

Let π be a sequence of the N flights, and $\pi(i)$ the flight at position i in the sequence π. Now let p_i be the preferred landing time of flight i:

$$p_i = \arg \min_{E_i \leq t_i \leq L_i} f_i(t_i).$$

The initial sequence π is obtained by sorting the flights in order of non-decreasing p_i. In case of ties the flights (with equal p_i's) will be ordered in non-decreasing order of E_i, ties broken arbitrarily. So:

$$p_{\pi(1)} \leq p_{\pi(2)} \leq \cdots \leq p_{\pi(N)}$$

and if $p_{\pi(i)} = p_{\pi(i+1)}$ then $E_{\pi(i)} \leq E_{\pi(i+1)}$.

Note that this sequence already reflects the preferences of the airlines by sorting the flights by their preferred landing times. If these preferred landing times are equal or near the expected arrival times, this sequence will be similar to the current first-come-first-served schedule. Therefore, this initial sequence, together with its corresponding optimal landing times (obtained by solving the LP, if feasible), will already provide a reasonable schedule.

If this provides a feasible solution, the local search can continue. If there are no feasible landing times for this sequence (LP is infeasible), the sequence will be changed (repeatedly) in order to obtain feasibility.

This can be summarised as follows:

FIND INITIAL SOLUTION()
1 $\pi =$ sequence of sorted flights
2 **while** $LP(\pi)$is not feasible
3 **do** $\pi =$ sequence which is more likely to be feasible than π

Note that infeasibility of the LP means that, using this sequence, there is no schedule where all flights land in their possible landing interval (constraints 1) and all separation regulations are met (constraints 4). To obtain a schedule which is more likely to be feasible, the sequence of flights can be changed such that flights are able to land before their latest possible landing time and such that the total separation time will be less. To achieve the former, a new sequence will be obtained by looking at the latest possible landing times for every pair of adjacent flights in the old sequence in the following way:

Swap flight $\pi(i^*)$ and $\pi(i^* + 1)$ where

$$i^* = \arg \max_{i=1,\ldots,N-1} \{L_{\pi(i)} - L_{\pi(i+1)} : L_{\pi(i)} - L_{\pi(i+1)} \geq 0\}.$$

Then the LP is being solved for this sequence. This can be done efficiently by using the solution from the previous LP. When the LP is still infeasible this procedure is repeated, until a feasible solution is found or there exist no flight i for which $L_{\pi(i)} - L_{\pi(i+1)} \geq 0$.

In the latter case feasibility can perhaps still be obtained by a sequence that requires less total separation time.

Swap flight $\pi(i^*)$ and $\pi(i^*+1)$ where

$$i^* = \arg\max_{i=1,\ldots,N-1} \{ \; S_{\pi(i-1),\pi(i)} + S_{\pi(i),\pi(i+1)} + S_{\pi(i+1),\pi(i+2)} - S_{\pi(i-1),\pi(i+1)} - S_{\pi(i+1),\pi(i)} - S_{\pi(i),\pi(i+2)} : \\ S_{\pi(i-1),\pi(i)} + S_{\pi(i),\pi(i+1)} + S_{\pi(i+1),\pi(i+2)} - S_{\pi(i-1),\pi(i+1)} - S_{\pi(i+1),\pi(i)} - S_{\pi(i),\pi(i+2)} \geq 0\},$$

with $S_{\pi(0),\pi(1)} := S_{\pi(N),\pi(N+1)} := 0$.

If exhaustive application of this procedure does not give a feasible solution, it is likely there exists no feasible schedule.

4.2 Neighbourhoods

After finding a feasible (initial) solution we have to find a better solution in the neighbourhood of this solution. We will define two types of neighbourhoods: a swap and a shift neighbourhood.

4.2.1 Swap Neighbourhood

Let π be the (feasible) sequence of the flights in the previous iteration. The swap neighbourhood of π consists of all sequences that are similar to π except that two flights have swapped positions. This neighbourhood contains at most $N(N-1)/2$ sequences.

Note that not all swaps are feasible: If possible landing time intervals of two flights do not overlap it will not be possible to swap those two flights. However, in other situations the new sequence can be adjusted slightly to make it feasible.

Let us make this more precise. Consider swapping the flights at position i and j ($i < j$) in π as depicted in figure 2. The rectangles depict the possible landing intervals of the flights. So, $L_{\pi(i)} > E_{\pi(j)}$, meaning that flight $\pi(j)$ can land before flight $\pi(i)$. Suppose the flight at position k ($i < k < j$), cannot land earlier than flight $\pi(i)$ ($E_{\pi(k)} > L_{\pi(i)}$) but can land earlier and later than flight $\pi(j)$ ($E_{\pi(j)} \leq E_{\pi(k)} \leq L_{\pi(j)}$). So swapping the flights at positions i and j would cause infeasibility because flight $\pi(k)$ would be positioned before flight $\pi(i)$.

However, this swap can be made feasible by moving flight $\pi(i)$ to position $j-1$, moving flight $\pi(k)$ to position j, moving flights $\pi(k+1), \pi(k+2), \ldots, \pi(j-1)$ to positions $k, k+1, \ldots, j-2$ and of course flight $\pi(j)$ to position i. This procedure can be performed repeatedly for all flights between positions i and j which cannot land before flight $\pi(i)$ and a similar procedure can be performed repeatedly for all flights between positions i and j which cannot land after flight $\pi(j)$. The swap neighbourhood includes swaps, made feasible, using these procedures. This way the search space is explored more thoroughly than when simply disregarding swaps that lead to infeasibility.

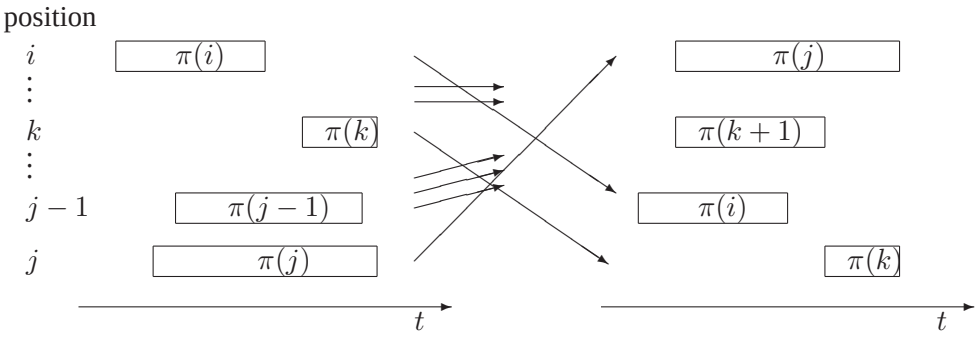

Fig. 2 Swap neighbourhood

4.2.2 Shift Neighbourhood

Another neighbourhood from a sequence π can be obtained by removing one flight and inserting it at another position. This neighbourhood contains at most $N(N-1)$ sequences.

Similar as with the swap-neighbourhood we extend this by a procedure to ensure feasibility for some shifts.

Consider the same sequence of flights as in the example in the previous subsection. Now the shift is to remove the flight at position i in π and insert it at position $j > i$ (see figure 3). Normally, flights $\pi(i+1), \ldots \pi(j)$ will be moved to positions $i, \ldots, j-1$, but since flight $\pi(k)$ cannot land before flight $\pi(i)$, we can only move flights $\pi(i+1), \ldots \pi(k-1)$ to positions $i, \ldots, k-2$. However, we can make this shift feasible by moving flight $\pi(k)$ to position j and flight $\pi(i)$ to position $j-1$. Further flights $\pi(k+1), \ldots, \pi(j)$ are moved to positions $k-1, \ldots, j-2$.

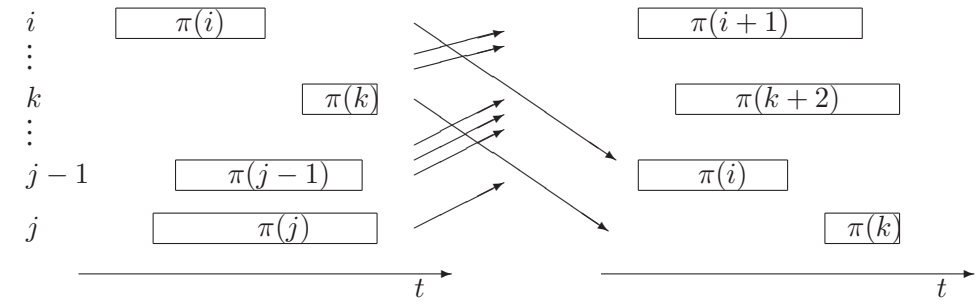

Fig. 3 Shift neighbourhood

Again this procedure can be performed repeatedly for all flights between positions i and j which cannot land before flight $\pi(i)$.

4.3 Selection of neighbour

The selection of a neighbour (of π) can be done in various ways. A simple method is to choose a neighbour randomly. A more advanced method, based on an approximation of the best improvement,

is introduced in this section. With this method it is expected that a neighbour with an improved objective value is found earlier in the neighbourhood search, resulting in a lower computation time.

Let $t_1^\pi, \ldots, t_N^\pi$ be the optimal landing times of the flights when landing in the sequence π (obtained by solving $LP(\pi)$). The gain in objective of a certain neighbour can be estimated using these optimal landing times.

The gain from swapping the flights at positions i and j, $g_{swap}^\pi(i,j)$, can be estimated by:

$$g_{swap}^\pi(i,j) = f_{\pi(i)}(t_{\pi(i)}^\pi) + f_{\pi(j)}(t_{\pi(j)}^\pi) - \left(f_{\pi(i)}(t_{\pi(j)}^\pi) + f_{\pi(j)}(t_{\pi(i)}^\pi) \right)$$

The first part are the (exact) cost for flight $\pi(i)$ and $\pi(j)$, in sequence π. The second part are the estimated cost for these flight after performing the swap, by assuming the landing times are also swapped. This is just an estimation because the optimal landing times of these and other flights may change in the new sequence.

The gain from shifting the flight at position i to position j can be estimated in a similar manner by:

$$g_{shift}^\pi(i,j) = f_{\pi(i)}(t_{\pi(i)}^\pi) - f_{\pi(i)}(t_{\pi(j)}^\pi)$$

or probably more accurately by:

$$g_{shift}^\pi(i,j) = f_{\pi(i)}(t_{\pi(i)}^\pi) - f_{\pi(i)}(t_{\pi(j)}^\pi) + \sum_{k=i+1}^{j} \left(f_{\pi(k)}(t_{\pi(k)}^\pi) - f_{\pi(k)}(t_{\pi(k-1)}^\pi) \right)$$

For all neighbours (including possibly infeasible ones), the appropriate estimation of the gain is calculated. The neighbour with the highest estimated gain is selected. For this selected neighbour the exact sequence of flights is now determined by applying the actual swap or shift operation as described in section 4.2. Then the LP for this sequence is solved. If the objective value of the LP is indeed improved, the selected neighbour is accepted as the new solution. Otherwise the neighbour with the second best estimated gain is selected next and so on.

This is done to minimise the number of neighbours that are evaluated. Evaluation of a neighbour means performing the actual swap/shift operation and solving the LP. By evaluating neighbours in decreasing order of their estimated gain, it is expected to find an improved sequence within the first few neighbours evaluated. The number of evaluations is consequently expected to be lower than when using random neighbour selection. The downside is that the gains for all neighbours have to be estimated beforehand. However, these estimations are easy to calculate (compared to the evaluation). Moreover to speed things up some more, we can choose not to evaluate neighbours with a (large)

negative estimated gain, because it is very likely that the LP-objective of these sequences will not improve. This will however introduce a small risk of missing a better solution.

The whole local search algorithm can be summarised as follows:

LOCAL SEARCH()

1 $\pi =$ FIND INITIAL SOLUTION()
2 $N(\pi) :=$ Set of neighbours of π
3 Estimate gains for all members of $N(\pi)$
4 **while** $N(\pi) \neq \emptyset$
5 **do** $\pi' =$ Sequence for the neighbour with maximum estimated gain in $N(\pi)$
6 (determined by performing the swap/shift operation)
7 **if** $LP(\pi')$ is feasible and $z_{LP(\pi')} \leq z_{LP(\pi)}$
8 **then** $\pi = \pi'$
9 $N(\pi) =$ Set of neighbours of π
10 Estimate gains for all members of $N(\pi)$
11 **else** $N(\pi) = N(\pi) \setminus \pi'$
12 **return** $\pi, t_1^\pi, \ldots, t_N^\pi$

5 A small example

In this example the effects of different cost functions in different situations are demonstrated.

The example contains 14 flights that, according to the timetable, are supposed to arrive between 8:00 and 8:15 in the morning (see table 1). The flights belong to three different airlines (A,B and C).

A and B are airlines that use this airport as a hub, therefore they consider the service to transfer passengers very important. Their costs are based on the number of the transfer passengers that miss their connecting flights in case of a certain delay. Since the cost functions are required to be convex, this is done by increasing the slope of the cost function with the cumulative number of missed transfers. Note that these cost functions will differ from day to day, because the number of transfer passengers will not be the same.

Airline C only cares about punctuality, and consequently the slope of the cost function (after the arrival time according to the timetable) is constant. Flight C3 is considered more important (resulting in a cost function with a steeper slope) than flight C1 and C2.

Now we will use our algorithm to schedule these landings in three different scenarios. In the first scenario the visibility conditions are good. In the second we will assume low-visibility conditions, resulting in larger separation requirements. In the third scenario we will assume the runway cannot be used before 8:15. The schedules resulting from these scenarios are listed in table 2. These schedules can be compared to the FCFS schedule, in which the flights land in the same order as listed in the table and the landing times comply to the separation regulations.

In the first scenario there is no capacity shortage, and all flights are roughly on time. The sequence of the flights is changed somewhat, such that flights with relatively fast increasing costs, are landing on time (or a few minutes early). Some other flights have a small delay (e.g. A3, B3 and B4), but this will not lead to significant costs, because these delay do not result in missed transfers, as can be seen in table 1.

In scenario 2 the situation becomes a lot worse, because of the low-visibility conditions. The original timetable cannot be realized at all and the average delay is around 20 minutes. Again, this is not evenly spread out over the flights: No flights from airline C are delayed. If flight B5 was delayed 20 minutes, 7 passengers would have missed their transfer. A similar effect occurs with flight A1 and B6. On the other hand, flights A3 and B4 are delayed around 40 minutes, but only one passenger from these flights will miss a transfer.

Flight	arrival time			# passengers missing transfers after				
	timetable	earliest	latest	15	30	45	60	minutes delay
A1	8:00	8:00	8:20	0	21	25	41	
A2	8:00	8:00	11:45	0	2	3	7	
B1	8:00	8:00	11:45	0	3	18	24	
B2	8:00	8:00	11:45	0	3	3	4	
C1	8:05	8:00	11:50					
A3	8:05	8:00	11:50	0	0	0	0	
B3	8:05	8:00	11:50	0	1	5	18	
B4	8:05	8:00	11:50	0	0	2	3	
B5	8:10	8:00	11:55	0	7	7	8	
C2	8:10	8:00	8:30					
B6	8:15	8:05	12:00	8	9	13	14	
A4	8:15	8:05	12:00	2	2	13	21	
C3	8:15	8:05	12:00					
B7	8:15	8:05	12:00	2	2	6	17	

Table 1 Flight data

Flight	Timetable	Scenario 1	Scenario 2	Scenario 3
A1	8:00	8:00	8:00	8:15
A2	8:00	8:02	8:17	8:40
B1	8:00	8:03	8:28	8:29
B2	8:00	8:06	8:32	8:54
C1	8:05	8:05	8:03	8:33
A3	8:05	8:18	8:48	9:03
B3	8:05	8:19	8:36	8:47
B4	8:05	8:21	8:44	8:59
B5	8:10	8:08	8:24	8:36
C2	8:10	8:10	8:10	8:25
B6	8:15	8:12	8:06	8:18
A4	8:15	8:13	8:21	8:43
C3	8:15	8:15	8:14	8:22
B7	8:15	8:16	8:39	8:51

Table 2 Schedules obtained by the local search optimisation method

In scenario 3 flights from airline C receive a (large) delay. Now it is clear that flight C3 is considered more important than C1 and C2, because it receives a much smaller delay.

In table 3 the (approximate) number of missed transfers and minutes of delay under the FCFS schedule and the schedule, obtained by our method is shown. All three airlines do much better (especially in scenario 2 and 3) on the criteria they consider important, using the optimised schedule.

The difference between airline A and B (with more detailed cost functions) on one hand and C on the other hand, can be explained in the following way. A and B are likely to receive relatively small delays (if needed), but the breakpoints form a threshold. A less detailed cost functions is more likely to receive no delay at all, but if a delay is inevitable, it is likely to be quite large (because there are no breakpoints behaving as thresholds). This effect is caused by the definition of the scaled cost functions, because the average area under the cost functions of all airlines (ignoring the correction for different interval lengths) needs to be equal. This effect is depicted in figure 4.

	Scenario 1		Scenario 2		Scenario 3	
	FCFS	Model	FCFS	Model	FCFS	Model
Missed transfers (airline A & B)	2	0	20	9	65	30
Minutes delay (airline C)	5	0	50	0	85	50

Table 3 Approximate comparison between FCFS and local search schedule

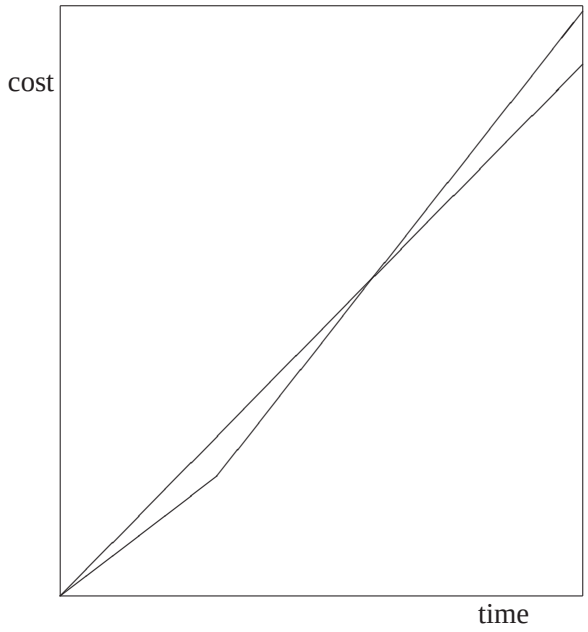

Fig. 4 A scaled cost function without breakpoints and a scaled cost function with one breakpoint

Of course this figure only reflects the situation that would occur if these two flights would be the only flights from two different airlines and the flights have the same preferred arrival time. When several airlines with more flights are involved, the effect is diminished because of difference in cost between flights from the same airline and the scaling factors (which are determined per airline and not per flight). So, some of the (“important”) flights from an airline that provides detailed cost functions may have larger scaled cost, even in the case of (almost) no delay, than some (“not so important”) flights of an airline that provides cost functions with less detail. Moreover, the possible landing intervals and/or preferred landing times will not be equal for many flights in reality, reducing the direct occurrence of this effect further.

6 Experiments

In this section the results, obtained with the algorithm, using the schedule from a major European hub between 20 and 26 September 2004 as input, are presented. In subsection 6.1 the input data are described in more detail. The savings in airline costs and the fairness of the distribution of these costs among the airlines, resulting from the algorithm, are presented in subsection 6.3. Subsection 6.4 describes results concerning the computation time of the algorithm and the quality of the solutions (compared to the optimal solution obtained by the MIP) using the different neighbourhoods.

6.1 Data

The data contains information of all 4135 arrivals at a major European hub between 20 and 26 September 2004. For all flights the following information was included:

- Flight ID (e.g. KL 1860)
- Aircraft Type
- Actual time (and date) of arrival
- Scheduled time (and date) of arrival
- Touch down time
- Runway used for landing
- Origin of flight

Over this 7 day period, 71 flights were removed because they contained no value for aircraft type or runway.

Since there is no need to simulate the exact circumstances between 20 and 26 September 2004, we will use the timetable arrival times from the data as expected arrival times (these determine also the initial FCFS sequence used in the local search). This is also because the causes for deviations from the timetable are not available and consequently we do not know if this was because of the arrival planning at the airport or (for example) a delay at the departure airport.

Not all the input needed by our method, such as possible landing intervals and cost functions, is provided by the data. This input needs to be filled in based on reasonable assumptions.

All cost functions are assumed to be convex, piecewise linear and to have a (unique) minimum at the scheduled arrival time. Further we assume that flights from within Europe have not yet departed and a maximum (departure) delay of 3 hours is acceptable (resulting in a possible landing interval of 3 hours). Inter-continental flights are assumed to be on their way (according to schedule), but by adapting their speed or route, they are able to arrive between 25 minutes before schedule and 30 minutes after schedule.

Specialists from the homebase carrier at this hub provided input on the (general structure of the) cost functions for this airline (and its partners). These costs are strongly related to the number of missed transfers. Exact passenger flows and related costs were not provided by the airline for confidentially reasons.

For other airlines, we assumed punctuality to be important and the cost to increase linearly after the scheduled time of arrival. Besides that, an arrived aircraft will be needed after a while to perform a subsequent (departing) flight, so there will be a point in time, after which the costs will increase faster than before, because this departure has to be rescheduled.

The exact values of parameters, such as the number of missed transfers as a function of arrival delay, linear delay cost, time of subsequent departures and (increased) linear cost for each flight, were obtained using random distributions (with realistic expectations and variances).

Now the data is split up into several problem instances. First, since our method schedules arrivals on a single runway, the data is split up per runway. Runways are not always active. The runway configuration (which runway(s) are used for landings and which for take offs) changes a few times per day because of weather conditions, and the expected number of landings and take offs. If the flights landing on a certain runway are ordered according to their scheduled touch down times, and for two consecutive flights k and $k+1$ the difference between their scheduled touchdown times is larger than 30 minutes, we will assume the runway was closed between those flights. Therefore, aircraft $1, 2, \ldots, k$ can be scheduled independently from aircraft $k+1, k+2, \ldots$ and thus we can create separate problem instances for these groups.

This procedure results in 96 problem instances (ignoring flights for runway 22, which can only be used by small aircraft). These instances contain between 1 and 501 flights.

6.2 Separation

In Table 4 the arrival separation distances under good visibility conditions, according to international regulations, are listed.

		following aircraft		
		Light	Medium	Heavy
leading	Light	3	3	3
aircraft	Medium	5	3	3
	Heavy	6	5	4

Table 4 Wake vortex separation in nautical miles for different weight categories

In low-visibility conditions, the separation distances must be larger. At this airport four different low-visibility conditions are distinguished. Under these conditions the minimum arrival separation is respectively 4, 6, 8 and 9 nautical miles.

6.3 Costs and fairness

Now, the local search algorithm with the swap neighbourhood (see section 4) was used to create arrival schedules for the 96 problem instances. It was assumed there were low visibility conditions during the whole week, such that the minimum separation distance was 6 miles. In practice this means that the timetable cannot be realised and large delays will occur, resulting in large costs. It is interesting to evaluate how fair these (scaled) costs are spread among airlines, using our algorithm.

As mentioned in section 4.1, the arrival sequence used in practice is similar to first come (scheduled), first served. This is also the initial sequence in the local search method and we can compare this schedule to the final schedule obtained using the algorithm.

We compare the average cost per aircraft (both scaled and real). This measure has only meaning for airlines with enough flights: Some flights will receive a very large delay. If an airline with only one flight a week receives such a delay, its average cost will be very large. However in the next week the same flight might be not delayed at all. So for this airline a week is not enough to evaluate its average costs. Therefore, we considered only airlines with have on average 5 flights or more per day (35 flights or more per week). There are 10 airlines with this many flights in our data. For reasons of confidentiality the real names of the airlines are omitted.

In table 5 the average cost per flight for these 10 airlines are listed. All airlines achieve improvements, which can be considered important for the acceptance of this method. Although the real cost have no scale and therefore no meaning in terms of “real” money, the relative savings are meaningful. These savings are on average 47 percent of the current FCFS costs. This shows that, especially during low visibility conditions, tremendous savings can be accomplished using optimisation.

From table 5 can also be concluded that when the average FCFS costs from an airline are below (above) average, the gain and local search costs are also below (above) average. This can be explained by the fact that a small (large) share of the flights from these airlines are scheduled in a peak period. For flights planned outside peak periods, it might be hard to reach improvement, because there will not be much delay and thus cost are already low. This occurs for example for airlines B, D and F.

For flights in a peak period, some profitable position changes are probably easily found, but there will also be a lot of other flights with large costs, because of the capacity shortage occurring during peak

		FCFS		Local Search		Gain	
Airline	# flights	Scaled Cost	Real Cost	Scaled Cost	Real cost	Scaled Cost	Real Cost
A	1476	532	4996	405	3717	127	1279
B	220	747	1802	345	902	402	900
C	137	1179	3852	556	1439	623	2412
D	94	1051	2228	239	942	812	1285
E	79	1763	4801	832	2213	931	2588
F	70	574	1505	475	1490	99	15
G	70	1601	4348	713	2237	888	2111
H	61	812	712	74	131	738	581
I	54	2195	5396	1123	3273	1072	2123
J	41	1973	8041	1016	3781	958	4260
Mean		1243	3768	578	2013	665	1755

Table 5 Average costs per flight for 10 biggest airlines

periods. This occurs with flights from airlines I,J, and A. As a result of the large number of flights from airline A, sometimes the only choice might be to either delay one (group) or another (group of) flight(s) from airline A.

It can be concluded that both the cost resulting from the FCFS schedule and the local search schedule depend on the original timetable. However, the design of the timetable is beyond the scope of this research.

6.4 Algorithm's performance

It is also interesting to look at the performance of the algorithm. This can be done with respect to optimality and computation time.

Our method was implemented in C++. The MIP and LP problems are solved using ILOG CPLEX 7.5. All computations were performed on a Compaq computer with an Intel Pentium III processor (866 MHz), 256 MB physical memory and a Linux operating system.

6.4.1 Optimality

Some instances can be solved in reasonable time using CPLEX's MIP-solver. The solution obtained is optimal and can be used to evaluate the quality of the solutions obtained using the local search methods with the shift and swap neighbourhoods, as described in section 4.

These instances contained between 15 and 51 flights. Smaller instances were not included in the comparison because they often generate trivial schedules (e.g. the FCFS schedule is optimal).

The instances were solved under two visibility conditions: good visibility and the low-visibility conditions as used in the experiments in subsection 6.3. This was done because, for some instances, the MIP-formulation could be solved in reasonable time for good but not for low-visibility conditions.

A total of 21 instances were solved using the MIP-solver. The comparison of the solutions is shown in table 6. The gap is defined as the additional cost of the local search schedule compared to the optimal cost (as found by the MIP solver). Assuming these numbers are representative for other instances as well, we can combine this with the results from subsection 6.3. Those were obtained using the swap-neighbourhood and the cost savings are 47% compared of the initial FCFS costs. Putting this together, shows that the local search method, using the swap neighbourhood, on average yields 127% of the optimal costs, while the current FCFS schedule on average yields 240% of the optimal costs. Using the shift neighbourhood, on average 107% of the optimal costs would be yielded. This shows the local search method already achieves a very large part of the possible savings and is therefore very useful in practice. Note that the local search solution is always better than the FCFS solution, because the latter is used as the initial solution for the local search method.

	Shift	Swap
# optimal instances	12	5
average gap	7%	27%
maximum gap	51%	135%

Table 6 Quality of the solutions found using different neighbourhoods for 21 instances

When an instance cannot be solved using the MIP-solver in reasonable time, the results from the swap and shift neighbourhood can still be compared. This was done for 14 additional instances (most under low-visibility conditions). Together with the 21 MIP-instances, this gives 35 instances, of which in 19 the shift neighbourhood yielded a better solution than the swap-neighbourhood. In 10 cases the reverse happens and in 6 cases the solutions were equal. The maximum gap (compared to the best solution from the two neighbourhoods) is now 37% for the shift neighbourhood and 120% for the swap neighbourhood. The average gaps are 5% and 10% respectively. However, the total scaled costs, from these instances, are 3% larger using the shift neighbourhood than using the swap neighbourhood. This indicates that the swap neighbourhood yields better solutions in cases were costs are relatively large (e.g. low-visibility conditions and peak periods).

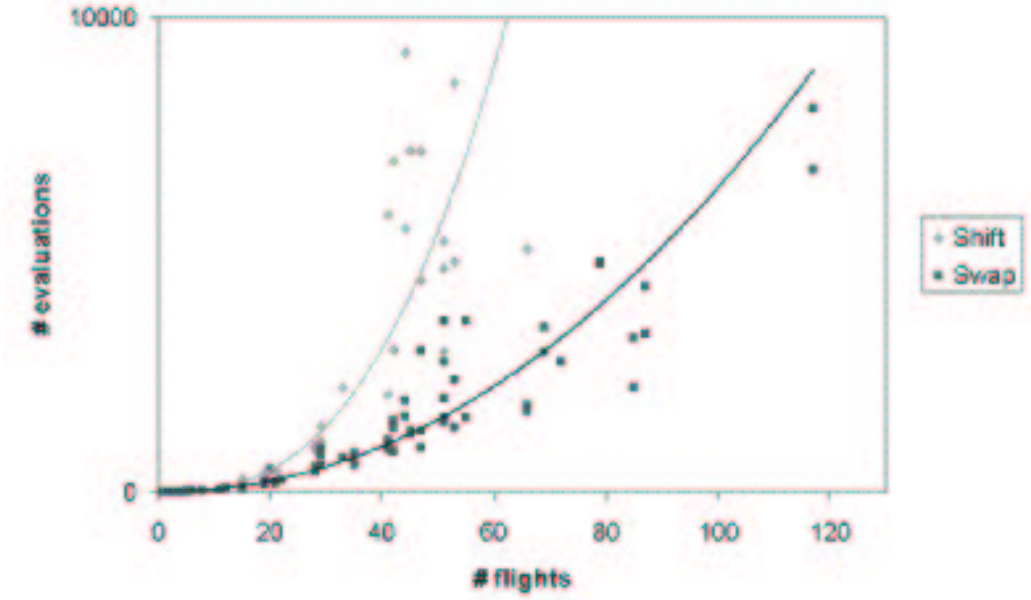

Fig. 5 Relation between number of evaluations and number of flights and trendlines

6.4.2 Computation times

The computation times of an instance depend mainly on the number of flights and the number of neighbour evaluations (see section 4.3). Such an evaluation includes solving the LP-problem for the corresponding sequence. The time needed to solve this LP, increases somewhat with the number of flights.

Let N be the number of flights. As stated in section 4.2, the swap neighbourhood contains at most $\frac{N(N-1)}{2}$ sequences and the shift neighbourhood $N(N-1)$. After finding the final solution, all sequences in the neighbourhood of the solution have to be evaluated (to be sure it contains no better solution). The final solution is found usually after fewer evaluations using the swap neighbourhood than when using the shift neighbourhood.

It is for those two reasons that we find a larger total number of evaluations using the shift neighbourhood. The time needed to perform an evaluation will be approximately equal using both neighbourhoods for a certain instance, because a similar LP-problem of the same size has to be solved.

The time needed for an evaluation of a neighbour is (using both neighbourhoods) approximately 50 milliseconds for an instance containing 20 till 50 flights, 150 milliseconds for 50 till 80 flights and around 450 milliseconds for an instance containing 120 flights.

The relation between the number of flights and the number of evaluations needed are shown in figure 5. It can be concluded that the total computation time using the swap neighbourhood is approximately

within 1 minute for instances up to 40 flights and about ten minutes for instances with 80 flights. For the shift neighbourhood, it is within 1 minute for instances up to 40 flights and about ten minutes for instances with 60 flights.

7 Conclusions and further research

In this paper a method to include airlines' preferences in the scheduling of aircraft landings is presented. Airlines provide a cost function for each of their flights. These functions are scaled per airline to obtain equity between airlines. The schedule also incorporates safety regulations.

Experiments using data from a week at a major European hub, show considerable cost reductions (compared to the current First-Come-First-Served schedule), especially under low-visibility conditions. The method yields cost savings for all airlines, which may be helpful for the acceptance of the method.

An optimal schedule can be obtained using a mixed integer programming solver. However, computation times become very large. Therefore, also a heuristic using local search (to obtain a landing sequence) and linear programming (to obtain the optimal landing times) was introduced. This heuristic can solve instances containing over 100 flights within reasonable time, while still providing substantial savings compared to the FCFS schedule. Comparing some instances that could be solved using the MIP solver showed that the average savings yielded by the heuristic are a large part of the total possible savings.

Further research to improve the performance of the local search heuristic is possible. If schedules have to be generated fast, evaluating less possible landing sequences (by excluding ones with do not seem promising) is an option. Another way to accomplish this, might be limiting the maximum number of positions used in a swap or a shift (swap only flights that are less than 3 position apart in the current sequence). These methods might however introduce a small risk of obtaining a worse solution than using the current method. To obtain a better solution, it might be profitable to use the shift and swap neighbourhood combined in one search, by alternating them or using one exhaustively and then using the other.

From an operational viewpoint it is interesting to include the possibility of cancelling flights into the model. Another possibility for further research is to include the scheduling of departing flights in this schedule, because they are mutually dependent on the arriving flights. These approaches provide possibilities for additional savings. Also research on the sensitivity of resulting schedules to disruptions, would be interesting.

8 Acknowledgements

We thank the airline specialists for the fruitful discussions. Also we would like to thank Ger Koole from the Vrije Universiteit, Amsterdam and Rene Verbeek, Michel van Eenige and Ronny Groothuizen from the National Aerospace Laboratory NLR for their input.

9 References

1. Free-for-all flights. *Scientific American*, 273(6):34–37, 1995.
2. N. Ashford and P.H. Wright. *Airport Engineering*. Wiley-Interscience, 3rd edition, 1992.
3. M. Ball, T. Vossen, and R. Hoffman. A general approach to equity in traffic flow management and its applications to migating exemption bias in ground delay programs. In *Proceedings 5th USA/EUROPE Air Traffic Management R&D Seminar*, 2002.
4. J.E. Beasley, M. Krishnamoorthy, Y.M. Sharaiha, and D. Abramson. Scheduling aircraft landings - the static case. *Transportation Science*, 34(2):180–197, 2000.
5. J.E. Beasley, M. Krishnamoorthy, Y.M. Sharaiha, and D. Abramson. Displacement problem and dynamically scheduling aircraft landings. *Journal of the Operational Research Society*, 55(1):54–64, 2004.
6. J.E. Beasley, J. Sonander, and P. Havelock. Scheduling aircraft landings at london heathow using a population heuristic. *Journal of the Operational Research Society*, 52(5):483–493, 2001.
7. G.C. Carr, H. Erzberger, and F. Neuman. Airline arrival prioritazion in sequencing and scheduling. In *2nd USA/EUROPE Air Traffic Management R&D Seminar*, 1998.
8. G.C. Carr, H. Erzberger, and F. Neuman. Delay exchanges in arrival sequencing and scheduling. *Journal of Aircraft*, 36:785–791, 1999.
9. E.P. Gilbo and K.W. Howard. Collaborative optimization of airport arrival and departure traffic flow management strategies for cdm. In *3rd USA/EUROPE Air Traffic Management R&D Seminar*, 2000.
10. R. Hoffman and M.O. Ball. A comparison of formulations for the single-airport ground-holding problem with banking constraints. *Operations Research*, 48(4):578–591, 2000.
11. T. Inniss and M.O. Ball. Estimating one-parameter airport arrival capacity distributions for air traffic flow management. *Air Traffic Control Quarterly*, 12(3):223–251, 2004.
12. L. Kholfi Loan, G. S. Donohue, and C-H. Chen. Proposal for demand management using auction-based arrival slot allocation. In *Proceedings of the 5th EUROCONTROL / FAA ATM R&D Seminar*, 2003.
13. M.S. Nolan. *Fundamentals of Air Traffic Control*. Wadsworth Pub Co, 3rd edition, 1998.
14. E.H. Phillips. Free flight poses multiple challanges. *Aviation Week & Space Technology*, 144(13):27, 1996.
15. S.J. Rassenti, V.L. Smith, and R.L. Bulfin. A combinatorial auction mechanism for airport time slot allocation. *Bell Journal of Economics*, 13(2):402–417, Autumn 1982.
16. O. Richetta. Optimal algorithms and a remarkably efficient heuristic for the ground-holding problem in air traffic control. *Operations Research*, 43(5):758–770, 1995.
17. T. Vossen and M. Ball. Slot trading oppurtunities in collaborative ground delay programs. Working paper, availabe via http://mail3.rhsmith.umd.edu/Faculty/KM/papers.nsf.