Integrated Gate and Bus Assignment at Amsterdam Airport Schiphol

G. Diepen*, J.M. van den Akker* J.A. Hoogeveen*

Department for Information and Computing Sciences
Utrecht University
P.O. Box 80089, 3508 TB Utrecht, The Netherlands
{diepen,marjan,slam}@cs.uu.nl

Extended abstract

Abstract. At an airport a series of assignment problems need to be solved before aircraft can arrive and depart and passengers can embark and disembark. A lot of different parties are involved with this, each of which having to plan their own schedule. Two of the assignment problems that the 'Regie' at Amsterdam Airport Schiphol (AAS) is responsible for, are the gate assignment problem (i.e. where to place which aircraft) and the bus assignment problem (i.e. which bus will transport which passengers to or from the aircraft). Currently these two problems are solved in a sequential fashion, the output of the gate assignment problem is used as input for the bus assignment problem. We look at integrating these two sequential problems into one larger problem that considers both problems at the same time. This creates the possibility of using information regarding the bus assignment problem while solving the gate assignment problem. We developed a column generation algorithm for this problem and have implemented a prototype. To make the algorithm efficient we used a special technique called stabilized column generation and also column deletion. Computational experiments with real-life data from AAS indicate that our algorithm is able to compute a planning for one day at Schiphol in a reasonable time.
Keywords: gate assigment, integrated planning, airports, column generation, integer linear programming

1 Introduction

Between the time an aircraft lands at an airport and the time it departs again many things must happen. One of the most obvious things is that passengers needs to disembark the aircraft. Moreover, the aircraft needs to be refueled, new passengers need to board, new supplies have to be put on board, the aircraft has to get cleaned. All of the actions take place while the aircraft is standing at a *gate*. We will consider the arrival of an aircraft until the following departure of

* Supported by BSIK grant 03018 (BRICKS: Basic Research in Informatics for Creating the Knowledge Society)

ATMOS 2008
8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems
http://drops.dagstuhl.de/opus/volltexte/2008/1591

the same aircraft as one *flight*. The *gate assignment problem* deals with assigning a given set of flights to a set of gates such that certain criteria are met.

In this paper, we consider the gate assignment at Amsterdam Airport Schiphol (AAS). We investigate the daily planning, i.e. the creation of a planning for the upcomimg day on the basis of the available information about the flights of that day. In Diepen et al. [4], we have presented a column generation algorithm to create an assignment for aircraft to gates that is as robust as possible, meaning that any small deviation from the scheduled arrival and departure times should not result in lots of rescheduling.

Some flights are not assigned to a gate with an airbridge but to a so-called remote stand. This implies that passengers have to be transported to and from the aircraft by buses. We have shown how we can create a robust schedule for these platform buses by a similar type of column generation algorithm (see Diepen [3]) in case the gate assignment is given.

This approach resembles the way AAS is actually solving these two problems currently. First the gate assignment problem is solved, the solution of which is then used as input for the bus planning problem. Although the bus planner have the possibility to influence the gate planning by providing preferences, in general the two problems are solved in a sequential way.

Observe that this could imply that a schedule for the gate assignment results in a very bad schedule for the bus planning. In many cases minor changes to the original solution for the gate assignment problem would allow better assignments for the buses. So although this would mean a sub-optimal solution for the gate assignment problem to be used, the solution for both the gate and bus planning as a whole would improve.

In this paper, we focus on *the integration of gate assignment and bus planning*. Our goal is to achieve better overall robustness and a more efficient bus planning without too much negative effects on the gate assignment. The airport authorities at AAS have indicated that robustness is very important for them, in order to limit the amount of gate changes during the day of operations.

During the last years, a signficant amount of research has been performed on the integration of real-life scheduling problems. For example Freling, Huisman, and Wagelmans [6] look into the integration of solving the combination of the vehicle and crew scheduling problems that arise in the public transport scheduling. They present two different models and algorithms for solving the integrated version of the two problems, and compare the results to the results obtained by using the standard sequential approach.

One of the areas where the integration of real-life scheduling problems is investigated a lot, is in the *airline* industry. Cordeau et al. [2] investigate the integration of the aircraft routing problem with the crew scheduling problem. They propose a solution approach based on Benders decomposition and show that solving these two problems as one integrated problem yields significant cost savings. Other integrations that have been considered are schedule assignment and the fleet assignment problems (see Rexing et al. [9] and Lohatepanont and

Barnhart [8]) and the integration of the fleet assignment and the crew scheduling problems (see Gao [7], Clarke et al. [1], and Sandhu and Klabjan [10]).

At Amsterdam Airport Schiphol, the software package currently in use for solving the gate assignment problem, uses a rule based approach for optimizing the assignment. It includes many aspect, however, it does not support the main thing we aim for, robustness. Furthermore, it is also capable of scheduling additional processes besides the assignment of aircraft to gates. For instance, in Vancouver the same program is used and there the scheduling of the push back trucks is also handled by the program.

The purpose of the research described in this paper is to enable the use information of regarding the bus planning problem while solving the gate assignment problem. Instead of an iterative method in which the separate problems are solved in turns and are allowed to send constraints or preferences to each other, our approach is to combine the two assignment problems into *one big problem* and solving this one big problem as a whole, where the objective is to maximize overal robustness.

The outline for the remainder of the paper is as follows: In Section 2 we will give the problem formulation and our model and in Section 3 we present solution method. Furthermore, in Section 4 we will report on the results of the experiments that we performed and finally, in Section 5 we give our conclusions.

2 Problem formulation

In this section, we describe the problem and present an integer linear programming formulation. For the upcoming day we want to create a gate assignment for a given set of flights and a planning for the platform buses transporting passengers to and from flights at a remote stand.

For the gate asssignment several properties of the flights are important:

- Arrival and departure time
- Region of origin and destination (Shengen/EU/Non-EU)
- Size category
- Ground handler

At AAS the ground handlers are divided into two groups: KLM Ground Services and other companies. Clearly, two flights cannot be assigned to the same gate at the same time. At AAS the minimum amount of idle time between two consecutive flights at a gate is 20 minute. For each gate it is known which regions (because of safety regulation), size categories, and grounds handlers it can serve. This results in constraints to ensure that at a gate there are only flights maching the properties of the gate with respect to region, size of the aircraft and ground handler.

Moreover, certain preferences might be taken into account. For example, some airlines such as KLM have their own gates or want their flights to be grouped as much as possibe on certain gates, for example we could require that at least 5 out of 7 Swiss flights are on a specific gate.

Flights that stay on the ground for a longer period, eg. 3 hours, may have to be sprit. This means that after some time the flight is removed from the gate and later is moved back to some (possibly other) gate. We included this in our model, but omit the description for reasons of brevity.

Our objective is to create an assignment schedule that is as robust as possible, meaning that the resulting schedule is able to cope with minor disturbances during the actual day as well as possible. The following picture shows an example of a schedule that is typically non-robust and can be improved by interchanging flights 3 and 4.

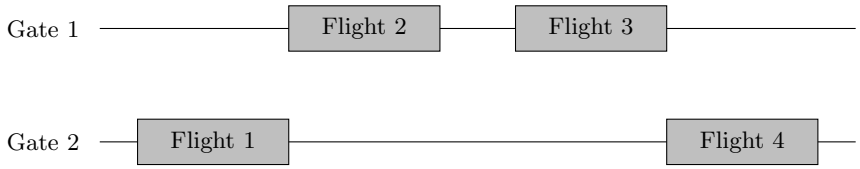

Fig. 1. Example of a non-robust schedule.

Observe that a schedule is best able to cope with disturbances if all idle times between each pair of consecutive flights on a gate are as large as possible. We model this with a cost function that greatly penalizes short idle times, while giving very low cost to large, and thus favorable idle times.

For the cost of the idle time t between two consecutive flights v and w on a gate we use the same cost function presented in [4]

$$c^G(t) = conv(v, w)1000(\arctan(0.21(-t)) + \frac{\pi}{2}),$$

where $conv(v, w)$ denotes the convenience multiplier expressing the preference of flight w directly succeeding flight v on a gate. For example, this multiplier is large is v and w belong to the same airline since in this case the airline has a clear incentive to make v depart on time.

If a flight is handled at a remote stand, the passengers are moved to and from the terminal by bus. The number of buses needed depends on the number of passengers. In this way, each flight assigned to a remote stand, results in a number of bus trips. At arrival all trips takes place at the same time, and for the departure there by rule have to be at least two trips and the first trip starts already some time before the departure of the flight. When ordering buses and drivers, AAS can specify the amount buses required per 15 minutes. As a results the bus drivers (about 60) on a day work in about 20 types of shifts, where shifts longer than 4.5 hours contain a mandatory break.

To maximize robustness, we make use of a similar cost function of the idle times t between consecutive trips of the same bus. The exception is that at each given time we have significantly lowered the total cost, this to resemble the fact that the gate assignment is still the more important problems of the two:

$$c^B(t) = 50(\arctan(0.21(-t)) + \frac{\pi}{2}),$$

By taking the sum of the total cost of both sub problems, we now have a representation for the quality of the robustness of a solution as a whole.

The ILP formulation. The model is obtained by combination and extension of the separate models presented in [4] and [3] to solve the gate assignment and the bus planning problems respectively.

Our model is based on so-called *gate plans*, which consist of a set of flights assigned to one gate. We aggregate gates with the same properties into groups of gates and each such group we refer to as a *gate type*. These properties contain at least the origin/destination, size and ground handler. However, a trivial aggregation in which each separate gate (except for the platform stands) is considered a single type is also possible.

We define the decision variable

$$x_i = \begin{cases} 1 \text{ if gate plan } i \text{ is selected} \\ 0 \text{ otherwise,} \end{cases}$$

Since it might be non-trivial to assign all flights to a gate, allow a flight to be unassigned at high cost. This is modelled by the binary variable UAF_v. Let V denote the number of flights, A the number of gate types, S_a the number of gates of type a, and K the number of preferences. Now the robustness cost of the gate assignment are given by:

$$\text{Min} \quad \sum_{i=1}^{N} c_i^G x_i + \sum_{v=1}^{V} Q_v \mathrm{UAF}_v$$

and the gate plan have to satisfy the following constraints:

$$\mathrm{UAF}_v + \sum_{i=1}^{N} g_{vi} x_i = 1 \qquad v = 1 \ldots V \tag{1}$$

$$\sum_{i=1}^{N} e_{ia} x_i \leq S_a \qquad a = 1 \ldots A \tag{2}$$

$$\sum_{i=1}^{N} \sum_{v=1}^{V} \sum_{a=1}^{A} p_{vak} e_{ia} g_{vi} x_i \geq P_k \qquad k = 1, \ldots, K \tag{3}$$

where

$$g_{vi} = \begin{cases} 1 \text{ if flight } v \text{ is in gate plan } i \\ 0 \text{ otherwise,} \end{cases}$$

$$e_{ia} = \begin{cases} 1 \text{ if gate plan } i \text{ is for gate type } a \\ 0 \text{ otherwise,} \end{cases}$$

$$p_{vak} = \begin{cases} 1 \text{ if flight } v \text{ has preference for gate of type } a \text{ in preference } k \\ 0 \text{ otherwise,} \end{cases}$$

Constraint (1) ensures that all flights are either present in one of the selected gate plans, or the unassignment variable for the flight will have the value 1, resulting in a penalty in the objective function.

Constraint (2) ensures that we select as many gate plans of a certain type as there are gates of the type and Constraint (3) ensures that we fullfill all of the preferences that are given with regards to the gate assignment. Here P_k is the minimum required number of flights with a preference for gate type a that we have to assign to a gate of type a to meet the preference constraints, e.g. the constraint can be that at least 7 out of the 10 flights of a certain airline are at a given gate.

In case we only need to solve the bus planning problem, we are given a set of flights of which it is known where exactly they are standing. With this information we can create the trips needed to transport all the passengers and in the model we must ensure that each of these trips is either driven by a bus, or it is left unassigned with a penalty cost.

For the combination of the two problems, we do not yet know which flights will be placed on the platform (and also, on which platform) and therefore we have to find a way to determine which trips we actually need to assign to buses.

To handle this problem, we generate all possible trips for flights that could be assigned to the remote stands. This means that for each of these flights we create the trips for each of the platforms that it can be assigned to. For example, if an arriving flight requires two trips because of the number of passengers and it can be assigned to the D/E platform, as well as the B platform it means that we will create two trips from the D/E platform and two trips from the B platform to the terminal building. Similarly, not only different platforms, but also different destinations in the terminal building must be considered. For each possible combination we would have to create the trips also. To allow for this coupling we will work with all possible trips and determine which of these are needed in a solution and which are not. For this purpose we will use the variables NNT_t for each trip t to denote whether the trip t needs to be assigned to a bus or that the trip is irrelevant for the assignment problem.

Similar to the gate assignment, we define bus plans as the set of trips performed by one bus. We define

$$y_j = \begin{cases} 1 \text{ if bus plan } j \text{ is selected} \\ 0 \text{ otherwise,} \end{cases}$$

and the binary variable UAT_t to signal if trip t is unassigned. Let T be the number of trips, B be the number of shift types and T_b be the number of buses

with drivers available is shift b. We now obtain the following model:

$$\text{Min} \quad \sum_{i=1}^{N} c_i^G x_i + \sum_{v=1}^{V} Q_v \text{UAF}_v + \sum_{j=1}^{M} c_j^B y_j + \sum_{t=1}^{T} R_t \text{UAT}_t$$

subject to

$$(1) - (3)$$

$$\sum_{j=1}^{M} f_{jb} y_j \leq T_b \qquad b = 1 \ldots B \tag{4}$$

$$\text{NNT}_t + \text{UAT}_t + \sum_{j=1}^{M} h_{tj} y_j = 1 \qquad t = 1 \ldots T \tag{5}$$

$$\text{NNT}_t + \sum_{i=1}^{N} \sum_{v=1}^{V} t_{tvi} g_{vi} r_i x_i = 1 \qquad t = 1 \ldots T \tag{6}$$

$$x_i \in \{0, 1\} \qquad i = 1 \ldots N \tag{7}$$

$$y_j \in \{0, 1\} \qquad j = 1 \ldots M \tag{8}$$

where

$$f_{jb} = \begin{cases} 1 \text{ if bus plan } j \text{ for a shift of type } b \\ 0 \text{ otherwise,} \end{cases}$$

$$h_{tj} = \begin{cases} 1 \text{ if trip } t \text{ is in bus plan } j \\ 0 \text{ otherwise,} \end{cases}$$

$$t_{tvi} = \begin{cases} 1 \text{ if assigning flight } v \text{ to gate plan } i \text{ implies trip } t \text{ must be driven} \\ 0 \text{ otherwise,} \end{cases}$$

$$r_i = \begin{cases} 1 \text{ if gate plan } i \text{ is for a remote stand} \\ 0 \text{ otherwise,} \end{cases}$$

Constraint (4) ensures that for each bus shift, we select at most the number of buses present in that shift.

Constraint (5) states that either not needed, or, in case it is needed, must either be assigned to a bus plan or it must be explicitly become unassigned at high cost.

Without any further constraints on the NNT_t variables, the easiest solution would be to set the value of all of these variables to 1 and all of the trip constraints would be satisfied right away. Constraint (6) ensures that this cannot happen for trips that are defined for flights assigned to the remote stands. It is also this constraint that actually links the gate and bus model into one large model.

3 Solving the problem

3.1 Assigning flights to gate plans and trips to bus plans

Observe that the above model determines for each group of gates and each group of shifts an equal sized set of gate plans and bus plans respectively. To

approximate the optimal solution of the above ILP-formulation, we will first relax the integrality constraints (7) and (8). After that we will solve the resulting LP relaxation to optimality by making use of column generation.

The pricing problem. After each iteration of the column generation process, we need to determine whether other columns exist that might improve the value of the objective function, the so-called *pricing problem*. In our case we have to solve two types of pricing problems, one for finding gate plans and one for finding bus plans.

The pricing problem for the gate assignment part boils down to a set of shortest path problems. For each gate type a we define a graph G^a, the nodes of which are the flights that are allowed to be assigned to gate type a and there is an arc between each pair of flights (v, w) such that w can directly succeed v on that gate, i.e., the difference between the arrival time T_w^{arr} of flight w and the departure time T_v^{dep} is at least 20 minutes. Furthermore we add a source vertex s with an arc to every node v and a sink t and an arc from every node to t. Now every path in G_a corresponds to a feasible gate plan and vice versa. To be able to solve the pricing problem as a shortest path problem, we set the cost of arc (v, w) equal to the contribution of flight v to the reduced cost as follows:

$$c^G(T_w^{\text{arr}} - T_v^{\text{dep}}) - \pi_v - \sum_{k=1}^{K} p_{vak}\psi_k - \sum_{t=1}^{T} t_{tv}\rho_t.$$

where the dual multipliers π_v for flight v and ψ_k for preference k follow from Constraint (1) and Constraint (3) respectively. Moreover, ρ_t is the dual multiplier of Constraint (6), which only applies to gate plans that are for remote stands (because only then $r_i = 1$). The last term which incorporates the 'coupling' constraint is the only difference with the pricing problem for the gate assignment problem. We may assume that the flights are sorted by their arrival times, which implies a topological order on the vertices of the graph. Because we now have a DAG with a topological order it is possible to find the shortest path in $O(|V| + |E|)$ time.

The pricing problem with regards to the bus problem boils down to a similar type of shortest path problem and is the same as the pricing problem for solving only the bus planning problem separately. The only difference is that the size of the individual graphs is larger due to the increased number of trips.

Because solving all of the pricing problems in each iteration may be rather time consuming, we have tried out different strategies with regards to which of the pricing problems we solve during each iteration. One possible approach is to interleave the solving of the pricing problems; one iteration we solve the pricing problems for the buses and the other iteration we solve the pricing problems for the gates.

Although, after some initial tests we found that searching for both gate and bus plans with negative reduced cost from the beginning on turned out to work better than the other possibilities.

In [4] and [3], we generated a pool of additional columns that can be added to the ILP and enable us to solve the ILP in a reasonable amount of time. For gate assignment these column are obtained by after the pricing problem has been solved, forbidding one flight in the gate plan and resolve the shortest path problem. We perform this step for every flight in the optimal solution of the pricing problem. For bus planning we generate additional columns in the same way. When solving the problems separately, the columns are added when we start solving the ILP. However, when solving the integrated problem all additional columns with negative reduced cost are already added during the column generation process.

Improvements in solving the LP. During our first experiments, it turned out that the LP problem tends to require a long solution time and be a very degenerate. This degeneracy appears in two ways during the column generation process. First, resolving the restricted master problem after new columns have been added takes quite many iterations and second, new columns that are generated with negative reduced cost do not improve the objective function after they have been added to the restricted master problem.

We have applied two different approaches to improve the solving of the LP. The first approach we used is *column deletion* and consists of the removal of columns with too large positive reduced cost after every given number of iterations. The effect of this removal is not only that the model is simplified and some degeneracy is removed, but also that the resulting model is smaller and therefore it can be solved more quickly. For solving the problems separately, this approach showed promising results for decreasing the computation required time when .

The second approach we implemented is so-called **stabilized column generation**. This technique was introduced in du Merle et al. [5] and consists of a combination of two techniques. One technique is the addition of bounded surplus and slack variables to the original primal problem to overcome degeneracy. The second technique consists of adding surplus and slack variables that have a positive coefficient in the objective function. Combining these two techniques both stabilizes and accelerates the column generation procedure. It decreases the amount of degeneracy a.o. because the slack and surplus variables give more possibilities for assigning a positive value to a newly added column. Moreover, it has a positive effect on the tailing-off effect, i.e. slow convergence. A more elaborate explanation is omitted for reasons of brevity.

Solving the ILP. After the LP is solved to optimality by means of column generation, we are not finished yet because this solution might be fractional. In case it is integral, we are finished since we have an integral solution that is optimal. If we do not have an integral solution, we proceed as follows:

1. first we add all unique extra gate and bus plans that were generated as extra columns while solving the pricing problems.
2. we then add all the unique variables that were taken out during the column generation

3. we reinstate the integrality constraints (7) and (8)

Solving the resulting ILP turned out to be still quite difficult. In order to speed up this solving, we added additional constraints to the problem. These constraints act as a rounding-heuristic. For each flight and for each bus these additional constraints were created in the following way:

1. Determine if a flight or trip that is only present in selected gate plans and bus plans respectively that are all of the same type, meaning that in the fractional solution a flight or a bus trip is always assigned to one particular gate type or one particular bus shift.
2. Create a constraint that ensures the flight or the trip has to be assigned to that particular gate type or bus shift in an integral solution.

Although the above constraints might cause the optimal solution of our initial ILP to be cut off, our experiments did not show any noticeable negative side effects with regards to the cost of the integral solution compared to the optimal fractional solution.

3.2 Assigning gate and bus plans to the actual gates and buses

After solving the ILP from the previous section, we have determined the set of gate and bus plans that provide a (near) optimal solution. For each group of gates and each group of shifts we have an equal size set of gate plans and bus plans respectively. The one thing still left to do is to assign each gate plan and each bus plan to each unique gate and bus respectively.

In case of the bus planning problem, this part is trivial since the buses within one shift do not have any differences at all; it really does not matter to which of these buses a particular bus plan is assigned to.

However, for the gate assignment problem it depends on the definition of the gate types. If each single gate is a separate type, we already have an assignment of flights to physical gates and this step is also trivial.

If we have grouped the gates with certain equal properties into types, the individual gates within such a type still might be different on some other, less important properties. These additional properties can be used for determining to which physical gate a particular gate plan needs to be assigned.

Since the size of these problems is relatively small (in the order of 5 to 10 gates within one group) it is probably most effective to leave this up to the gate planner to do this manually.

4 Experimental results

For testing our model, we wrote a prototype in C++ and ran numerous experiments. All experiments were ran on on Pentium 4 2.8 GHz computer equipped with 1GB of RAM. The solver we used for solving all (I)LP problems is Cplex 9.1.3 via the Concert Technology interface.

AAS provided us with both data regarding the gate assignment problem, which consisted of all flight information for three high-season (HS) days and three low-season (LS) days and data regarding the bus planning problem with all information regarding buses for one complete month.

From the supplied gate data we created two types of instances. In one type of instances we aggregate all gates with identical properties (e.g. size, region, ground handler, pier) into groups of gates. We refer to this type of instances as Grouped Gates (GG). Furthermore, we constructed instances where every gate is considered as a group with size one except for the platform gates. Recall that for these instances our algorithm directly assigns flights to physical gates. We refer to this type of instances as Single Gates (SG). This deaggregation results in over twice the number of gate types, as can be seen in Table 1. This way we created 12 instances with regards to the gate and flight information. The high-season instances contain about 600 flights and about 1000 arival/departure events for the bus planning. For the low-season instances these numbers are 500 and 900 respectively.

To create a sufficiently large number of experiments, we combined each of the 12 gate assignment instances with the buses and shifts of all 30 of the bus planning problem instances. These instances contain about 60 buses and about 20 type of shifts (of which about 70 percent is long enough to contain a mandatory break). We may expect the set of buses available at each given time of the day should roughly be enough for driving all trips.

Instance	Gates	Gate types	Remote stands
Grouped	128	40	34
Single	128	94	34

Table 1. Sizes of the provided instances with regard to gates

In Table 2 we present the general results with regards to solving the LP part of the problem. We combined each instance of the gate assignment problem with the 30 available instances of the bus planning problem and we present the average time over these 30 instances needed for solving each combination, the minimum time, and the maximum time. We also present the average number of iterations needed to solve the LP relaxation and finally, we also present the average time needed in each iteration of the column generation process to solve the pricing problem and the time needed for resolving the Restricted Master Problem (RMP) after we have added the columns found when solving the pricing problem.

Our experiments indicate that the LP can be solved within a reasonable amount of time. From Table 2 we can see that a significant amount of the time needed for solving the LP-relaxation is spent in solving all the separate pricing problems. Since all parts of the pricing problem that need to be solved can be solved completely independent from each other, we could easily bring down

	Total time LP (s)				Avg time (s)/iter	
Instance	Average	Min	Max	Avg iter	RMP	Pricing
02-08-GG	1129.6	967.8	1472.0	161.67	2.8	3.9
02-08-SG	2070.1	1752.1	2657.7	171.90	4.8	6.8
03-08-GG	973.9	864.7	1213.2	148.27	2.6	3.7
03-08-SG	1847.4	1627.4	2337.8	163.07	4.4	6.5
04-08-GG	1142.6	1010.4	1641.3	157.50	3.2	4.0
04-08-SG	2575.2	2189.9	3970.3	212.77	4.6	7.2
15-03-GG	658.5	560.3	769.3	165.17	1.1	2.7
15-03-SG	1235.8	1094.6	1472.0	175.17	1.9	4.8
16-03-GG	710.0	623.8	850.4	161.90	1.3	2.8
16-03-SG	1383.4	1144.0	1661.5	175.87	2.5	5.0
17-03-GG	595.0	474.6	775.1	141.37	1.2	2.8
17-03-SG	1125.1	991.3	1422.4	151.70	2.2	4.9

Table 2. General LP results

the influence of the pricing problems on the total time needed for solving the LP-relaxation by making use of parallel programming.

To investigate the effect of the column deletion and the stabilized column generation, we also ran part of the instances without these enhancements. It could be clearly seen that the time needed to solve the LP relaxation to optimality explodes without the use of column deletion and stabilization. One part responsible for this huge increase in time needed is the large increase in the average time needed for solving one iteration of the RMP. This can be explained by the fact that after a couple of iterations, the model quickly becomes very large due to the fact that all columns stay in the model.

It turns out that without column deletion and stabilized column generation, the average number of iterations needed to solve the LP-relaxation to optimality is higher than when both are enabled, while the average time needed for solving the pricing problems is lower. The increase in number of iterations needed is an example of the so-called tailing-off effect. In the beginning there are big improvements in each iteration, while more and more iterations are needed when closer by the optimum. Using the stabilized column generation has a positive effect on this tailing-off effect, as can be seen by the number of iterations needed.

It turns out that the combination of column deletion and stabilized column generation are responsible for a huge improvement, in our experiments by a factor 2.5 up 19, in the time needed for solving the LP-relaxation to optimality with column generation. Interesting is the fact that the improvement seems larger when the instances are larger (see HS versus LS)

The results for solving the ILP are given in Table 3. The table shows that we were able to solve the very large ILP within a few minutes. In our experiments the integrality gap turned out to be very small.

As mentioned in Section 3 we added additional constraints to the model before solving the actual ILP. These additional constraints can be considered as a kind of rounding-heuristic in the way that if in the optimal solution for the

LP-relaxation a flight is always assigned to a certain type of gate in all selected gate plans, we add a constraint that enforces the flight to be assigned to a gate plan of that type.

The average number of constraints that were added for flights as well as for buses is shown in Table 3. These constraints result in ILP models that are a lot smaller and hence in a much smaller solution time. From our experiments we found that the additional constraints did not have a significant impact on the value of the final ILP solution and did not result in infeasibility of the ILP.

	Average additional constraints		Average solving
Instance	Flight constraints	Trip constraints	time ILP (s)
02-08-GG	121.4	57.6	43.5
02-08-SG	103.4	57.9	54.1
03-08-GG	117.8	57.1	42.0
03-08-SG	105.4	57.7	103.3
04-08-GG	119.3	57.2	82.7
04-08-SG	108.7	57.5	95.2
15-03-GG	108.9	58.4	86.5
15-03-SG	91.0	59.0	271.0
16-03-GG	107.0	59.1	45.8
16-03-SG	84.2	59.3	170.6
17-03-GG	118.5	59.9	20.6
17-03-SG	105.6	59.6	29.5

Table 3. General results ILP

One other way to speed up the process of solving the ILP we used is to first only solve the root node relaxation. We then add a so called cut up limit to the model that is 0.5% above the value of the root node. This cut up limit acts for the ILP solver as if an integral solution with that particular value has already been found, meaning that any node with a relaxation value greater than this cut up value is automatically pruned. Strictly speaking this might result in infeasibility of the ILP (when the optimal ILP solution exceeds the thhreshold), but this never occured in our experiments.

Furthermore, when looking at the time needed for solving the various final ILP models, we can see that these times are still within very acceptable ranges, also for the Single Gate Problems. This indicates that it is feasible to assign flights and trip directly to physical gates and buses respectively.

5 Conclusion and further research

We have investigated the combination of two assignment problems that in practice are solved in a sequential fashion. We formulated the combined problem in one large model for which we approximate the optimal solution by means of an approach based on column generation.

We implemented our algorithm and tested it with real-life data provided by AAS. The results show that our approach is capable of solving these real-life instances within acceptable time, especially given the fact that this approach solves two problems within about the same time that currently is available at AAS for the computer to present a solution for only the gate assignment problem.

We also showed that our approach is still capable of solving the instances within acceptable running times if we create a single gate type for each separate gate, except for the remote stands. This different model lead to over twice the number of gate types which significantly increased the size of the instances.

We are currently performing a simulation study of the platform buses, to evaluate the robustness of the column generation planning compared to a kind of first-come-first-served method as used at AAS. We can clearly see that the column generation schedule is more smooth, in the sense that the idle time is spread more evenly. Currently, the gate assignment at AAS needs a lot of replanning during the day of operation. However, comparing the quality of our resulting schedules to the actual schedules in use at AAS is difficult for a variety of reasons, the main one being the fact it is not possible to retrieve the schedule we would like to compare to, namely the initial schedule as produced by the computer for the upcoming day.

An interesting possibility of further investigation is to start looking at a more operational type of planning. It would be interesting to see how our suggested approach performs if we do not let it create a schedule from scratch but we supply it with a schedule and some disturbances and let the program try to resolve this updated problem.

One of the main things that would have to be considered for this approach is the fact that any new solution should not deviate too much from the currently existing solution. So when solving the problems after some parts are fixed (since they already happened) and other events have changed properties (e.g. earlier or later Estimated Time of Arrivals and Departures) the cost function would not only have to consider the robustness of the schedule, but also the similarity to the original-day-ahead schedule, since too many changes in a schedule will result in a lot of confusion for the different parties dependent on the schedule.

References

1. L. Clarke, C. Hane, E. Johnson, and G. Nemhauser. Maintenance and crew considerations in fleet assignment. *Transportation Science*, 30:249–260, 1996.
2. J.-F. Cordeau, G. Stojkovic, F. Soumis, and J. Desrosiers. Benders decomposition for simultaneous aircraft routing and crew scheduling. *Transportation Science*, 35(4):375–388, 2001.
3. G. Diepen. *Column Generation Algorithms for Machine Scheduling and Integrated Airport Planning*. PhD thesis, Utrecht University, (In preparation), 2008.
4. G. Diepen, J. M. v. d. Akker, J. A. Hoogeveen, and J. W. Smeltink. Using column generation for gate planning at amsterdam airport schiphol. Technical Report UU-CS-2007-018, Institute of Information and Computing Sciences, Utrecht, the Netherlands, 2007.

5. O. du Merle, D. Villeneuve, J. Desrosiers, and P. Hansen. Stabilized column generation. *Discrete Math.*, 194(1-3):229–237, 1999.
6. R. Freling, D. Huisman, and A. P. M. Wagelmans. Models and algorithms for integration of vehicle and crew scheduling. *Journal of Scheduling*, 6(1):63–85, 2003.
7. C. Gao. *Airline Integrated Planning and Operations*. PhD thesis, Georgia Institute of Technology, August 2007.
8. M. Lohatepanont and C. Barnhart. Airline schedule planning: Integrated models and algorithms for schedule design and fleet assignment. *Transportation Science*, 38(1):19–32, 2004.
9. B. Rexing, C. Barnhart, T. Kniker, A. Jarrah, and N. Krishnamurthy. Airline fleet assignment with time windows. *Transportation Science*, 34(1):1–20, 2000.
10. R. Sandhu and D. Klabjan. Integrated airline planning. In *AGIFORS Symposium 2004, Singapore*, 2004.