

Integration of gate assignment and platform bus planning*

G. Diepen[†] J.M. van den Akker[‡] J.A. Hoogeveen[§]

June 9, 2008

Abstract

In this paper we look at the gate assignment problem and the bus planning problem as they appear at Amsterdam Airport Schiphol (AAS). Furthermore, we consider the integrated version of these problems. Given the expected arrivals and departures for the next day, we aim at finding a robust solution, such that the amount of replanning due to deviations from the expected arrival and departure times is minimum. For the separate problems we present two similar integer linear programs (ILP) and solve the LP-relaxations through column generation. The generated columns and a set of additional columns are then used to solve the ILP. For the integrated approach we show how to link the models of the separate problems to form one large model. Computational results with real life data provided by AAS are promising and indicate that the algorithm is able to solve real-life instances within acceptable running times.

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 the passengers need to disembark the aircraft. Moreover, the aircraft needs to be refueled, new passengers need to board it, new supplies have to be put on board, and the aircraft has to get cleaned. These latter two actions are taken care of by the so-called *ground handler*. All of these actions take place while the aircraft is standing at a gate. We will refer to the arrival of an aircraft till the following departure of the same aircraft as a *flight*.

One of the most important problems at the airport is the so-called *gate assignment problem*: we have to find for each flight a gate, where we have to satisfy a number of constraints. Next to the obvious constraint that there should be a minimum time between two consecutive flights at the gate, we have for example

- Gates can handle only flights operated by aircraft of certain sizes.
- Gates can handle only flights for certain origins/destinations (e.g. because of safety regulations).
- Gates can handle only flights assigned to certain ground handlers.
- Two adjacent gates can not be assigned flights operated by big aircraft at the same time.
- Two adjacent gates should not be assigned flights with equal departure times.

*This work was supported by BSIK grant 03018 (BRICKS: Basic Research in Informatics for Creating the Knowledge Society)

[†]Department of Information and Computing Sciences, Utrecht University, diepen@cs.uu.nl

[‡]Department of Information and Computing Sciences, Utrecht University, marjan@cs.uu.nl

[§]Department of Information and Computing Sciences, Utrecht University, slam@cs.uu.nl

Depending on the airport and its characteristics, many variants of the gate assignment problem have been researched and many solution methods have been suggested. A good overview of explored methods and models is given in van Orden (2002). In this paper we consider the situation at Amsterdam Airport Schiphol (AAS). Depending on the time horizon of the planning we distinguish between the following three planning problems: *seasonal planning*, *daily planning*, and *tactical planning*. The seasonal planning problem is a capacity planning problem. In this problem the gate planners must decide to accept or decline new requests from airlines to have their aircraft visit AAS. The daily planning concerns the creation of a planning for the upcoming day on the basis of the available information about the flights of that day. Finally, the tactical planning is an online planning problem. This problem concerns the resolving of conflicts that arise in the planned solution (i.e. the planning created the day before) due to disturbances.

The problem we consider in this paper is daily planning. When looking at this planning problem, different objectives can be taken into account. Examples are minimizing total waiting time for the aircraft (i.e. the time an aircraft has to be held after landing before the gate is free) or minimizing total towing actions. Another objective that is used very often in the literature is to minimize the walking distance for the passengers to and from the gate (Bihr 1990, Haghani and Chen 1998, Xu and Bailey 2001). Moreover, it is also possible to consider multi-criteria objectives. Dorndorf et al. (2007) not only present a state-of-the-art of the gate assignment problem in general but also discuss recent developments with regard to the multi-criteria objectives.

Since an airport is a very dynamic environment, an actual day will hardly ever go completely as planned; flights arrive either earlier or later than planned due to all kinds of reasons. The objective we are concerned with is to create a *robust* schedule, i.e., a schedule that is able to cope with small perturbations in arrival or departure time without the need to replan big parts of the schedule. During meetings we had with the gate planners at AAS, finding a robust schedule was identified as the main target of their daily planning activities. Moreover, every change needed in the gate assignment during the actual day has an effect on a very broad range of parties: the ground handler, the security personnel, the passengers, etc.

Therefore, creating a schedule that needs fewer changes during the tactical planning is of great importance for a variety of parties. The same objective has been considered by Bolat (2000) and van Orden (2002).

The question is how to measure the robustness of a schedule. The probability that a conflict arises between two consecutively scheduled flights at the same gate depends on the ‘reliability’ that the aircraft will stick to their assumed departure and arrival times and on the idle time between these two consecutive flights. The first part is outside the scope of the gate planners, and consequently we focus on optimizing the choice of the pairs of flights that we put consecutively at a gate. One approach for optimizing the individual idle times, and thus finding a robust schedule, is to minimize the variance of the idle time between successive flights at a gate. This approach is used in Bolat (2000) where the problem is formulated as an Integer Linear Program (ILP) using binary decision variables that link flights to positions at a gate. Based on this research a similar approach was followed in van Orden (2002) for modelling the gate assignment problem for AAS. Although the suggested model showed promising results, a weak point of this model was that it was not possible to solve problems with more than 80 aircraft and 20 gates in a reasonable amount of time.

In this paper we present an algorithm to find a robust gate assignment schedule for a full day of traffic at AAS, which has about 600 flights. We include all real constraints occurring at the airport that we have identified in discussions with the gate-planners of the airport. To attain robustness, we optimize the idle time between all consecutive flights at the gates. We formulate this problem as an ILP and use a completely different representation from the one used in van Orden (2002). This different representation is derived from the one used in Freling et al. (2001) for the vehicle and crew scheduling. We present an algorithm based on column generation to find a very good approximation for the optimum in this model. Our experiments indicate that this algorithm is able to solve real-life instances to near-optimality within a few minutes.

After having solved the gate assignment problem, we come to the so-called *bus planning problem*. At AAS there are two kinds of gates: the regular gates, which have a passenger air bridge, and

the remote stands. Passengers on flights that are parked at a remote stand have to be transported by buses. Each flight on a remote stand leads to a number of trips, which have to be driven. Hence, we have to assign buses (and busdrivers) to trips; again, we want to create a schedule that is as robust as possible. This problem largely resembles the gate assignment problem, but there is one major difference: if a busdriver has a shift that lasts longer than 4.5 hours, then he/she is entitled to a break of 45 minutes. We show how we can solve the bus assignment problem using a column generation approach similar to the one used to solve the gate assignment problem. We have implemented our algorithm and tested it with real-life data provided by AAS. Again, we find a solution which is near-optimal in a matter of minutes.

At AAS, the gate assignment problem and bus assignment problem are solved sequentially: the outcome of the gate assignment problem serves as input for the bus assignment problem. With a bit of bad luck, the schedule for the gate assignment results in an input for the bus planning problem that allows poor solutions only, whereas applying some minor changes to the original solution for the gate assignment problem would already allow better assignments for the buses. So although this would mean that a sub-optimal solution for the gate assignment problem would be used, the solution for the gate and bus planning as a whole would improve. Therefore, we have studied the *integrated gate and bus planning problem* as well. Here we solve both problems at once, and we measure the outcome value as a weighted sum of the two outcome values for the separate problems. A major problem here is that we do not know in advance which trips need to be driven, as we do not know which flight are assigned to the remote stands. We show how we can solve the integrated problem by an algorithm based on column generation. Again we have implemented algorithm and tested it on real-life instances provided by AAS. Our algorithm solves these instances within the same amount of computation time as is currently spent at AAS for the computer to present a solution for only the gate assignment problem based on a rule-based optimization approach.

The outline for the remainder of this article is as follows: In Section 2 we give a detailed description of the gate assignment problem. In Section 3 we formulate the gate assignment problem as an ILP and present our algorithm to find an approximate solution. In Section 4 we consider the bus planning problem and in Section 5 we consider the integration of the gate assignment and bus planning problem. In Section 6 we present the experimental results, and finally in Section 7 we draw some conclusions and indicate some future research topics.

2 Problem description gate assignment

As mentioned in the previous section, our objective is to maximize the robustness of a solution to the gate assignment problem. To maximize the robustness we maximize the idle time between all pairs of consecutive flights at a gate, ensuring that each flight can arrive either a bit too early or a bit too late without the need for replanning the schedule. An example of a non-robust schedule can be found in Figure 1. This schedule does not have a lot of margin between flight 2 and flight 3. By modifying the schedule such that flight 3 is assigned to gate 2, while flight 4 gets assigned to gate 1 we introduce a lot more robustness in the solution.

One of the things that must be taken into consideration during the gate assignment process at AAS is whether two consecutive flights assigned to the same gate, are operated by the same airline or share the same ground handler. Such a situation is *convenient*, since the airline and the ground handler respectively will have an incentive to make the first flight leave on time. Furthermore, some airlines are known to be unreliable, meaning that if a flight of such an airline is due to depart at a certain time, then there is a great chance that it is delayed.

There are several hard constraints in the gate assignment problem. Obviously, each flight must be assigned to a gate, and two flights cannot be assigned to the same gate at the same time. But there are many more hard constraints, concerning the properties connected with the flights and the gates. The properties that are known for each flight are:

- The region of the origin of the flight,

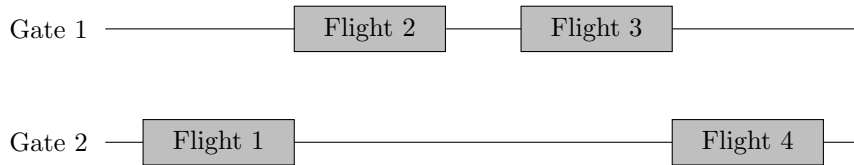


Figure 1: Example of a non-robust schedule.

- The region of the destination of the flight,
- The size category of the flight,
- The ground handler for the flight.

The region can be either Schengen (which refers to the countries that signed the Schengen Agreement), European Union (EU), or Non-EU. Often the region of the origin and the region of the destination of a flight are the same, but there are many exceptions, including e.g. transit flights that do not have AAS as their final destination. With respect to the size category of the flight, there are 8 categories at AAS: category 1 for the smallest aircraft up to category 8 for the biggest aircraft. Finally, the ground handlers are divided into two groups at AAS: KLM Ground Services, and all other companies.

For each of the gates it is known which regions, which size categories, and which ground handlers it can serve. When assigning flights to a certain gate, we need to satisfy the following three essential properties:

- The regions of origin and destination of the flight must match the possible regions of the gate.
- The size category of the flight must match the possible size categories of the gate.
- The ground handler of the flight must match the possible ground handlers of the gate.

One important issue of the gate assignment problem is that some of the flights stay at AAS for a longer period of time; for example, they arrive early in the morning and leave again late in the afternoon. If there are many such *long-stay* flights that stay at the gate, then the number of available gates quickly decreases. To circumvent this problem, the gate planners at AAS have the possibility of *splitting* the stay of long-stay flights into three different parts:

- *Arrival part.* After the aircraft lands, it will stay at the gate for 65 minutes, after which it is towed to some buffer stand.
- *Intermediate part.* During this part the aircraft resides on a buffer stand, where it does not use precious gate capacity.
- *Departure part.* The aircraft is taken from the buffer to the appropriate gate, 95 minutes before the aircraft will depart.

At AAS only flights that stay longer than three hours are considered for such a splitting. The advantages of splitting long-stay flights are three-fold. First, there is the obvious extra capacity that becomes available for the assignment of other aircraft. The second advantage concerns flights with different regions for the origin and destination: such flights normally would have to be assigned to a gate that is multi-purpose with respect to the region property, and these gates are quite scarce. When such a flight is split into parts, the separate parts do not have to be assigned

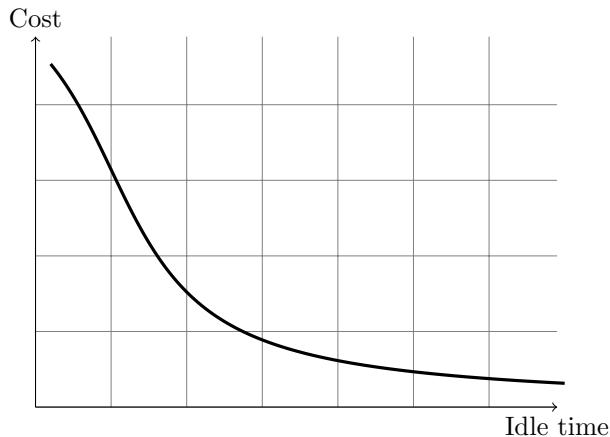


Figure 2: Cost function for valuing idle time.

to the same gate and thus can be assigned to two single region gates. Third, the decoupling of the parts yields additional flexibility.

Currently the process of splitting the long-stay flights is done manually. First the gate planners try to solve the gate assignment problem without splitting any flights. If the available capacity is not sufficient to accommodate all flights, the gate planners determine which flights should be split and then solve the problem again. This step is repeated several times until sufficient capacity is available.

3 Problem formulation

To value the idle time between two consecutive flights we define a cost function. This cost function for the idle time must reflect the natural appreciation of a solution. First of all, it should penalize very small idle-times with very high cost and it should only mildly penalize rather large idle times. Second, the function should be steep in the beginning (for small idle times) and then flatten out, to reflect that for small idle times any improvement is very beneficial, whereas for already large idle times an extra increase is of minor importance. Third, it should be possible to combine this with a refinement reflecting the *reliability* of a flight, which is discussed below. We found that a cost-function based on the arctangent fulfills the desired properties best. This function is defined as follows:

$$c(t) = 1000(\arctan(0.21(5 - t)) + \frac{\pi}{2}),$$

where t is the amount of idle time.

Recall that there could be an advantage (or disadvantage) in assigning certain pairs of flights consecutively to the same gate. To model such a relation we introduce a convenience multiplier $conv(v, w)$ for flights v and w . To compute the cost of placing flight w immediately after flight v at the same gate, the cost corresponding to the idle time in between these flights will be multiplied by this multiplier. If putting flight w after flight v is desirable (e.g. flight v and w are operated by the same airline or handled by the same ground handler), then the convenience multiplier is given a value less than 1, thus decreasing the cost. On the other hand, inconvenient situations (e.g. when flight v is operated by an unreliable airline) can be penalized by giving the multiplier for these situations a value greater than 1, thus increasing the cost of such an assignment.

Based on the approach used by Freling et al. (2001) for the single-depot vehicle scheduling problem, we split the gate assignment problem into two phases. In the first phase we aggregate gates with the same properties (*identical gates*) into groups of gates and each such group we refer to as a *gate type*. We now introduce a *gate plan* as a series of flights that are to be assigned to

the same gate of a certain gate type. Gate plans enable us to check feasibility easily: all flights present in the gate plan must satisfy the properties of the gate type that the gate plan corresponds to and no two flights should be assigned at the same time. Furthermore, calculating the cost of a gate plan boils down to summing the cost of all idle times between consecutive flights within the gate plan. With this representation solving the first phase comes down to finding a set of gate plans such that we have a gate plan for each physical gate, and such that each flight is present in exactly one of the gate plans.

After we have obtained a solution for the first phase, for each gate type we have as many gate plans as there are gates of that gate type. In the second phase we undo the aggregation of identical gates into groups of gate types and we assign each gate plan to a physical gate.

3.1 Assigning flights to gate plans

When we look at the number of possible gate plans, we can clearly see that the number of possible gate plans is enormous. Suppose for the moment we do have the complete set of all possible gate plans, the gate assignment problem can be formulated as determining which of these gate plans we must select. Now for each gate plan i we introduce a decision variable x_i as follows:

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

Let V denote the number of flights, A the number of gate types, S_a the number of gates of type a , and N the number of gate plans. Now the basic model for the gate assignment problem is as follows:

$$\text{Minimize } \sum_{i=1}^N c_i x_i$$

subject to:

$$\sum_{i=1}^N g_{vi} x_i = 1 \quad \text{for } v = 1, \dots, V \quad (1)$$

$$\sum_{i=1}^N e_{ia} x_i = S_a \quad \text{for } a = 1, \dots, A \quad (2)$$

$$x_i \in \{0, 1\} \quad \text{for } i = 1, \dots, N \quad (3)$$

where

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

$$e_{ia} = \begin{cases} 1 & \text{if gate plan } i \text{ is of type } a; \\ 0 & \text{otherwise.} \end{cases}$$

We will extend this basic model to cover more possibilities. One important issue not addressed by the above model is the fact that it should be possible to give *preference* to placing a flight at a certain gate type. Such preferences can be used to model that certain flights are preferably assigned to certain gate types, e.g. because of the size of the waiting area or because of airlines having their “own” gates where at least a certain percentage of their flights have to be assigned to. Each preference consists of a set of flights, a set of gate types, and the minimum number of flights out of the given set that should be assigned to the given set of gate types.

Preferences can be modelled in the ILP by adding the following constraints to the model:

$$\sum_{i=1}^N \sum_{v=1}^V \sum_{a=1}^A p_{vak} e_{ia} g_{vi} x_i \geq P_k \quad \text{for } k = 1, \dots, K \quad (4)$$

where

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

P_k denotes the minimum number of flights that have to be assigned to a given gate type according to preference k , e.g. an airline must have at least 6 flights at their “own” gates. K denotes the total number of preferences.

We extend the above model to deal with the case that there is not enough capacity to accommodate all flights. To solve this problem we add a penalty variable UAF_v (unassigned flight) for every flight v to constraint (1) in the following way

$$\begin{aligned} \sum_{i=1}^N g_{vi}x_i + \text{UAF}_v &= 1 \quad \text{for } v = 1, \dots, V \\ \text{UAF}_v &\geq 0 \quad \text{for } v = 1, \dots, V. \end{aligned} \tag{5}$$

The extra variable UAF_v for flight v is added with a very high cost coefficient Q_v in the objective function. Now it is possible to have flights not being assigned to gates, but this option comes at a high price. The unassigned flights present in the final solution are then assigned to gates manually by the gate planners, who have the ability to overrule some of the constraints, if necessary. We can select the cost coefficients of the UAF_v variables to model the effort it takes to assign a flight manually; for example, in general it is easier to assign a flight operated by a small aircraft somewhere manually than a flight operated by a big aircraft. The values for these cost coefficients were determined via preliminary computations.

One thing not addressed in the model yet is the fact that flights with a long stay can be split into three parts of which two parts must be assigned to a gate. To model this possibility, for each long-stay flight v we create two *split flights* v_A and v_B , which refer to the arrival and departure part of flight v , respectively. The intermediate parts of the flights are not modelled because the buffer stands for these intermediate parts are not part of the gate assignment problem.

Since these three flights v , v_A , and v_B concern the same aircraft, we must ensure we model their dependency. This can be achieved by splitting the original constraint (1) for flight v into two separate constraints

$$\sum_{i=1}^N (g_{vi} + g_{v_A,i})x_i + \text{UAF}_{v_A} = 1 \quad \text{and} \quad \sum_{i=1}^N (g_{vi} + g_{v_B,i})x_i + \text{UAF}_{v_B} = 1.$$

Here UAF_{v_A} and UAF_{v_B} indicate the possibility of not assigning (a part of) flight v ; their cost coefficients each get a value half of the value of the original cost coefficient UAF_v .

Furthermore, we have to include the split flights in the preference constraints (4). Since each split flight only represents half of the original flight, each gets a coefficient 0.5 by redefining p_{vak} as follows

$$p_{vak} = \begin{cases} 1 & \text{if flight } v \text{ has preference on gate type } a \text{ in preference } k; \\ 0.5 & \text{if the unsplit version of flight } v \text{ has preference on gate type } a \text{ in} \\ & \text{preference } k; \\ 0 & \text{otherwise} \end{cases}$$

The ILP formulation presented above models the problem correctly, but unfortunately the size of the problem is enormous, since the number of possible gate plans is enormous. Therefore we try to approximate the optimal solution by taking only *presumably useful* gate plans into account. To identify these, we first relax the integrality constraints (3) and then solve the resulting LP-relaxation by column generation. We then reinstate the integrality constraints and solve the resulting ILP formulation with the columns generated. As a side-effect, we can use the value of the LP-relaxation as a measure for the quality of the obtained ILP formulation.

3.2 Pricing problem

After we have solved the LP-relaxation for a given set of columns, we find a dual multiplier π_v for the constraint (1) corresponding to flight v , a dual multiplier λ_a for the constraint (2) corresponding to type a , and a dual multiplier ψ_k for the constraint (4) corresponding to preference k . Therefore, the reduced cost for a gate plan i is equal to

$$c_i - \sum_{a=1}^A e_{ia} \lambda_a - \sum_{v=1}^V (g_{vi} \pi_v + \sum_{k=1}^K \sum_{a=1}^A g_{vi} e_{ia} p_{vak} \psi_k).$$

Note that for the original parts of a long-stay flight the above must be slightly changed. Since the original part of a long-stay flight is present in two constraints, we must subtract the two dual multipliers π_{v_A} and π_{v_B} instead of only π_v .

It is well-known from the theory of linear programming that we have solved the LP-relaxation to optimality if the reduced cost of each gate plan is greater than or equal to zero. To check this, we compute the minimum reduced cost over all feasible gate plans; this is called the *pricing problem*. We solve this problem by composing a network such that each feasible gate plan corresponds to a path in this network, and vice-versa. Moreover, we choose the lengths of the arcs such that the length of a path equals the reduced cost of the corresponding gate plan. Hence, we can then solve the pricing problem by solving a shortest-path problem in this network.

We solve the pricing problem for each gate type separately. For each type of gate a we introduce a Directed Acyclic Graph (DAG) $G_a = (V_a, E_a)$. We add a vertex to this graph for every flight v that is allowed to be assigned to a gate of type a . Furthermore, we add vertices s and t , denoting the source and sink respectively. If two flights v and w are allowed on a gate of type a and the arrival time T_w^{arr} of flight w is greater than or equal to the departure time T_v^{dep} of flight v plus the minimum idle time T_v^{min} required after flight v , then a directed edge from vertex v to w is added to the graph. Furthermore, a directed edge from the source vertex s to every vertex v is added, as well as a directed edge from every vertex v to the sink vertex t . Hence,

$$E_a = \{(v, w) | T_w^{\text{arr}} \geq T_v^{\text{dep}} + T_v^{\text{min}}\} \cup \{(s, v), (v, t) | \text{for all } v\}.$$

It can be easily seen that every path from s to t in G_a represents a feasible gate plan of type a and vice-versa. What is left is to set the lengths of the arcs. If we look at the reduced cost, then we see that, if a flight v is selected and succeeded by flight w in a gate plan of type a , then the contribution of flight v to the reduced cost of the gate plan is

$$\text{conv}(v, w) c(T_w^{\text{arr}} - T_v^{\text{dep}}) - \pi_v - \sum_{k=1}^K p_{vak} \psi_k.$$

Recall that $\text{conv}(v, w)$ denotes the convenience multiplier indicating the advantage (or disadvantage) of putting flight v directly before w . We put the cost of the arc (v, w) in G_a equal to the contribution of flight v to the reduced cost. The additional arcs (s, v) and (s, t) get cost $-\lambda_a$, and the additional arcs (v, t) get cost equal to $-\pi_v - \sum_{k=1}^K p_{vak} \psi_k$. Note that the cost $c(T_w^{\text{dep}} - T_v^{\text{arr}})$ of putting flight w immediately after flight v in the gate plan is constant; after each iteration, we only have to update the cost terms containing the dual multipliers.

Since exactly one of the outgoing edges of vertex v will be used if v occurs in a path, the total cost of a path equals the reduced cost of the corresponding gate plan.

For solving the pricing problem we need to find the gate plan with minimal reduced cost. Since each path in the graph corresponds to a possible gate plan and the path length corresponds to the reduced cost of the represented gate plan, finding the gate plan with minimal reduced cost comes down to finding the shortest path in the presented graph. Without loss of generality we assume that all flights are sorted by their arrival times. This assumption implies a topological order on the vertices in the graph, namely the order of the flight indices. Because we now have a DAG with a topological order it is possible to find the shortest path in $\mathcal{O}(|V| + |E|)$ time (cf. Cormen et al. (2001)).

When a gate plan with minimum reduced cost has been found, there are two possibilities:

- The new gate plan has *negative reduced cost*. This means that by adding this gate plan to the master problem, the objective value of the master problem might decrease and thus we add this gate plan to the master problem.
- The gate plan has *zero reduced cost* in which case there exists no gate plan with negative reduced cost.

For each of the gate types, we need to check whether a gate plan with negative reduced cost exists. If for none of the gate types a gate plan with negative reduced cost exists, then the master problem has been solved to optimality.

3.3 Solving the restricted ILP

After the master problem has been solved to optimality we only have a solution for the LP-relaxation. If this solution happens to be integral, then we have a solution to the original ILP formulation of the gate assignment problem, too. If the solution is fractional, then we have to convert it to an integer solution. In our case it is not necessary to obtain the exact optimum, but a good approximation will suffice. To this end, we use the ILP-solver CPLEX to solve the ILP with the limited set of columns. In our preliminary computations, we only used the restricted set of columns generated by the column generation. It turned out that this took too much time and memory. Moreover, if solutions were found within reasonable running time, then the quality of these solutions was really bad. The large running time and memory consumption can be explained by the fact that the restricted set of columns generated for solving the LP is too restrictive for the ILP: if flight v is part of a selected column, then none of the other columns containing v can be used anymore. Hence, if our *first-choice* column contains a flight that is included in one of the already selected columns, then we need a *second choice* column that does not contain this already covered flight. As the first-choice column was generated once, when solving a pricing problem, we decided to create a set of additional second-choice columns each time when solving the pricing problem.

To create these second-choice columns we used the following procedure. We first solve the pricing problem, that is, we find the shortest path. We then take out the nodes in the shortest path one by one and solve the shortest path problems for each of the resulting graphs. These additional columns are added to a column pool. After the master LP problem has been solved to optimality, we determine the set of unique columns from the ones that are in the column pool. This set of unique columns is then added to the restricted ILP problem. After these unique columns were added to the ILP problem, the ILP solver was able to solve the problem in a matter of minutes and sometimes even seconds instead of running for hours or even days.

3.4 Assigning gate plans to gates

After solving the first phase, we have a set of gate plans with just as many gate plans of type a as there are physical gates of type a , such that the total schedule is robust against small variations caused by for example delays of flights. In the second phase of the problem we have to determine which gate plan is assigned to which physical gate.

In the first phase we have introduced constraints dealing with just one gate type. We did not consider relations between specific physical gates, like the constraint that two flights having the same departure time can not be assigned to directly opposite gates due to the impossibility of a simultaneous push-back of both flights. We consider these types of constraints when we assign the gate plans to the physical gates in the second phase.

In van Orden (2002) some additional constraints have been formulated that need to be addressed in the second phase. These are:

- Avoid putting two flights next to each other that have an overlapping wingspan.
- Avoid putting two flights with equal departure time next to each other because of conflicting push-back.

- Avoid putting two flights that have the same departure time on opposite gates because they cannot have a push-back at the same time.
- Flights from US carriers need extra facilities (e.g. possibility of closing parts of the waiting room at the gates to which they are assigned).
- Minimize the walking distance for the passengers. This concerns both arriving or departing passengers and transfer passengers.

Although the first one of these constraints at first sight seems to be a good example of a second phase constraint, it turned out to be not important at all. After receiving detailed information of the gates at AAS from the gate planners it turned out that this constraint did not exist for any gate at AAS. But there does exist a strongly related property at AAS though, which decrees that it is possible to combine two gates of a small category to one gate of a bigger category. This constraint cannot be addressed in the first phase, since we do not know then which two gate plans will be next to each other in the final solution. In theory it is possible to take this constraint into consideration when solving the first phase by creating a new category of gates consisting of the two smaller gates. The pricing problem for this gate type would then consist of two paths that both contain the selected vertices corresponding to the aircraft of a bigger category. This problem is harder to solve. Therefore, and also since there are not so many of these possibilities, we have decided to ignore the possibility of combining the gates.

After completing the second phase it will be known which gate plans and thus which flights will be assigned to the gates that can be combined. When there are big flights left that are still not assigned, the gate planner will be able to manually combine a set of these smaller gates into one gate of a bigger category and assign a bigger flight to this combined gate.

Although the gate planners at AAS do not have information regarding possible connecting flights of the passengers, one way they try to maximize passenger comfort is to minimize the maximum walking distance. This is achieved by putting flights with a large number of passengers at the best gates. Since we are assigning entire gate plans to gates now, we have less flexibility, but we can use the same principle. Gates that are closer to the beginning of the pier are considered to be better gates. So also in this case the number of passengers will be an important factor in the decision: when there are more gate plans to choose from, the one with the largest number of departing passengers will be assigned to the best gate.

The full assignment problem of the second phase can be decomposed into a number of smaller assignment problems for each type of gate. Most of these subproblems can be solved independently, but some are dependent, since they involve gates that have a neighboring or opposite gate of a different type. The dependent subproblems need more attention. To determine the benefit of assigning a flight to a certain physical gate, the gate planners at AAS use a number of different rules.

Presumably, the best option is to present the gate planners with the results from the first phase and have them assign the gate plans to the physical gates. This can be done *manually* since the size of these problems is rather small; generally the maximum number of gates within one gate type is around eight. Only for remote stands this number is higher, but the flexibility for assigning gate plans to these remote stands is really high. Finally, sometimes it may turn out to be beneficial to swap two flights from two gate plans, resulting in a better solution.

3.5 Directly assigning flights to gates

The two phases of first assigning flights to gate plans and then assigning the gate plans to gates can be integrated by modelling each single physical gate as a separate type and including all the constraints concerning specific physical gates.

As a first step, we defined in our original ILP formulation all the gates except for the remote stands as separate types. The reason we do not consider each remote stand as a separate type is that there does not exist any significant difference between these gates (e.g. they do not have waiting rooms and they all require a bus to transport passengers to and from the terminal

building). Our computational experiments reveal that considering all gates as separate types is still computationally feasible. Furthermore, it decreases the amount of work in the second phase, because it limits the second phase to swapping of some gate plans between physical gates and if necessary including unassigned flights and swapping flights between gate plans.

However, the model for assigning flights to gate plans does not yet include all constraints for the case we have a separate type for each gate. The constraint that flights from the United States need extra facilities can directly be included in the definition of which flight is allowed on which type of gates. For minimizing the maximum walking distance we would need to adapt the objective function or introduce preference constraints. Moreover, preventing simultaneous push backs would also require specific constraints. To find out which constraints should preferably be included in the model and which constraints can better be handled manually by the gate planners, is a matter of further research. Presumably, there will always be need for some kind of second phase especially if it is necessary to manually violate some of the constraints to accommodate all flights.

4 The bus planning problem

In this section we use the solution of the gate assignment problem as input. Hence, we know which flights have been assigned to which remote stand. Given the number of passengers on the flight, we can construct the set of trips that needs to be driven. For this transport, there are two types of buses available at AAS: the *peak bus* and the *Cobus*. The peak buses are standard buses that are also used for city trips in public transportation. Peak buses are capable of transporting up to 50 persons at once. The Cobus is a bus that is specifically designed for use at airports. Compared to the peak bus it has several advantages, a clear one is that it is larger and can transport up to 70 persons at once. Other advantages are that the bus has a lower floor (thus giving easy access) and it has doors on both sides. This latter advantage is particularly useful, since it will not matter from which direction an aircraft at the platform is approached.

Every season the planners at AAS look at the planned flights and estimate the number of buses needed in each 15 minute time interval of a day for the complete upcoming season. This information is then sent to the Haagse Tram Maatschappij (HTM) bus company and they will create *shifts* for buses and drivers such that they fulfill the capacity requirements for each of the 15 minute intervals. The shifts have four attributes, namely a *starting time*, an *ending time*, a *type of bus* that drives the shift, and the *number* of buses within the shift. Each bus within a shift is driven by one driver.

The planners at AAS do not have a direct influence on these shifts and they must use the shift information as input for their planning. Looking at the type of bus driving the shifts, one can see that the majority of the shifts at AAS are driven by the Cobuses.

Our goal is to find a planning for the buses that is as *robust* as possible. We use the same function to measure the robustness as we did for the gate assignment problem. From discussions with the planners at AAS, we learned that in the case of a departing flight, there is a preference to let the first trip to the aircraft be driven by a Cobus. This is not a mandatory rule, but in case of choice the planners prefer this. To model this, we use an additional penalty cost in case the first trip to a departing aircraft is driven by a peak bus.

To solve this problem, we use a column generation approach. We create *bus plans*, which consist of a set of trips that will be driven by one bus; bus plans correspond to shifts. For a given bus plan to be feasible we require that:

- all trips present in the bus plan can be realized within the start and end times of the associated shift;
- no two trips are conflicting (i.e. there must be enough time to drive from the destination of one trip to the origin of the next trip).

For the shifts that last longer than 4.5 hours (so-called *long shifts*), we need an extra constraint on the set of trips being driven. Whenever a driver has a shift that lasts more than 4.5 hours, he

is required by law to have a 45 minute break that does not start within 1.5 hours of the shift start time and does not end within 1.5 hours of the shift end time. So somewhere around the middle of the shift, a 45 minute period of time should be reserved for a break. These 45 minutes do not include the time it takes the driver to drive from the last trip before the break to the canteen and from the canteen to the start of the first trip after the break.

The bus planning problem can be formulated as an ILP using a similar formulation as used for the gate assignment problem. Again, our approach is to solve this ILP for a limited number of bus plans, which are determined by solving the LP-relaxation through column generation. Moreover, we generate many more bus plans to serve as second-choice columns in the same way as we did in the gate assignment problem.

The pricing problem boils again down to solving a shortest path problem in a directed acyclic graph; here the trips correspond to the vertices, and there is an arc between two vertices when the trips can be driven consecutively. Since a shift longer than 4.5 hours requires a mandatory break, we add *break vertices* to the graphs corresponding to shifts which last longer than 4.5 hours in the following way:

- Add a break vertex between two trips which are far away in time, such that there is time for a break of at least 45 minutes and the driving time to and from the break location,
- Add a break vertex between the source vertex and any vertex of which the corresponding trip has a start time that is at least 135 minutes (i.e. 90 minutes buffer and 45 minutes break) later than the start time of the shift. This indicates a driver starts his shift and does nothing and then starts his break,
- Add a break vertex between any vertex of which the corresponding trip has an end time that is least 135 minutes earlier than the end time of the shift and the sink vertex. This indicates a drivers has his break and after that does not drive any trips anymore for the duration of his shift.

Finding the bus plan with minimum reduced cost for a given shift b now comes down to finding the shortest source-to-sink path in the graph G_b . For the shifts that have a duration longer than 4.5 hours we must include one mandatory break. We solve this by solving a dynamic programming problem with two state variables per vertex: one stores the cost of the minimum cost path to this vertex, whereas the second one stores the cost of the minimum cost path with a break to this vertex. Furthermore, we also need to keep two references to the predecessors: one for the case a break has been taken already and one for the case no break has been taken.

5 The integrated problem

Current practice at AAS is to use a hierarchical approach: first the gate assignment problem is solved and then the bus planning problem with the outcome of the gate assignment problem as input. It is well-known that an integrated approach usually gives better results. Especially in the airline industry, there are many examples of integration, like integrating the planning of the aircraft routing problem with the crew scheduling problem Cordeau et al. (2001), integrating the schedule design problem and the fleet assignment problem (Rexing et al. 2000, Lohatepanont and Barnhart 2004), and the integration of the fleet assignment and the crew scheduling problems (Gao 2007, Clarke et al. 1996, Sandhu and Klabjan 2004).

We apply an integrated approach for solving the gate assignment problem and the bus planning problem by combining the two problems into one big problem and solving this one big problem as a whole. As the cost function we simply use the sum of the two cost functions, but the one for the gate assignment problem is counted 20 times as much to indicate that this problem is more important.

The model we will use consists of the combination of the separate models presented to solve the gate assignment and the bus planning problems, together with a set of constraints linking the two models. These linking constraints see to it that the trips that have to be driven by the buses

correspond to flights that are put on the platform. We generate all possible trips by considering each flight on each remote stand. We determine which of these will be needed in a solution and which not; for this purpose we use the binary variables NNT_t (not needed trip) for each trip t : if this gets value 1, then the trip does not need to be driven.

In our ILP formulation we use gate plans and bus plans respectively. For each gate plan i we have a binary variable x_i defined as follows:

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

and for each bus plan j we have a binary variable y_j defined as follows:

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

Furthermore, we denote the cost of gate plan i with c_i^G . This cost is the sum of the cost of the idle times between all consecutively assigned flights v and v' within the gate plan, multiplied by the *convenience-multiplier* that measures the preference of this pair of consecutively assigned flights. Similarly, c_j^B denotes the cost of bus plan j and is equal to the sum of the cost of all idle times between consecutively driven trips.

To deal with possible shortage of capacity we allow flights to be not assigned to gates and trips to be not assigned to buses by introducing the penalty variables UAF_v for each flight v and UAT_t for each trip t respectively. To ensure not all flights and trips are left unassigned, these penalty variables have a very high cost Q_v and R_t for flight v and trip t respectively in the objective function.

Furthermore, V denotes the number of flights, T denotes the number of trips, A denotes the number of gate types, S_a denotes the number of gates of gate type a , K denotes the number of preferences, P_k denotes the minimum number of flights that that to be assigned to a set of gate types in preference k , B denotes the number of shifts, T_b denotes the number of buses in shift b , and S_t denotes the number of times trip t needs to be covered.

Now the integrated model is as follows:

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

subject to:

$$\text{UAF}_v + \sum_{i=1}^N g_{vi} x_i = 1 \quad , \text{ for each } v = 1, \dots, V \quad (6)$$

$$\sum_{i=1}^N e_{ia} x_i \leq S_a \quad , \text{ for each } a = 1, \dots, A \quad (7)$$

$$\sum_{i=1}^N \sum_{v=1}^V \sum_{a=1}^A p_{vak} e_{ia} g_{vi} x_i \geq P_k \quad , \text{ for each } k = 1, \dots, K \quad (8)$$

$$\sum_{j=1}^M f_{jb} y_j \leq T_b \quad , \text{ for each } b = 1, \dots, B \quad (9)$$

$$S_t \cdot \text{NNT}_t + \text{UAT}_t + \sum_{j=1}^M h_{tj} y_j = S_t \quad , \text{ for each } t = 1, \dots, T \quad (10)$$

$$\text{NNT}_t + \sum_{i=1}^N \sum_{v=1}^V t_{tv} g_{vi} r_i x_i = 1 \quad , \text{ for each } t = 1, \dots, T \quad (11)$$

$$x_i \in \{0, 1\} \text{ , for each } i = 1, \dots, N \quad (12)$$

$$y_j \in \{0, 1\} \text{ , for each } j = 1, \dots, M \quad (13)$$

$$\text{UAF}_v \geq 0 \text{ , for each } v = 1, \dots, V \quad (14)$$

$$\text{NNT}_t, \text{UAT}_t \geq 0 \text{ , for each } t = 1, \dots, T \quad (15)$$

where

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

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

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

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

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

$$t_{tv} = \begin{cases} 1 & \text{if trip } t \text{ corresponds to flight } v \\ 0 & \text{otherwise,} \end{cases}$$

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

Constraints (6)-(8) are taken from the gate assignment problem we presented earlier. Constraint (9) ensures that for each bus shift, we select at most the number of buses present in that shift. Constraint (10) ensures that a trip is 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; here S_t denotes the number of times a trip t needs to be assigned, which is > 1 in case of the super-trips.

Without any further constraints on the NNT_t variables, the easiest solution for the model 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 (11) 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 two separate models into one large model.

We apply the same solution approach again: we solve this ILP for a restricted set of gate plans and bus plans, which are generated by solving the LP-relaxation by column generation and by adding second-choice columns.

The pricing problem with regards to the bus problem is exactly the same as outlined for solving only the bus planning problem. The only difference is that the size of the graphs for each shift b is larger due to the increased number of trips.

When looking at the pricing problem that needs to be solved for the gate assignment part we have to make a small modification to the original pricing problem, because we have to cope with the new Constraint (11). For each trip t this constraint gives a dual multiplier ρ_t . Since this additional dual multiplier will only be applicable to gate plans that are for remote stands (because only then $r_i = 1$) we can modify the cost on the outgoing arcs of a vertex corresponding to flight v to a successor flight w in the graphs corresponding to gate types that are remote stands as follows:

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

Here, $\text{conv}(v, w)$ denotes the convenience multiplier and measures the preference value of flight w succeeding flight v on a gate. Furthermore, the dual multipliers π_v for flight v and ψ_k for preference k correspond to Constraint (6) and Constraint (8) respectively.

Instance	Flights	Gate types	Total gates
HS-1-GG	699	40	128
HS-2-GG	680	40	128
HS-3-GG	688	40	128
LS-1-GG	602	40	128
LS-2-GG	608	40	128
LS-3-GG	593	40	128
HS-1-SG	699	94	128
HS-2-SG	680	94	128
HS-3-SG	688	94	128
LS-1-SG	602	94	128
LS-2-SG	608	94	128
LS-3-SG	593	94	128

Table 1: Sizes of the provided instances. LS is low season, HS is high season.

One major problem we faced when solving the LP-relaxation was the massive degeneracy. 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 counter this problem of degeneracy. 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. This removal does not only remove some degeneracy, but also the resulting models are smaller and therefore can be solved more quickly. The removed columns are put in a repository and are added back to the problem when we solve the resulting ILP.

The second approach we implemented is *stabilized column generation*. This has been introduced by du Merle et al. (1999) for linear programming. It is based on adding surplus and slack variables to overcome degeneracy by perturbing the right-hand side. We refer to du Merle et al. (1999) for more details.

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 we check in which of the selected gate plans it is contained. If all of these gate plans are of the same type, then we add the constraint that the plane must be assigned to a gate of this type, which implies that we know whether it will require a bus or not. Similarly, we check for each bus in which of the selected bus plans it is contained; if these all are of the same type, then we formulate this as a hard constraint.

Although these constraints might cause the optimal integral solution 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.

6 Experimental results

We have implemented model and its solution in C++ and performed extensive computational experiments. All of the tests were conducted on a Pentium 4, 2.8 GHz with 1GB of RAM. For solving the LP problems and for solving the resulting ILP problem by branch-and-bound the Concert Technology interface of CPLEX 9.1.2 (ILOG 2005) was used.

AAS has provided us with six different sets of data, three of which contain the flights on busy high season days (HS), and three on low season days (LS). From each data set we derived two instances, one instance where we group gates with equal properties into a type and one instance where, except for the remote stands, each individual gate forms a separate type. The sizes of the different data sets are given in Table 1. For a data set X, instance X-GG denotes the instance

Instance	ILP default (s)	ILP enhanced (s)
HS-1-GG	7	6
HS-2-GG	18	9
HS-3-GG	9	8
LS-1-GG	118*	52*
LS-2-GG	101	24
LS-3-GG	58	24
HS-1-SG	21	19
HS-2-SG	48	24
HS-3-SG	26	15
LS-1-SG	168	120
LS-2-SG	73 [§]	126
LS-3-SG	94	113

* Take out freq 40 resulted in one unsolvable LP.

[§] Default CPLEX was not able to solve 3 instances within 60 minutes.

Table 2: Running times for solving the ILP with and without enhancements, averaged over three different values for the take-out frequency.

with grouped gates and X-SG the instance with a gate type for each individual gate. This results in a total of 12 instances for the gate assignment problem.

The use of column deletion has a significant impact on the running times of the column generation procedure. After preliminary experiments the threshold for removal was set by taking the average reduced cost of the columns added in the previous iteration and multiply this average with -0.75 . Furthermore the frequency of taking out columns with reduced cost above the threshold is set to every 80 iterations.

To decrease the size of the branch-and-bound tree we *gamble* for a small integrality gap. In CPLEX we manually initialize an upper bound on the best integral solution at a value of 0.35 percent above the optimal value of the LP-relaxation, which implies that all nodes with a larger lower bound are pruned. This turns out to significantly decrease the solution time. Moreover, we experimented with different settings of CPLEX, such as more elaborate preprocessing and more aggressive cut generation. In Table 2 we show the running times for solving the ILP for the default settings of CPLEX and for the enhanced settings. For each instance we give the average running time over the different take-out strategies. In the enhanced settings, after 90 seconds, the solution process is aborted and more aggressive parameters settings are used for CPLEX, resulting in more time spent on preprocessing the problem which led to smaller running times for solving the restricted ILP. Moreover, using the enhanced settings we were able to solve more instances.

We will refer to the combination of using a take-out frequency of 80 and the above enhancements for the ILP as the *best variant*.

In Table 3 we present the running times needed for the different steps of solving the gate assignment problem. Here, the column convert denotes the time needed to determine all unique columns in the column pool, add these columns to the model, and convert the model to an ILP. It can be seen that even the Single-Gates instances can be solved in just a matter of minutes.

Our results indicate that the integrality gap is very small. This implies that our method is able to find *practically optimal solutions* for real-life instances in about six minutes. Moreover, the results indicate it is feasible to consider single gates as a group. Although this does not make the second phase unnecessary, it makes it easier.

Results of the integrated problem

To create a sufficiently large test set, we combined each of the 12 gate assignment instances with each of the 30 bus planning problem instances that were supplied by AAS. Since the information provided regarding the buses considers all days of the week, while the flight information only

Instance	LP (s)	Convert (s)	ILP (s)	Total (s)
HS-1-GG	139	13	6	160
HS-2-GG	128	12	6	148
HS-3-GG	118	13	5	138
LS-1-GG	61	8	21	91
LS-2-GG	67	8	23	100
LS-3-GG	67	8	5	81
HS-1-SG	310	30	12	355
HS-2-SG	257	28	13	300
HS-3-SG	236	28	13	279
LS-1-SG	160	16	161	340
LS-2-SG	146	18	21	187
LS-3-SG	122	17	196	339

Table 3: Time needed for the different phases for best variant.

Instance	Total time LP (s)			Avg iter	Avg time (s)/iter	
	Average	Min	Max		RMP	Pricing
HS-01-GG	1129.6	967.8	1472.0	161.67	2.8	3.9
HS-01-SG	2070.1	1752.1	2657.7	171.90	4.8	6.8
HS-02-GG	973.9	864.7	1213.2	148.27	2.6	3.7
HS-02-SG	1847.4	1627.4	2337.8	163.07	4.4	6.5
HS-03-GG	1142.6	1010.4	1641.3	157.50	3.2	4.0
HS-03-SG	2575.2	2189.9	3970.3	212.77	4.6	7.2
LS-01-GG	658.5	560.3	769.3	165.17	1.1	2.7
LS-01-SG	1235.8	1094.6	1472.0	175.17	1.9	4.8
LS-02-GG	710.0	623.8	850.4	161.90	1.3	2.8
LS-02-SG	1383.4	1144.0	1661.5	175.87	2.5	5.0
LS-03-GG	595.0	474.6	775.1	141.37	1.2	2.8
LS-03-SG	1125.1	991.3	1422.4	151.70	2.2	4.9

Table 4: General results of solving the LP relaxation of the integrated problem.

consisted of three consecutive days of the week, this combination results in considering each gate assignment instance with different availability of bus capacity.

In Table 4 we present the results of solving the integrated problem. It can be seen that the maximum time needed for solving the LP is about 67 minutes. Though this might look as a lot, we must consider the fact that this is for the Single-Gates instances. For the Grouped-Gates instances the running times are a lot lower. Furthermore, all pricing problems that need to be solved each iteration can be solved in parallel, which would yield a significant decrease in solution times. Finally, the maximum time needed for solving the resulting ILPs is about four minutes.

7 Conclusion and further research

We have investigated the gate assignment problem at AAS and developed a two phase solution approach. In the first phase we assign the flights to gate plans, while in the second phase we assign the gate plans to the actual gates. It turns out that the second phase boils down to a set of small problems that can be solved manually. For the first phase, we have presented a different way of formulating the gate assignment problem as an ILP. Our model includes all realistic constraints that are actually used for solving the gate assignment problem at AAS. Furthermore, we have described a method to find an approximate solution for this new ILP formulation. We have implemented it in C++, where we use CPLEX 9.1.2 for solving the (I)LPs.

Our implementation was tested with real life input data, provided by AAS. These instances

could be solved in a matter of minutes to near-optimality and sometimes even to optimality. The planning software currently in use at AAS is based on a greedy algorithm that assigns flights to gates on the basis of an optimal point score per flight. This score includes different issues, such as preferences of airlines and ground handlers, but it does not contain any robustness measure. Our advanced LP-based algorithm and the planning software of AAS using a greedy algorithm have comparable running times. Experiences from the gate planners at AAS reveal that currently a considerable amount of time (a few hours) has to be spent on replanning the computer generated schedule in order to make it more robust, because the current planning software does not consider robustness at all. Since our algorithm focuses explicitly on robustness, the gate planners have more time for solving the conflicts that arise during the actual day.

Besides implementing the method for solving the gate assignment problem, we also implemented the method for solving the bus planning problem and the integrated problem. Our experiments indicate that the integrated problem can be solved within acceptable running times.

We have not compared the results obtained through the integrated approach to the results by the sequential approach. To make a meaningful comparison, we need to simulate the gate and bus assignment for an entire day to find out about the number of and impact of the necessary replannings. Similarly, we have not compared our plans to the plans used by AAS, since it is not possible for us to retrieve the schedule we would like to compare to, namely the initial schedule as produced by the computer for the upcoming day. We are currently performing a simulation study of the platform buses to evaluate the robustness of the column generation planning compared to a kind first-come-first-serve 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.

References

- Bihl, R.A. (1990). A conceptual solution to the aircraft gate assignment problem using 0,1 linear programming, *Proceedings of the 12th Annual Conference on Computers and Industrial Engineering*, Pergamon Press, Inc., Elmsford, NY, USA, pp. 280–284.
- Bolat, A. (2000). Procedures for providing robust gate assignments for arriving aircrafts, *European Journal of Operational Research* **120**: 63–80.
- Clarke, L., Hane, C., Johnson, E., and Nemhauser, G. (1996). Maintenance and crew considerations in fleet assignment, *Transportation Science* **30**: 249–260.
- Cordeau, J.F., Stojkovic, G., Soumis, F., and Desrosiers, J. (2001). Benders decomposition for simultaneous aircraft routing and crew scheduling, *Transportation Science* **35**(4): 375–388.
- Cormen, T.H., Leiserson, C.E., Rivest, R.L., and Stein, C. (2001). *Introduction to Algorithms, Second Edition*, The MIT Press/McGraw Hill.
- Dorndorf, U., Drexl, A., Nikulin, Y., and Pesch, E. (2007). Flight gate scheduling: State-of-the-art and recent developments, *Omega* **35**: 326–334.
- du Merle, O., Villeneuve, D., Desrosiers, J., and Hansen, P. (1999). Stabilized column generation, *Discrete Math.* **194**(1-3): 229–237.
- Freling, R., Wagelmans, A.P.M., and Paixao, J.M.P. (2001). Models and algorithms for single-depot vehicle scheduling, *Transportation Science* **35**(2): 165–180.
- Gao, C. (2007). *Airline Integrated Planning and Operations*, PhD thesis, Georgia Institute of Technology.
- Haghani, A. and Chen, M.C. (1998). Optimizing gate assignments at airport terminals, *Transportation Research Part A: Policy and Practice* **32**: 437–454.
- ILOG (2005). ILOG CPLEX v9.1, <http://www.ilog.fr>.

- Lohatepanont, M. and Barnhart, C. (2004). Airline schedule planning: Integrated models and algorithms for schedule design and fleet assignment, *Transportation Science* **38**(1): 19–32.
- Rexing, B., Barnhart, C., Kniker, T., Jarrah, A., and Krishnamurthy, N. (2000). Airline fleet assignment with time windows, *Transportation Science* **34**(1): 1–20.
- Sandhu, R. and Klabjan, D. (2004). Integrated airline planning, *AGIFORS Annual Symposium 2004, Singapore*.
- van Orden, A. (2002). *Gate assignment: Methods and models*, Master’s thesis, Department of Mathematics, Utrecht University.
- Xu, J. and Bailey, T.G. (2001). The airport gate assignment problem: Mathematical model and a tabu search algorithm., *HICSS '01: Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34)-Volume 3*.