Scheduling Aircraft Using Constraint Relaxation

Pim van Leeuwen[1], Nicolas van Hanxleden Houwert[2]

[1] National Aerospace Laboratory NLR
Anthony Fokkerweg 2, 1059 CM Amsterdam, The Netherlands
e-mail: leeuwenp@nlr.nl
[2] Hogeschool van Amsterdam
Instituut voor Informatica en Elektrotechniek,
Weesperzijde 190, 1097 DZ Amsterdam, The Netherlands
e-mail: N.van.Hanxleden.Houwert@ie.hva.nl

Abstract. In this paper, an aircraft departure scheduling tool for airports is presented based on constraint satisfaction techniques. Airports are getting more and more congested with the available runway configuration as one of the most constraining factors. A possibility to alleviate this congestion is to assist controllers in the planning and scheduling process of aircraft. In order to offer such assistance, it is important to realise that the scheduling problem is inherently over-constrained. In this paper, a tool is presented based on constraint satisfaction as a means to model the scheduling problem. The tool offers assistance in the establishment of an optimal or near-optimal departure schedule by relaxing soft constraints if needed. This goal is accomplished by incorporating constraint relaxation techniques into the constraint satisfaction problem, using ILOG Solver and Scheduler as an implementation environment.

1 Introduction

Due to the increase of air traffic in Europe, airports are becoming a major bottleneck in Air Traffic Control (ATC) operations. Expansion of airports is an expensive and time-consuming process and has a strong impact on the environment. Aviation authorities are seeking methods to increase airport capacity, while at least maintaining the current level of safety. This paper presents a tool to support ATC controllers in the establishment of optimal departure sequences. The scheduling tool provides a decision support function that has been designed to achieve a maximum throughput at the available runways whilst guaranteeing a satisfactory and safe solution at all times.

The objective of a runway departure sequencing function is to establish an optimal sequence in which aircraft can depart from the available runways at an airport and start their initial climb phase. Various technical and operational rules restrict the usage of runways, such as separation criteria for aircraft, timeslots in which aircraft should depart, and aircraft performance limits.

The work presented in this paper is based on and extends previous research done at NLR, which resulted in a departure scheduling tool based on constraint satisfaction techniques (see [8], [11], [24]). In this paper, first the operational problem of departure management is addressed by describing briefly current practice and identifying the role of departure planning at airports. Second, the tool is described in detail and an example solution is presented. Finally, some conclusions are drawn.

2 Departure Management

Controllers in airport control towers are responsible for the overall management of surface traffic at an airport. This is a difficult process: even under normal operating conditions at least three different controllers (one for each of the ‘pre-flight’, ‘taxiways’ and ‘runways’ areas) manage the aircraft on the airport. Each controller will try to establish an optimal plan for his/her own area and will try to hand over the aircraft to the next controller in an efficient way. Departure management at the runways is the responsibility of the runway controller. The tool featured in this paper intends to assist the runway controller in establishing optimal departure sequences, taking the plans of other controllers into account when needed.

The runway controller is the last planner in line and is dependent on the sequence of aircraft that is handed over by the previous (taxiway) controller. Typically, only minor changes to the provided sequence can be made through the use of runway holdings and intersection take-offs at the runway. The current way of working leads to a sub-optimal use of the available runway capacity, since the provided departure sequences are for the largest part fixed. In fact, the *runways* are the scarcer resource at airports. We will assume, therefore, that the *runway controller* (supported by the scheduling tool) determines the sequence of aircraft to obtain an optimal use of the runway capacity at the airport.

The departure management task entails the establishment of an optimal sequence of departing aircraft (the schedule) and the assignment of departure plans to these aircraft. Departure plans consist of start-up times at the gates, taxi plans for taxiing to the runways, and runway plans for take-off. The focus here is on the establishment of runway plans - start-up times and taxi plans can be derived from these plans. Runway plans specify which aircraft should use which runway for take-off, and at what time. Important for the establishment of runway plans is the so-called wake vortex separation, restricting departing aircraft at the same runway because of preceding aircraft that may be too close. Another relevant issue concerns the timeslot assigned to each aircraft. At most European airports, timeslots are co-ordinated time intervals of about 15 minutes in which aircraft should take off. Co-ordination is done with the CFMU (Central Flow Management Unit) in Brussels before the flight starts; the CFMU planning aims at obtaining a constant traffic flow through all flight sectors into which Europe is divided. For the airport controllers, this CFMU restriction ensures that the sectors are not overloaded by the feeders - the points where controllers hand over the flights to the next one.

3 Departure Scheduling using Constraint Satisfaction

Scheduling and planning have a long relationship with constraint representation and constraint-based reasoning ([5], [10], [18]). Constraints specify relationships between plans and specify how scarce resources can be used or when different parts of a plan need to be executed. Moreover, the separation rules that are applicable in air traffic control (specifying minimum distances between aircraft at for example the runway) can be regarded as restrictions or constraints. For this reason, constraint satisfaction has been chosen as the appropriate technique for solving runway planning problems, resulting in the tool presented in this paper.

This section describes the design of the NLR departure scheduling tool. First, a problem description is given of the departure scheduling problem encountered in practice. Second, the problem description is mapped onto a constraint satisfaction model, specifying the variables, domains, and constraints relevant to the tool presented.

3.1 Departure Scheduling Problem Description

Airports can be said to provide a variety of resources used by all departing, arriving, and ground traffic. For the departure scheduling problem, the existence of runways, Standard Instrument Departure (SID) routes and Terminal Manoeuvring Area (TMA) exit points are of specific importance. Runways connect to SID routes, which specify the route aircraft can take in airspace around the airport. SID routes lead to TMA exit points, marking the boundaries of the airspace around the airport (see [6], [11]). Figure 1 schematically depicts part of the topology of an example airport: Prague.

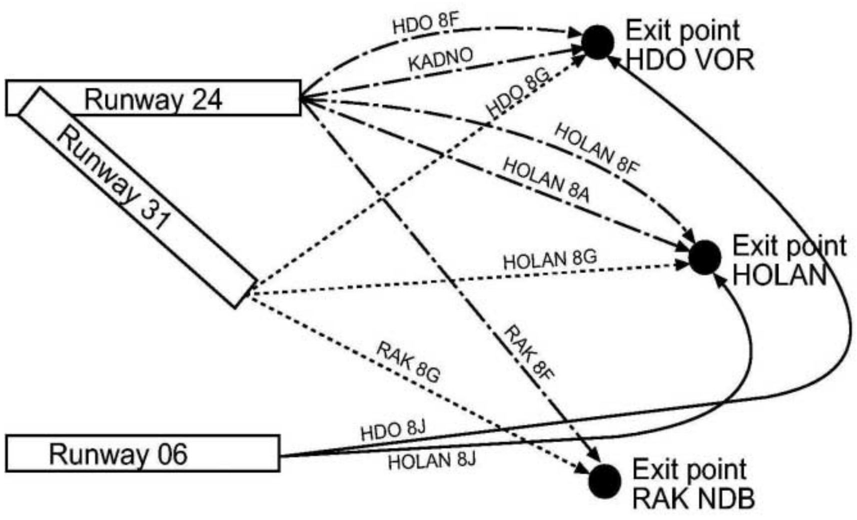

Fig. 1. Schematic representation of part of Prague airport: runways, SID routes, exit points

Given an airport with its runways, SID routes, and exit points, the departure management problem consists of allocating these resources and a suitable timetable to each flight to be scheduled. Suppose that $F_1, F_2, .., F_n$ is the set of flights to be scheduled. Assume, furthermore, that for each flight F_j is given:

- The aircraft with its corresponding properties (e.g., its speed and weight class).

- The destination within the Terminal Manoeuvring Area, which is the exit point.
- The CFMU (Central Flow Management Unit) time interval within which the flight needs to take-off.

Then for each flight F_j , the following will need to be planned:

- A take-off time, the time at which the aircraft should start its take-off roll at the runway.
- A runway, leading to the SID route.
- A SID flight time, the time needed to fly from runway to exit point.
- A SID route, leading to the TMA exit point.
- An exit time, the time at which the aircraft should pass the TMA exit point (depending on the total flight time).

3.2 Stating the Problem as a Constraint Satisfaction Problem

To solve the departure scheduling problem using constraint satisfaction, the existing departure scheduling tool has been designed to formulate it as a Constraint Satisfaction Problem (CSP). A CSP can be defined as:

- a set of *variables* $X=\{x_1,...,x_n\}$.
- a set of *domains* $D=\{D_1,...,D_n\}$, where D_i is a set of possible values for each variable x_i in X.
- a set of *constraints* $C=\{C_1,...,C_m\}$, restricting the values that the variables can simultaneously take.

In [24], a departure scheduling tool modelled according to this ‘classical’ CSP-scheme demonstrates acceptable performance for medium sized airports such as Prague airport. Further tests, however, lead to the conclusion that for larger airports with heavy traffic the situational complexity increases so dramatically that very large execution times or even failures to reach a solution are the result. Clearly, this is not acceptable for a decision support tool intended to assist air traffic controllers under real operating conditions. To overcome this problem, then, it was recognised that the problem space of departure scheduling is inherently over-constrained under circumstances of heavy traffic for large airports. This paper introduces constraint relaxation techniques to generate next-best solutions for situations of high complexity. To allow for constraint relaxation, the tool presented here will further divide the above constraints into sets of *soft* and *hard constraints*, from which the soft constraints can be relaxed (for literature on soft constraints, the reader is referred to [7], [14], [16], [23]). The Constraint Satisfaction Problem can now be reformulated as a constraint network $N=(X, D, C_h, C_s)$ in which X and D are defined as before, and:

- $C_h \subseteq C$ is the set of hard constraints.
- $C_s \subseteq C$ is the set of soft constraints.
- C_h and C_s are disjunct, and $C = C_h \cup C_s$.

3.2.1 Variables and Domains in Departure Scheduling

Above, the departure scheduling problem has been mapped onto a CSP-model, distinguishing variables, associated domains, soft, and hard constraints. The variables in the problem space have been identified with the flights that need to be scheduled. A flight has been defined as the total path of an aircraft from the gate via take-off to its exit point. Flights have been composed of the following parts:

- take-off at the runway.
- SID at the SID route.
- exit at the exit point.

The domains to which these flight parts (the variables) need to be assigned fall into two categories:

- the time point or time range, stating when a particular part of a flight needs to start.
- the resources (runways, SID routes, exit points) needed by that part of the flight.

With respect to the first category of values, time has been defined as a non-negative integer with 30-second- intervals as a unit. The departure scheduling tool extracts the resources corresponding to the second category from the airport topology, defining the runways, flight-routes, exit points and their connections for a given airport.

3.2.2 Constraints in Aircraft Scheduling

Constraints in a CSP restrict the combinations of values assigned to the variables in the domain. During the design of the departure scheduling tool, a number of constraints C_1 , C_2 ,…, C_m have been formulated to restrict the combinations of assigned times and allocated resources to (the relevant parts of) the flights to be scheduled. Given its problem space, the following types of constraints have been distinguished:

- Resource constraints, specifying which resources a flight (each part of it) requires.
- Order constraints, restricting the time-order of the parts constituting a flight.
- Timeslot constraints, stating that flights need to take-off within their CFMU-timeslot.
- Separation constraints, formulating minimum separation times between aircraft of specific speed and weight classes for runways and exit points.
- Topology constraints, describing which runways connect to which flight-routes and which exit points.
- Additional controller-imposed constraints, reflecting controller decisions to let aircraft depart in a specific order or at specific time-intervals or from specific runways.

As explained above, a distinction is made between hard constraints that cannot be relaxed, and soft constraints that do allow for relaxation. The soft constraints C_S can be relaxed in the following order[1] (i.e., with the first group of constraints as the first candidate for relaxation):

1. Conflicting additional controller-imposed constraints.

[1] In fact, the tool allows the user to change the relaxation order by changing the weights of the corresponding constraint groups if needed.

2. Timeslot constraints (for flights without a forced time of departure) for *refined* timeslots.
3. Preferred runway constraints.
4. Timeslot constraints (for flights without a forced time of departure) for *normal* timeslots.

The first group consists of controller-imposed constraints that are in conflict with themselves (e.g., when a controller forces two flights on the same runway with the same time of departure). These constraints *must* be relaxed to make the problem space consistent and are therefore the first candidate for relaxation. The second group of constraints specify *refined timeslots* for flights and are relaxed to overcome the problem of an over-constrained planning domain. Refined timeslots indicate special, narrow slots within which flights should take off (within their normal timeslot) to further enhance the runway throughput. Relaxation of these refined slots is allowed within the borders of the encompassing, normal timeslot. The third group specifies additional constraints to indicate runway preferences of controllers or pilots for specific flights. Runway preferences that air traffic controllers specify for certain flights can thus be abandoned if relaxation demands it. The fourth and final group consists of flights with normal timeslots: these are allowed to be relaxed, to a certain extent. All constraints in C that do not belong to C_s are members of the set of hard constraints C_h, which cannot be relaxed.

Within the above CSP-model of variables, values, soft, and hard constraints, the tool tries to find an assignment of departure times and an allocation of resources to each part of each flight. The scheduling tool tries to accomplish this assignment by use of ILOG Solver and Scheduler as an implementation environment supporting CSP modelling and solving (see [24] for further reading). If, however, such an assignment cannot be found, or cannot be found in time, relaxation of soft constraints needs to be performed to generate a next-best solution.

4 Scheduling Aircraft using Constraint Relaxation

This section describes the relaxation module that extends the existing constraint satisfaction based scheduling tool (reported on in [24]). The extension is invoked only in case of an over-constrained problem space, i.e., when no solution can be found, or none can be found within a certain time-limit (configurable by the user). First, the three relaxation techniques are described on which the relaxation module is based; second, the modelling is discussed in detail.

4.1 Constraint Relaxation Techniques

The following constraint relaxation techniques and concepts have been chosen for the relaxation module extending the CSP module to deal with over-constrained search spaces. These relaxation techniques are applied to the soft constraint groups listed in section 3.2.2.

4.1.1 User-Controlled Constraint Relaxation

The first group of section 3.2.2, conflicting controller-imposed constraints, are relaxed through User-Controlled Constraint Relaxation. User-Controlled Constraint Relaxation [2, 24] is a constraint relaxation technique used to allow the user to control the relaxation of constraints in case of an over-constrained problem space. By use of this technique, the controller can determine himself which constraints need to be relaxed to make the planning consistent again. Of course, the relaxation module provides support during this process. A limited list of controller-imposed conflicts and possible solutions is shown to assist the controller in the process, minimising the number of irrelevant options the controller needs to consider.

4.1.2 Weighted CSP

Timeslots and preferred runways are relaxed within the framework of *Weighted CSP* (see [1], [7], [16], [20]). Through the allocation of weights to soft constraints, the CSP is translated into a *Constraint Optimisation Problem*, aimed to minimise the total costs of departure schedules. Thus, weighted cost functions are associated with the soft constraints to allow for the calculation and comparison of different solutions. This scheme can be combined with *meta-constraints* to allow for a maximally flexible mechanism of soft constraint relaxation.

4.1.3 Meta constraints

Meta constraints are constraints imposed upon constraints, indicating a higher level type of constraints to structure and explicitly control the constraint relaxation process [3, 17]. Meta constraints allow for meta reasoning: a form of reasoning which explicitly specifies *which* type or group of constraints should be relaxed *to what extent*. To specify meta constraints, first a set of meta variables needs to be established related to the variables distinguished in the CSP-domain. These meta variables could specify the aggregated cost of a set of variables in the domain, for example. A meta constraint could then lay down the rule that for any solution involving relaxation these meta variables should correspond to each other according to a specific proportion. For instance, a meta constraint could specify that the relaxation cost for preferred runways should be exactly the same as the relaxation cost for refined timeslots. Many other meta constraints can be established, specifying the relaxation priority of certain soft constraints, the extent to which relaxation is allowed, or the interdependence of different relaxation constraints. In the next subsection, the implementation of these and the above given relaxation techniques is further elaborated.

4.2 Modelling Constraint Relaxation

In this subsection, first the modelling of conflicting controller-imposed constraints (constraint group 1 of section 3.2.2) is detailed. Following, the cost functions of the normal timeslot-, refined timeslot and preferred runway constraints (groups 2 to 4) are discussed.

4.2.1 Conflicting Controller-imposed Constraints

Controller imposed constraints that result in conflict are solved by giving the user a choice in the relaxation process. This User-controlled Constraint Relaxation is invoked when a syntactical conflict is detected. For example, the controller may force four flights to take off in a specific order, whereby a circular conflict is detected by the system:

```
# Conflict: circulair forced take-off times for four flights

force;AA1 < AA2
force;AA2 < AA3
force;AA3 < AA4
force;AA4 < AA1
```

Fig. 2. Conflict: circular forced take-off times

In this example, since AA1 < AA2 < AA3 < AA4 (with ‘<’ signifying ‘should take off before’), the last controller imposed constraint forcing AA4 to take off before AA1 results in a conflict, which is syntactically detected by the system. Another example of a detectable conflict is the following:

```
# Conflict: forced runway that does not match the flight’s exitpoint

add;AA1;B747;Outbound;RAKNDB;2000;0

force;AA1;06
```

Fig. 3. Conflict: forced runway not matching the flight’s exit point

In Figure 3, flight AA1, an outbound Boeing 747 having RAKNDB as exit point, t=2000[1] as estimated take-off time and 0 as priority is forced by the controller to take off from runway 06, whereas this runway is not connected via any SID-route to this exit point (see Figure 1 for the airport topology).

The question is, how User-Controlled Constraint Relaxation should be implemented to allow the controller to solve the above conflicts. Obviously, in the world of ATC, where workload is a crucial factor, the number of unnecessary system warnings should be minimal. Therefore, the system should be configured to resolve the above conflicts automatically if possible; only if this is not feasible or undesirable should the controller be bothered to choose from a limited set of alternatives (the actual User-

[1] t=2000 indicates that take off should occur 2000 seconds from now.

Controlled Constraint Relaxation). Thus, only in exceptional situations does the controller intervene, choosing in the first example to place flight AA4 before flight AA1 or in the latter example to reverse the forced take-off from runway 06 for a runway connecting to exit point RAK NDB.

4.2.2 Cost Functions for Timeslots and Runways

As indicated in 4.1.2., the Weighted CSP framework is used for timeslot and preferred runway constraints, defining cost functions for soft constraints that can be relaxed. Below, the cost functions are given for refined timeslot-, preferred runway- and normal timeslot constraints (corresponding to groups 2 to 4 of section 3.2.2). It is important to note that the parameters determining the shape of these cost functions are merely first estimates: further tuning is needed based on the outcome of operational tests. The ultimate goal, arrived at by first defining these cost functions, is to specify where the constraint relaxation module can find an optimal solution for circumstances in which the problem space is over-constrained.

4.2.2.1 Cost function for relaxation of refined timeslots

The following cost function is defined for the relaxation of refined timeslots:

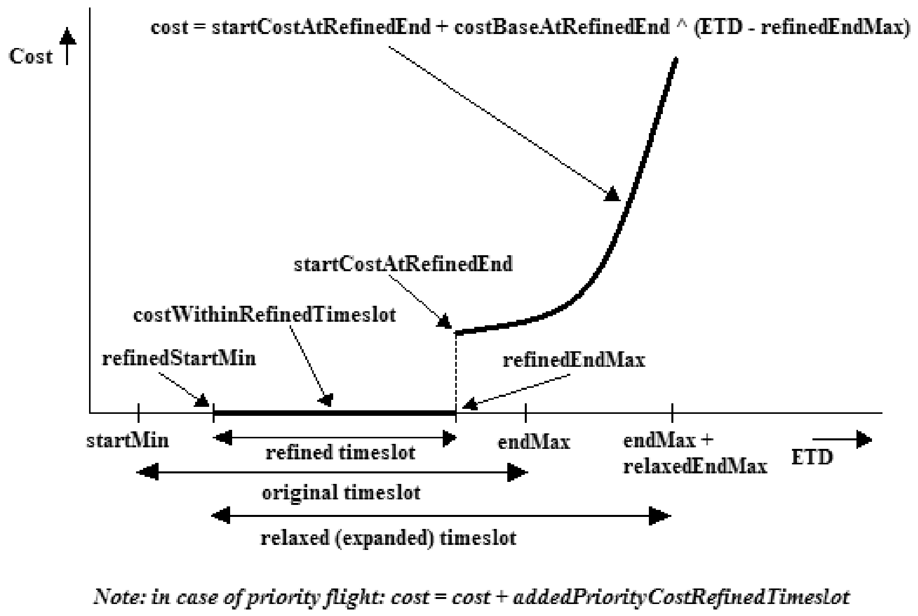

Fig. 4. Cost function for refined timeslots

In this cost function, the x-axis indicates the Estimated Time of Departure (ETD) of a flight, whereas the y-axis denotes the corresponding cost. In figure 4, the costs within the refined timeslot (between *refinedStartMin* and *refinedEndMax*) equal *costWithin-RefinedTimeslot* (typically *0*). Relaxation after the refined timeslot corresponds to an exponential cost function that runs from *refinedEndMax* to *endMax* + *relaxedEnd-Max,* expressing the idea that the more a flight is delayed, the more customer satisfaction is jeopardised and the larger the risks are of delaying other traffic, arriving late at the destination airport, etc. This exponential cost function is described as follows:

```
cost = (startCostAtRefinedEnd + costBaseAtRefinedEnd ^
(ETD - refinedEndMax))
```

with *startCostAtRefinedEnd* a configurable starting value and *costBaseAtRefinedEnd* the configurable exponential base. In case of a flight having high priority (designated by the controller), the configurable value *addedPriorityCostRefinedTimeslot* is added to the cost function to express the idea that a priority flight has a higher cost to be delayed in case relaxation is warranted.

4.2.2.2 Cost function for relaxation of preferred runways

The following cost function is defined for the relaxation of preferred runways:

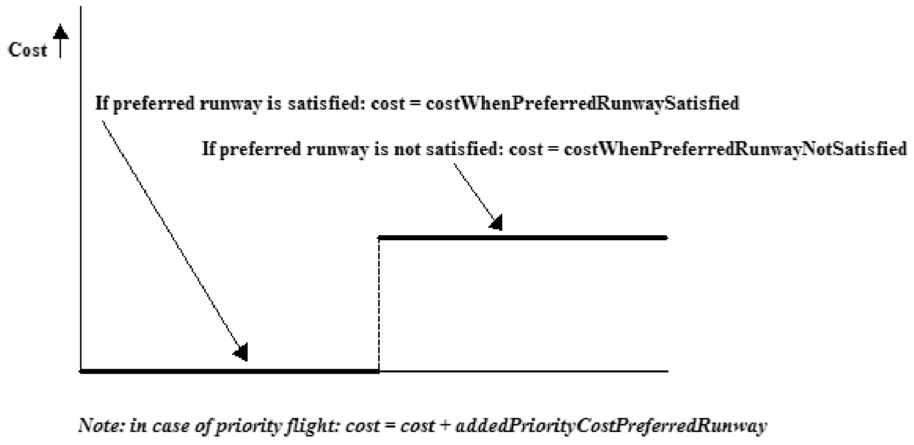
Cost
If preferred runway is satisfied: cost = costWhenPreferredRunwaySatisfied
If preferred runway is not satisfied: cost = costWhenPreferredRunwayNotSatisfied
Note: in case of priority flight: cost = cost + addedPriorityCostPreferredRunway

Fig. 5. Cost function for preferred runways

This cost function is very simple: it returns a cost of *costWhenPreferredRunwaySatisfied* (typically *0*) when a flight takes of from the requested preferred runway, and *costWhenPreferredRunwayNotSatisfied* otherwise. Similarly to the cost function for refined timeslots, a value of *addedPriorityCostPreferredRunway* is added to the cost in case of a high priority flight.

4.2.2.3 Cost function for relaxation of normal timeslots

The following cost function is defined for the relaxation of normal timeslots:

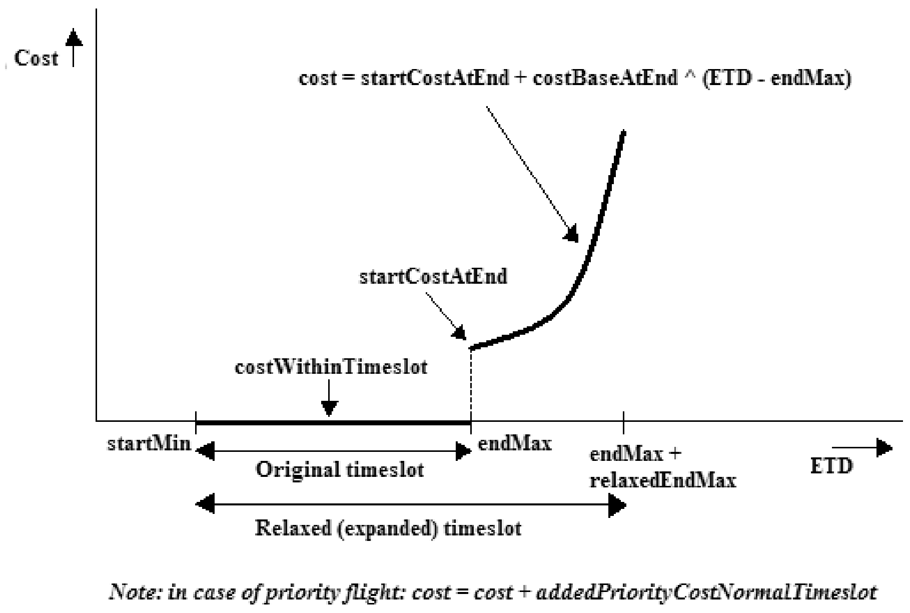

Fig. 6. Cost function for normal timeslots

If the flight takes off within the original timeslot (between *startMin* and *endMax*), the associated cost equals *costWithinTimeslot*, a constant that is typically *0.* Before *start-Min*, no relaxation is allowed, but after *endMax*, the take off time of a flight can be relaxed up to *relaxedEndMax*, a configurable variable. Relaxation within [endMax, relaxedEndMax] corresponds to the following exponential function:

```
cost = startCostAtEnd + costBaseAtEnd^(ETD – endMax)
```

Once again, a configurable value called *addedPriorityCostNormalTimeslot* is added to the cost function to express the idea that a priority flight has a higher relaxation cost than a non-prioritised flight.

4.2.3 Meta Constraints for Normal Timeslot, Refined Timeslot and Preferred Runway Constraints

Having defined the cost functions according to the Weighted CSP framework, higher level rules need to be established to guide the search process towards an optimal solution. To this end, meta constraints have been implemented imposing restrictions on the soft constraints that can be relaxed. To specify meta constraints for refined timeslot-, preferred runway- and normal timeslot constrains, a set of meta variables has been created. Each meta variable in this set corresponds to a (set of) variable(s) in the problem domain. For instance, the following meta variables have been implemented to model cost restrictions:

- XcostNormalTimeslot: specifying the total cost of all normal timeslot flights.
- XcostRefinedTimeslot: specifying the total cost of all refined timeslot flights.
- XcostpreferredRunway: specifying the total cost of all flights with preference for a runway.

Given these three meta variables, for example, the following meta constraint can be defined to specify that the costs of bound refined timeslot-, preferred runway and refined timeslot variables should be the same in any valid solution:

```
(XcostNormalTimeslot == XcostRefinedTimeslot ==
 XcostpreferredRunway);
```

Moreover, the optimisation goal can now be specified as follows:

```
XcostTotal = XcostNormalTimeslot + XcostRefinedTimeslot
+ XcostpreferredRunway;
model -> add(IloMinimize(*env, XcostTotal));
```

where IloMinimize sets the Ilog goal to minimize the cost *XcostTotal* for the environment *env* and the model *model* therein.

5 Results

The performance of the original CSP tool, obtained on a Sun Sparc 20 workstation running under Sun OS 5.6, has shown to be acceptable for conditions not exceeding a certain level of complexity (as reported in [24]). The relaxation module presented in this paper does not have a negative influence on performance under these circumstances, since it is not engaged unless the complexity reaches the threshold. If however this threshold – a configurable time-limit[1] - is reached, the implemented relaxation module proves to be very successful indeed. Figure 7, for instance, demonstrates how the system deals with conflicting controller constraints (group 1 of section 3.2.2):

```
# Conflict: Flights AA1 and AA2 have the same forced runway and etd

Solution by the constraint relaxation module:

Flight AA1 at t = 2000: 33 --> LUB5G_R33 --> LUB
Flight AA2 at t = 2000: 23 --> LUB5B_R23 --> LUB
```

Fig. 7. Result of solving conflicting controller constraints

[1] The system starts the relaxation module automatically if the CSP module cannot reach a solution within the time-limit.

In the situation depicted in Figure 7, two flights AA1 and AA2 having the same estimated time of departure (ETD) have been forced to depart from runway R33 by the controller. The solution provided by the system is to force flight AA2 to depart from R23, leading to exit point LUB as well (and assuring that no take-off delay occurs). Another example conflict solved by the relaxation module is the following:

```
# Conflict: four flights having the same etd and take-off runway

add;AA1;A1;B747;Medium;xxx;Outbound;Active;TypeS;LUB5B;2000;0
add;AA2;A2;B747;Medium;xxx;Outbound;Active;TypeS;LUB5B;2000;1
add;AA3;A3;B747;Medium;xxx;Outbound;Active;TypeS;LUB5B;2000;0
add;AA4;A4;B747;Medium;xxx;Outbound;Active;TypeS;LUB5B;2000;1

force_runway;AA1;33;0
force_runway;AA2;33;0
force_runway;AA3;33;0
force_runway;AA4;33;0

First solution by the constraint relaxation module, cost: 83.6406.

Flight AA1 at t = 2360: 33 --> LUB5G_R33 --> LUB
Flight AA2 at t = 2240: 33 --> LUB5G_R33 --> LUB
Flight AA3 at t = 2120: 33 --> LUB5G_R33 --> LUB
Flight AA4 at t = 2000: 33 --> LUB5G_R33 --> LUB

Second solution by the constraint relaxation module, cost: 33.6406.

Flight AA1 at t = 2360: 33 --> LUB5G_R33 --> LUB
Flight AA2 at t = 2120: 33 --> LUB5G_R33 --> LUB
Flight AA3 at t = 2240: 33 --> LUB5G_R33 --> LUB
Flight AA4 at t = 2000: 33 --> LUB5G_R33 --> LUB
```

Fig. 8. Result of solving timeslot constraints

In this figure, first flight plans AA1 to AA4 are added to take-off all within refined timeslots at t=2000 to destination LUB5B (the exit point), whereas the priority of AA2 and AA4 is high (the underlined last digit equals 1). Next, the air traffic controller forces all four flights to take off from the same runway, R33. Given this over-constrained situation[1], the relaxation module is engaged to solve the conflict. In the first loop, the module offers a solution in which flights AA1 to AA3 are relaxed yielding a total cost of 83.6406. Since AA2 and AA4 have a high priority, it would be smarter to relax flights AA1 and AA3 the most, which is done in the second solution. Note that the controller imposed take-off runways are not relaxed, since these runways are not preferred but obliged (force_runway). The relaxation module loops through a number of possible solutions in search of an acceptable solution. Both the number of search loops and the maximum search time the scheduling tool spends per loop are configurable by the user.

In fact, situations may arise in which not only the hard-constrained scheduling module, but also the relaxation module fails to reach an acceptable solution within

[1] This situation is over-constrained, since all flights are to be separated 2 or 3 minutes on the same runway, and a refined timeslot of 4 minutes is assumed for each flight.

time[1]. In such rare situations[2], even the relaxation of constraints cannot prevent the failure to establish a schedule. Decision support can then no longer be offered, and the scheduling tool will simply schedule the flights on their calculated take-off times (derived from the typically rough and inefficient tactical planning made the day before) and summon the controllers to request new CFMU time slots for those flights that cannot make their calculated take-off time (thus delaying them even further). The tool therefore does guarantee a solution at all times, although the schedule thus provided is far from elegant. Fortunately, however, these situations are very rare, since relaxing the numerous constraints has proved to be a very powerful means to overcome over-constrained problem spaces in practice and find sub-optimal solutions that are satisfactory (a conclusion supported by both operational tests, see section 7).

6 Related Work

In the field of air traffic management, many different techniques such as evolutionary computing techniques, fuzzy logic and agent technology have been applied to solve planning problems (e.g., [9], [19], [22]). Several planning problems, however, are stated as constraint satisfaction or relaxation problems (e.g., [4], [8], [15], [18]). Among the constraint satisfaction solutions applied to the area of Departure Management are RESO[18], and DSP[15]; a system including Departure Management in a broader scope is TARMAC[4].

Closest to our work is the Departure Manager Runway Event Sequence Optimiser (RESO) [18]. This prototype application is aimed to de-conflict aircraft with similar departure times, minimising the amount of delay incurred by departing aircraft and reducing the percentage of aircraft that miss their time window. In contrast to the work presented here, RESO focuses more on static planning than run-time dynamical planning. Similarly to the NLR scheduling tool, however, RESO does allow for the relaxation of constraints, permitting schedules with aircraft planned outside their assigned timeslots.

The Departure Spacing Program (DSP) places a greater emphasis on flow management [15]. The DSP calculates departure schedules by co-ordinating the release of departures from multiple airports to produce a level of demand that can be managed by controllers as departure traffic converges on common departure flow fixes. The DSP can provide a smooth flow of traffic in the sense of scheduling aircraft at a departure flow fix to separate them by intervals of time. Thus, the DSP can guarantee that the total number of aircraft a sector controller must handle at any given time, which is destined for the constrained fix, is kept at a manageable level.

The aim of the Taxi And Ramp Management And Control (TARMAC) system is to combine a runway occupancy-planning tool, a movement area planning system and an apron planning tool [4]. The controller is assisted in his planning of the aircraft motions on the apron, especially with respect to pushback times of departing aircraft

[1] *Not in time* needs to be stressed here, since the module can always find a solution, but this may take too long in an operational setting.

[2] In both operational tests of the relaxation module, involving actual ATC-controllers and realistic scenarios, no such situation was encountered.

and taxi ways. The TARMAC planning unit supports the controller by visualising future traffic situations, by recognising planning conflicts and undetermined taxi sequences, and by automatically planning sequences in many safe situations.

7 Conclusions and Further Work

The departure scheduling tool presented in this paper provides runway controllers with a decision support tool to establish optimal or sub-optimal departure sequences for aircraft. As a consequence, runway capacity will be effectively enhanced without any physical changes to the airport infrastructure. The major advantage of this extended tool lies in its flexibility: any airport topology can be used, inbound traffic is taken into account, and certain take-off times or take-off orders can be fixed while others can be scheduled. Finally, in contrast to the first prototype (see: [24]), the relaxation module provides a solution at all times through its use of an elaborated constraint relaxation scheme to overcome the highly complex or conflicting situations that may be encountered in practice.

The performance our solution achieves is acceptable for practical application at any airport and under any circumstance. When the traffic load reaches a certain threshold, the relaxation routine may be engaged to importantly enhance the system's capabilities to reach an acceptable solution. The tool has been tested twice under real-time conditions at NLR's Tower Research Simulation facility at Schiphol airport; during both tests, air traffic controllers were involved to evaluate the performance of the tool and to establish realistic cost functions. The most important outcome of these tests has been that given an over-constrained traffic situation, the relaxation module was indeed able to provide an acceptable solution where the original scheduling module was not (at least, not within the given time-limit). Following these tests, research will be directed towards implementing the established controller's cost functions and capturing cost functions for all other parties involved in the departure management process at airports (such as the airlines, the ground handlers and the people living in the vicinity of the airport).

Given these promising test results and the fact that controllers remain involved in the process at all times (and are guaranteed to get a solution), we expect the operational acceptance of the tool to be high. In an operational setting, air traffic controllers will be able to impose restrictions on a schedule beforehand, and make modifications to calculated solutions afterwards by re-planning parts of the generated schedule. The scheduling and planning tool therefore only finds solutions that match the idea of a 'good' solution that runway controllers already have.

References

1. Barták, R., Guide to Constraint Programming, Extending CSP, Charles University, Faculty of Mathematics and Physics, Department of Theoretical Computer Science and Mathematical Logic, 1998

2. Becker, M., F. Smith, Mixed-Initiative Resource Management: The AMC Barrel Allocator, Proceedings 5th International Conference on AI Planning and Scheduling, April, 2000
3. Berlandier, P., The use and interpretation of meta level constraints, EPIA'93, 1993
4. Dippe, D.: The DLR Activities for Development, Test and Evaluation of a Ground Movement Management and Control System", AIAA-95-3369-CP, Proceedings of the AIAA Guidance, Navigation and Control Conference, pp. 1788-1797, Baltimore, MD (1995)
5. Do, M. B., S. Kambhampati, Planning as constraint satisfaction: Solving the planning graph by compiling it into CSP, in: Artificial Intelligence 132 (2), pp. 151-182, Nov., 2001
6. FAA, Aeronautical Information Manual – Official Guide to Basic Flight Information and ATC Procedures, February 2000
7. Freuder, E.C., R.J. Wallace, R. Hefferman, Ordinal Constraint Satisfaction, p. 15, 5th International Workshop on Soft Constraints, Held in conjunction with 9th International Conference on Principles and Practice of Constraint Programming, CP2003, Actons Hotel, Kinsale, County Cork, Ireland, 29 September - 3 October, 2003
8. Hesselink, H.H., Basjes, N.: MANTEA Departure Sequencer - Increasing Airport Capacity by Planning Optimal Sequences, Proceedings of Eurocontrol/FAA ATM'98 Conference, 1-4 December 1998, Orlando, NLR-TP-99279 (1998)
9. Hesselink, H.H., Kuiper, H., van den Akker, J.M.: Application of Genetic Algorithms in the Aerospace Domain, NLR-TP-96655, Proceedings of the Fourth European Congress on Intelligent Techniques and Soft Computing (EUFIT), Aachen, Germany (1996)
10. Hesselink, H.H., Paul, S,: Planning Aircraft Movements in Airports with Constraints Satisfaction, September 1998, NLR-TP-98397 and published by Springer-Verlag in Proceedings of the Third IMA Conference on Mathematics in Transport Planning and Control, ISBN 0-08-043430-4 (1998)
11. Hesselink, H.H., Visscher, J.J.E.: Design of Runway Planning and Sequencing, Amsterdam, NLR TR-97094 L and in MANTEA/ISR-TEC-D5.1-057 (1997)
12. ILOG Scheduler Reference Manual, version 5.0, July 2000, ILOG
13. ILOG Solver Reference Manual, version 5.0, August 2000, ILOG
14. Kumar, V., Algorithms for Constraint Satisfaction Problems; A Survey, in AI Magazine 13 (1), pp. 32-44, 1992
15. Overmoe, C.: Requirements Document for Departure Spacing Program, Proof of Concept Phase II, FAA (1999)
16. Petit, T., C. Bessiere, J.C. Regin, A general conflict-set based framework for partial constraint satisfaction, p. 14, 5th International Workshop on Soft Constraints, Held in conjunction with 9th International Conference on Principles and Practice of Constraint Programming, CP2003, Actons Hotel, Kinsale, County Cork, Ireland, 29 September - 3 October, 2003
17. Petit, T., J.C. Régin, C. Bessière, Meta-Constraints on Violations for Over-Constrained Problems, ILOG, ILOG-LIRMM, 2000
18. Roberts, S., Foster, E.: (2001). D-MAN User Guide, EUROCONTROL DMAN/UG/001
19. Robinson III, J.E., Davis, T.J., Isaacson, D.R.: Fuzzy Reasoning-Based Sequencing of Arrival Aircraft in the Terminal Area, AIAA Guidance, Navigation and Control Conference, New Orleans, LA, August 1997
20. Rudová, H., Constraint Satisfaction with Preferences, Ph.D. Thesis, Masaryk University, Faculty of Informatics, 2001
21. Smith, S., O. Lassila, M. Becker, Configurable, Mixed-Initiative Systems for Planning and Scheduling, in Advanced Planning Technology, A. Tate, Ed.: AAAI Press, 1996.
22. Tomlin, C., Pappas, G.J., Shankar, S.: Conflict Resolution for Air Traffic Management; a Case Study in Multi-Agent Hybrid Systems, Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, 1996
23. Tsang E., Foundations of Constraint Satisfaction, Academic Press, Essex, 1993

24. Van Leeuwen, P., Hesselink, H.H, Rohling, J.H.T.: Scheduling Aircraft Using Constraint Satisfaction, WFLP 2002, Proceedings of the 11th International Workshop on Functional and (Constraint) Logic Programming, Grado (Italy), ISBN 0-44-4513116 Vol. 76 (2002)