Nationaal Lucht- en Ruimtevaartlaboratorium
National Aerospace Laboratory NLR

NLR-TP-2002-299

Scheduling aircraft using constraint satisfaction

P. van Leeuwen, H.H. Hesselink and J.H.T. Rohling

Nationaal Lucht- en Ruimtevaartlaboratorium
National Aerospace Laboratory NLR

NLR-TP-2002-299

Scheduling aircraft using constraint satisfaction

P. van Leeuwen, H.H. Hesselink and J.H.T. Rohling

This report is based on a presentation held at the 11th Workshop for Functional and (constraint) Logic Programming, Grado – Italy, 20-22 June 2002.

Customer:	National Aerospace Laboratory NLR
Working Plan number:	I.1.B.2
Owner:	National Aerospace Laboratory NLR
Division:	Information and Communication Technology
Distribution:	Unlimited
Classification title:	Unclassified
	May 2002

Summary

In this paper, an airport departure scheduling tool for aircraft 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. The prototype presented here is aimed to offer such assistance in the establishment of an optimal departure schedule and the planning of initial climb phases for departing aircraft. This goal is accomplished by modelling the scheduling problem as a constraint satisfaction problem, using ILOG Solver and Scheduler as an implementation environment.

Contents

(20 pages in total)

1 Introduction

With 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 report presents a prototype to support airport tower 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 and reduce the controller's workload and the number of delays.

Scheduling departure sequences comprise sub-problems such as runway (entry) allocation, SID (Standard Instrument Departure) allocation, and the application of specific airport procedures (such as the take-off after procedure). The objective of a runway departure sequencing function is to establish an optimal sequence in which aircraft can depart from the available runways and start their initial climb phase. Various technical and operational rules restrict the usage of runways, such as separation criteria of aircraft, departure timeslots, and aircraft performance limits.

The work presented in this paper is based on research done at NLR (e.g., in the Triple-I and MANTEA projects, see [1], [2]). 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, a mapping of the departure management problem to constraint satisfaction is described. Third, the prototype 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 the 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 provide the aircraft to the next controller in an efficient way. Departure management at the runways is the responsibility of the runway controller. The prototype featured in this paper intends to assist the runway controller in establishing optimal departure sequences, taking the plans of other controllers into account where necessary.

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. At the runway, typically only minor changes to the provided sequence can be made through the use of runway holdings and intersection take-offs. 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. A way out to this problem follows from the recognition that the *runways* are the scarcer resource at airports. We will assume, therefore, that the runway controller (assisted by the prototype) 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 sectors in Europe. 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 Scheduling Aircraft using Constraint Satisfaction

Scheduling and planning have a long relationship with constraint representation and constraint-based reasoning (e.g., [3], [10]). 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. Therefore, constraint satisfaction is chosen as the appropriate technique for solving runway planning problems.

3.1 Problem Description

Airports can be said to provide a variety of resources used by all departing, arriving, and ground traffic. For the departure management problem at hand, 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 above the airport. SID routes lead to TMA exit points, marking the boundaries of the airspace around the airport (see: [1], [4]). Figure 1 below schematically depicts part of the topology of an example airport: Prague.

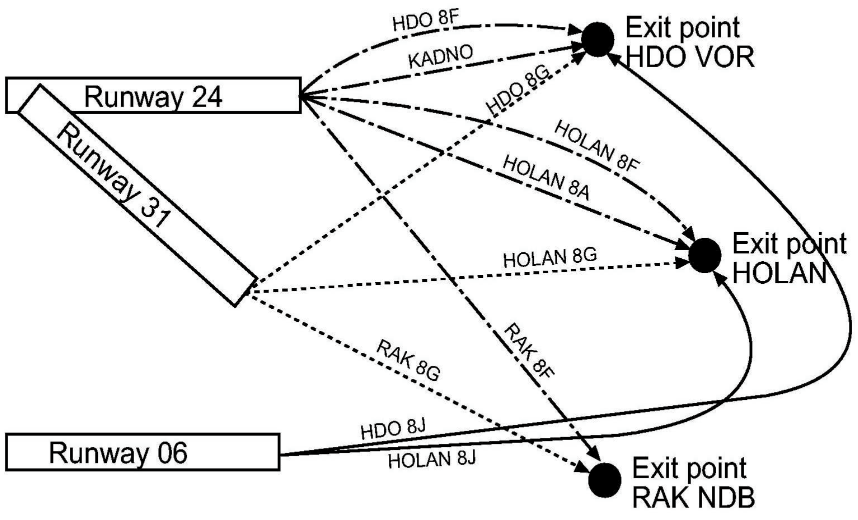

Figure 1: Schematic representation of part Prague airport: runways, SID routes and exit points

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 point, which is the TMA 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 total flight time, stating the time at which the aircraft should start flying its SID route (depending on the take-off time) and when it should complete the route.
- 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 management problem using constraint satisfaction, we need to formulate it as a Constraint Satisfaction Programming (CSP) problem. A CSP-problem can be defined as:

- a set of *variables* $X=\{x_1,...,x_n\}$.
- for each variable x_i, a finite set D_i of possible values (its *domain*).
- a set of *constraints* restricting the values that the variables can simultaneously take.

3.2.1 Variables and Domains in Aircraft Scheduling

In order to describe the planning problem as a CSP-problem, we need to distinguish the variables, their associated domains and relevant constraints in the problem description. The variables in the problem space can then be identified with the flights that need to be scheduled. A flight will be defined as the total path of an aircraft from the gate via take-off to its exit point: the destination at which the aircraft leaves the airport's airspace. Flights are then constructed of the following parts:

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

The gate itself, it should be noted, as well as the various taxi routes an aircraft could take to reach a runway are currently excluded from the problem space (but will be included in future prototypes). The values to which all constituents of flights 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 is defined as a non-negative integer with one-minute-intervals as a unit. The resources corresponding to the second category are extracted 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-problem restrict the combinations of values assigned to the variables in the domain. For the departure management problem, a number of constraints C_1 , C_2 ,…, C_m can be formulated to restrict the combinations of assigned times and allocated resources to all parts of the flights to be scheduled. Given its problem space, the following types of constraints can be 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 different speed and weight class for runways and exit points.
- topology constraints, describing which runways connect to which flight-routes and which exit points.
- additional tower-control constraints, reflecting controller decisions to let aircraft depart in a specific order or at specific time-intervals.

The different types of constraints will be further detailed in section 4.1 below. Having formulated the variables, values, and constraints relevant to the departure management problem, the task to be accomplished now is to find an assignment of times and allocation of resources to each subpath of each flight, such that none of the constraints is violated. This task can be described as a scheduling task (being a specific CSP problem): a process of allocating scarce resources to activities over a period of time. In the following section, ILOG Solver and Scheduler are introduced as a tool to model the departure management problem and find solutions.

4 Constraint-based Scheduling with ILOG

ILOG offers optimisation software suitable for modelling and implementing constraint satisfaction problems. ILOG Solver is a general constraint-based optimisation engine, providing optimisation technology for scheduling, sequencing, timetabling, or applications with logical constraints. ILOG Scheduler is an add-on to this Solver engine, created specifically for solving scheduling problems. As a C^{++} library, it can be easily integrated with existing software. Combined with the fact that ILOG Scheduler offers a great variety of scheduling algorithms and heuristics, this product was chosen to be used as an aid to model and implement the departure management problem as a CSP-problem.

4.1 Scheduling Departures with ILOG Software

Scheduling is defined as the process of allocating scarce resources to activities over a period of time. Constraint-based scheduling applies constraint programming techniques to solve scheduling problems. In ILOG terminology, a scheduling problem can be defined by [5]: solutions.

- A set of *activities*: the tasks to be completed resulting in a schedule.
- A set of *resources*: objects which add value to a product or service in its delivery and that can/need to be allocated to the activities.
- A set of *temporal constraints*: relationships between start and end times of activities.
- A set of *resource constraints*: demands of the activities upon the resources.

In the domain of departure planning, the flights to be planned can be modelled as activities to be scheduled. Furthermore, the taxiways, runways and exit points of an airport can be mapped onto resources (i.e., all objects that need to be shared by all aircraft to be scheduled). A time window will be assumed within which all flights need to be scheduled. Time and resources thus become the values to which the flights (the variables) need to be committed. Different types of constraints, as listed in section 3.3., can be mapped onto temporal or resource constraints in the ILOG environment. The following table details the mapping between the objects in the problem description, their CSP-equivalent and their ILOG counterpart:

Object	Description	CSP-equivalent	ILOG-equivalent
Flight: • take-off • SID • exit	The flight path to be scheduled; it can be subdivided into three subpaths	All subpaths are variables	All subpaths are activities
Runways, SID routes, exit points	All airport resources to be shared by the flights	Values	Resources
Time	The time within the time window	Values	Integers
Inbound aircraft	Aircraft arriving at the same runways used for departures (mixed-mode)	Constraints	Breaks on resources
Resource constraints	Restrictions specifying which resources are needed by each subpath of a flight	Constraints	Resource constraints
Order constraints	Restrictions on the order of take off, flight and exit point for flights	Constraints	Temporal constraints
Timeslot constraints	Restrictions specifying in which CFMU-timeslot flights need to take off	Constraints	Temporal constraints
Separation constraints	Restrictions formulating minimum separation times between aircraft of different speed and weight classes for runways and exit points	Constraints	Temporal constraints
Topology constraints	Restrictions describing which runways connect to which flight-routes and which exit points; Restrictions defining the format of runways, flight-routes and exit points	Constraints	Matrix of allowed combinations of resources; Matrix of necessary take-off and flight-times
Additional tower-control constraints	Controller decisions to let aircraft depart in a specific order or at specific time-intervals	Constraints	Temporal constraints

Table 1: Mapping the problem on CSP and ILOG.

In table 1, some interesting modelling decisions have been made. Section 4.2 describes the model, discussing the variables, values, and constraints mapped onto domain specific elements.

4.2 The Scheduling Model

4.2.1 Flights, resources and time

As indicated above, flights are the variables of the CSP-problem mapping onto *activities* in ILOG Scheduler. Every flight is divided into three subpaths: a take-off-, flight- and exit activity. Activities can be named as follows (these names will serve as an example below):

- $\textit{take-off}_1$: take-off of flight 1.
- $\textit{sidroute}_2$: following SID route of flight 2.
- $\textit{exit}_3$: passing exit point of flight 3.

The resources - the values to be assigned to the flights – for Prague airport are:

Resource	Examples of Prague airport
Runways	R24, R31, R06, (R13)[1].
SID routes	HDO 8F, HOLAN 8F, RAK 8F, VOZ 8F, BANAS 1A, HDO 8A, HOLAN 8A, LAGAR 8A, VOZ 8A (for runway R24); HDO 8G, HOLAN 8G, RAK 8G, VOZ 8G, BANAS 1B, LAGAR 8B, VOZ 8B (for runway R31); HDO 8J, HOLAN 8J, RAK 8J, VLM 9J, VOZ 9J, BANAS 1D, LAGAR 9D (for runway R06).
Exit points	HDO VOR, RAK NDB, VOZ NOR, BANAS, KADNO, HDO VOR, HOLAN, LAGAR, VOZ VOR, VLM VOR.

Table 2: Resources at Prague airport.

All resources in ILOG are modelled as *alternative resources*, indicating that such resources can be seen as sets of resources of which any element can be allocated (without preference) to the flights that require them. Thus, the set *runways* contains elements *R24*, *R31* and *R06*, for example. The time values to which the flights need to be committed are modelled as integers.

4.2.2 Inbound traffic

Inbound traffic is implemented by so-called *breaks* upon the resources defined in the model. Breaks are used to mark off time-intervals at which resources cannot be allocated. This is useful for mixed-mode operations: when the same runway is used for both departures and arrivals. To

[1] Runway R13 is currently not in use

model mixed-mode, breaks are defined for a runway indicating when it is occupied by arriving aircraft. During these breaks, the runway cannot be used for departures. For example:

```
model.add(R31.addBreak(10,15));
model.add(R06.addBreak(35,40));
```

adds two breaks, one from t=10 to t=15 for runway R31, the other from t=35 to t=40 for R06.

4.2.3 Resource constraints

Resource constraints connect activities with the resources they require. In the prototype, this connection is expressed by requirement constraints. To indicate that flight 1 requires a runway (an element of the set of runways) for its take-off, flight 2 a SID route to follow its SID and flight 3 an exit point to complete its path, the prototype specifies:

```
model.add(take-off1.requires(runways,1));
model.add(sidroute2.requires(SIDroutes,1));
model.add(exit3.requires(exitpoints,1));
```

4.2.4 Order constraints

Order constraints restrict the order in which the different activities can be scheduled. For example, to denote that the take-off should precede the SID for flight 1, the model should be extended as follows:

```
model.add(sidroute1.startsAtEnd(take-off1));
```

Alternatively, one could specify that exit follows the SID:

```
model.add(exit1.startsAtEnd(sidroute1));
```

4.2.5 Timeslot constraints

Timeslot constraints specify the CFMU-timeslot flights need to use for take-off. As an example, to codify that flight 1 should take-off in timeslot [0,15] the prototype implements:

```
model.add(take-off1.startsAfter(0));
model.add(take-off1.endsBefore(15));
```

4.2.6 Separation constraints

For a given airport, certain separation rules are prescribed. Separation settings define the minimum separation times aircraft at the runways or exit points need to obey. The prototype enables the user to configure separation criteria for runways and exit points for all relevant combinations of aircraft types.

For runways, one can set the minimum separation times for aircraft. In our prototype, two minimum separations can be distinguished: a large separation and a default separation. The large separation will be observed when an aircraft taking off after another aircraft is either in a larger speed class or in a smaller weight class. Under these circumstances, separation should be larger due to the stronger impact the wake vortex (i.e., the air turbulance) of the preceding aircraft can have on the following aircraft. For example, when a Cessna Citation (weight class Light) or a Fokker Friendship F27/500 (weight class Medium) takes off after a Boeing 747-400 (weight class Heavy) on the same runway, the large separation time needs to be observed. In any other situation, the default separation time should be adhered to.

This scheme is implemented by creating a transition times table, specifying the amounts of time that must elapse between the end of any activity A_1 and the beginning of any activity A_2 [6]. The table is then associated with the runway resources and filled with the minimum separation times to be observed for any two aircraft taking off. In our solution, the function GetTransitionTime returns the separation time between any aircraft with speed class *sp1* and weight class *wt1* that takes off before any aircraft with speed class *sp2* and weight class *wt2*:

```
IloNum GetTransitionTime(const SpeedClass sp1, const SpeedClass sp2,
const WeightClass wt1, const WeighClass wt2)
{
  if ((sp1 < sp2) || (wt1 > wt2))
    return maxRunwaySeparation;
  else
    return defaultRunwaySeparation;
}
```

This function is called when filling the transition times table *ttParam* with minimum separations for all combinations of speed and weight classes:

```
for (SpeedClass sp1=A; sp1<=E; sp1++)
  for (WeightClass wt1=Light; wt1<=Heavy; wt1++)
    for (SpeedClass sp2=A; sp2<=E; sp2++)
      for (WeightClass wt2=Light; wt2<=Heavy; wt2=++)
        (*ttParam).setValue((sp1*wt1),(sp2*wt2),
        GetTransitionTime(sp1, sp2, wt1, wt2));
```

Thus, the transition times table is created and filled, effectively constraining any planning solution to the minimum separations times to be observed for aircraft taking off.

For exit points, other separation times may and typically do apply. To separate aircraft on exit points, a minimum separation time can be set in a fashion similar to that for runways. In this way, it can be assured that sectors are not overloaded, since aircraft entering the sector via the exit points will be sufficiently spaced in time.

4.2.7 Topology constraints

In our solution, matrixes are defined to lay down the connections between runways, SID routes, and exit points corresponding to the airport topology used. For example, for Prague airport constraints are defined to indicate that runway R06 connects to SID routes HDO 8J, HOLAN 8J, RAK 8J, VLM 9J, VOZ 9J, BANAS 1D, LAGAR 9D leading to exit points HOLAN, HDO VOR, RAK, VLM VOR, VOZ VOR, BANAS and LAGAR.

Moreover, flight times over SID routes depend on the chosen flight route and the aircraft type. For example, the duration of flight route *HOLAN_8F* can be specified as follows (where the route length is *Distance_HOLAN8F* and *Speed* is a variable related to the speed class of the aircraft):

```
model.add(SIDroutes.select(HOLAN_8F) <=
         (Duration==(Distance_HOLAN8F/Speed));
```

This code specifies the constraint that if flight route *HOLAN_8F* is selected from the set of alternative resources *SIDroutes*, variable *Duration* is calculated from the distance of this route and the speedclass of the aircraft. Thus, since ILOG enables activities to be constructed with variable durations, all flight activities using route *HOLAN_8F* will be effectively set to require this resource for the period *Duration*.

4.2.8 Additional tower-control constraints

Additional tower-control constraints can be added to reflect controller decisions to influence the schedule. The prototype allows the tower-control to change the order or time interval in which specific flights should take-off. For example, to force flight 2 to take-off before flight 1, the prototype implements:

```
model.add(take-off1.startsAfterEnd(take-off2));
```

Similarly, to force flight 1 to take-off at t=15:

```
model.add(take-off1.startsAt(15));
```

5 Performance Results

Results of our departure scheduling prototype, obtained on a Sun Sparc 20 workstation running under Sun OS 5.6, show that acceptable performance can be achieved to enable its practical use. Table 3 below gives a list of aircraft planned to depart from Prague airport in a time interval of 50 minutes. The following parameters were used:

- Separation at the runways: 2 minutes default, 3 minutes after heavy or slow aircraft.
- Separation at the exit points: 5 minutes.
- Timeslots are 15 minutes per aircraft.

The optimisation function minimises the total time needed for all flights to take-off, follow its SID route and exit the Prague airspace. During the search, the algorithm chooses the runway and SID such as to yield an optimal solution. For example, when two flights have the same exit point and destination, the algorithm may choose different runways to let both aircraft take-off at the same time, minimising the total time needed for both flights to leave the airport. In the table, the number of aircraft assigned to the same timeslot and having the same destination is used as a measure for the complexity of the problem:

#aircraft	#same timeslot for a destination	#constraints	#backtracks	time to solution (s)
5	0	563	0	0.22
6	2 for HOLAN	656	0	0.19
7	3 for HOLAN	749	1	0.23
8	3 for HOLAN, 2 for RAK NDB	839	1	0.23
9	3 for HOLAN, 2 for RAK NDB	931	1	0.28
10	3 for HOLAN, 2 for RAK NDB	1019	1	0.33
11	3 for HOLAN, 2 for RAK NDB, 2 for LAGAR	1111	33	0.43
12	3 for HOLAN, 2 for RAK NDB, 3 for LAGAR	1203	37	0.46

Table 3: Solutions for a departure planning at Prague airport.

Table 3 shows a minimal increase in solution time with the number of aircraft to be planned. The search strategy used by the algorithm tries to assign start times and resources to each activity (take-off, flight, exit) while minimising the total time. Once a start time or resource is assigned, this fact is propagated as a new constraint to the set that still needs to be assigned. This constraint propagation explains why potentially conflicting situations – flights having the same destination and timeslot – do not result in extensive backtracking behaviour: flights assigned to time slots and resources are propagated reducing the search space for subsequent flights.

Another test can be done to measure the performance of our prototype under mixed-mode operation. To this end, three inbound aircraft are assumed:

- an aircraft arriving at runway R24 in [0, 5].
- an aircraft arriving at runway R24 in [20,25].
- an aircraft arriving at runway R31 in [15,20].

In this limited-runway-availability scenario, the following results are obtained:

#aircraft	#same timeslot for a destination	#constraints	#backtracks	time to solution (s)
5	0	565	0	0.24
6	2 for HOLAN	658	4	0.23
7	3 for HOLAN	751	136	0.63
8	3 for HOLAN, 2 for RAK NDB	841	131	0.64
9	3 for HOLAN, 2 for RAK NDB	933	437	1.43
10	3 for HOLAN, 2 for RAK NDB	1021	603	1.78
11	3 for HOLAN, 2 for RAK NDB, 2 for LAGAR	1113	1516	4.35
12	3 for HOLAN, 2 for RAK NDB, 3 for LAGAR	1205	2621	6.3

Table 4: Solutions for departure planning at Prague airport, including arrivals.

In the last row, a very difficult (and admittedly unrealistic) situation is created, since a total of eight aircraft is forced to take-off in overlapping timeslots where runway availability is severely limited due to the three arriving aircraft. During operational use, a time limit may be set to interrupt the search process for extremely complex situations. It should be noted, that situations of this complexity should not be encountered in practice – it was included here to show the performance limitations of the algorithm.

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., [7], [8], [9]). Most planning problems, however, are stated as constraint satisfaction problems (e.g., [2], [10], [11], [12]). Among the constraint satisfaction solutions applied to the area of Departure Management are RESO[10], and DSP[11]; a system including Departure Management in a broader scope is TARMAC[12].
Closest to our work is the Departure Manager Runway Event Sequence Optimizer (RESO) implemented by the National Air Traffic Services (NATS) [10]. This prototype application is aimed to de-conflict aircraft with similar departure times, minimizing 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.

The Departure Spacing Program (DSP) places a greater emphasis on flow management [11]. The DSP calculates departure schedules by coordinating 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.
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 [12]. The controller is assisted in his planning of the aircraft motions on the apron, especially with respect to pushback times of departing aircraft and taxi ways. The TARMAC planning unit supports the controller by visualizing future traffic situations, by recognizing planning conflicts and undetermined taxi sequences, and by automatically planning sequences in many safe situations.

7 Conclusions and Further Work

The departure scheduling prototype presented in this paper provides runway controllers with a decision support tool to establish 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 the prototype lies in its flexibility: any airport topology can be used, inbound traffic is taken into account, and certain take-off times or -orders can be fixed while others can be scheduled.

The performance our solution achieves is acceptable for practical application at airports such as Prague. A future step might be to evaluate the prototype for other airports, especially those where the topology and traffic load further increase the situational complexity. Performance may be enhanced by using constraint relaxation techniques in combination with the search time limit already implemented. When using such a scheme, acceptable but not optimal solutions could be offered to bind the search time for exceptionally complex situations.

We expect that the acceptance of the tool will be high, since runway controllers will remain involved in the scheduled process. They will be able to impose restrictions on a schedule beforehand, and make modifications to calculated solutions afterwards by replanning 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. 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)
2. 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)
3. 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)
4. FAA, Aeronautical Information Manual – Official Guide to Basic Flight Information and ATC Procedures, February 2000
5. ILOG Solver Reference Manual, version 5.0, August 2000, ILOG
6. ILOG Scheduler Reference Manual, version 5.0, July 2000, ILOG
7. 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)
8. 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
9. 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
10. Roberts, S., Foster, E.: (2001). D-MAN User Guide, EUROCONTROL DMAN/UG/001
11. Overmoe, C.: Requirements Document for Departure Spacing Program, Proof of Concept Phase II, FAA (1999)
12. 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)