Final version available at InformsPubsOnline: http://dx.doi.org/10.1287/trsc.2017.0750

Time-indexed formulations for the Runway Scheduling Problem

Pasquale Avella, Maurizio Boccia
DING — Dipartimento di Ingegneria — Università del Sannio, {avella,maboccia}@unisannio.it

Carlo Mannino
SINTEF ICT, Carlo.Mannino@sintef.no

Igor Vasilyev *
Matrosov Institute for System Dynamics and Control Theory of the Siberian Branch of the Russian Academy of Sciences, vil@icc.ru

The problem of sequencing and scheduling airplanes landing and taking off on a runway is a major challenge for air traffic management. This difficult real-time task is still carried out by human controllers, with little help from automatic tools. Several methods have been proposed in the literature, including Mixed Integer Programming (MIP) based approaches. However, in a recent survey (Bennell et al. (2011)) MIP is claimed to be *unattractive* for real-time applications, since computation times are likely to grow too large. In this paper we reverse this claim, by developing a MIP approach able to solve to optimality real-life instances from congested airports in the stringent times allowed by the application. In order to achieve this it was mandatory to identify new classes of strong valid inequalities, along with developing effective fixing and lifting procedures.

Key words: Air Traffic Management, Runway Scheduling Problem, Single Machine Scheduling, Mixed Integer Programming, Valid Inequalities, Fixing and Lifting Procedures.
History:

1. Introduction

In Air Traffic Management (ATM) the Tower Control is responsible for managing the immediate airport environment from the control tower. Tower controllers organize and expedite the flow of traffic on the ground and in the airspace adjacent to the airport. A major task of a controller is

* The research of I. Vasilyev is partially supported by the Russian Foundation for Basic Research, research project No. 14-07-00382

to avoid conflicts in the access to airborne and airdrome resources between airplanes that are currently under his/her responsibility. For instance, the runway can be occupied only by one airplane at a time. The controller in charge will make sure that take-offs and landings are separated in time sufficiently to respect this constraint. The controller can (to a certain extent) push forward or backward take-offs and landings and change the order of flights. Since separation times are sequence dependent, clearly any decision for one or more airplanes will affect neighboring ones, and consequently the entire sequence. It is apparent that for a human operator it is impossible to foresee or compute all the future effects of local decisions, which may lead to unwanted and unnecessary delays. Punctuality is one of major indicators of airline performance. Not surprisingly, in the past years, we experienced a growing interest in developing automatic decision support tools capable of assisting controllers in their difficult task. A number of sub-projects in the European aviation research framework SESAR joint undertaking (SESAR (2007)) were actually involved in such developments. Consequently, the interest of the scientific community has also grown significantly in the past years and the literature on this topic is already very vast. Concerning ATM at the airport, the practice and the literature typically consider three distinct problems: Departure Management (DMAN), Arrival Management (AMAN) and Surface Management (SMAN). DMAN (AMAN) amounts to establishing take-off times (landing times) for each departing (arriving) airplane, so as to minimize deviations from the official schedule, and avoid conflicts in the space near the airport and on the runway. SMAN determines the schedule of airplane movements in the airdrome according to (expected) landing times and requested take-off times, preventing conflicts on taxi-ways. Although in principle the three problems are tightly connected and should be solved jointly, already the stand-alone version of each problem is too complicated to be handled by a single controller. In the practice the responsibilities are further fragmented, and several controllers may be involved in the tasks associated with one problem.

In this paper we consider the integrated management of departures and arrivals on a single runway, and we will refer to this problem as RSP (Runway Scheduling problem). In RSP one wants to (jointly) schedule the take-offs and the landings of a set of airplanes. For each arrival flight, an arrival window is given, and the flight must land in this time window. Similarly, for each departure flight, a departure window is given: however the departure can be canceled (at high cost). Two successive flights on the runway must be separated by a minimum time interval which depends on the involved airplanes. The *official timetable* provides requested arrival and departure times. However, when one or more airplanes are delayed, a new schedule must be found, so that (some measure of) the deviation from the official timetable is minimized.

For details on the different approaches for RSP we refer the reader to a recent survey Bennell et al. (2011). The great majority of the approaches presented in the literature are heuristic or

meta-heuristic - see for instance Beasley et al. (2001), Atkin et al. (2007) and again Bennell et al. (2011) as well as the literature discussion in the recent papers by Lieder et al. (2015) and Furini et al. (2015).

Most likely because of the stringent computing times allowed by the real-time nature of the application, only a few papers on exact methods have appeared in the literature. These methods are basically of two types: dynamic programming approaches, and Mixed Integer Programming (MIP) based approaches.

Recently, a dynamic programming algorithm has been first proposed and then successfully implemented and tested in Briskorn and Stolletz (2014) and Lieder et al. (2015), respectively. In order to tame combinatorial explosion in their dynamic programming approach to AMAN, Balakrishnan and Chandran (2010) and Furini et al. (2014) adopted *constrained position shifting*, where the position of the airplane in the final sequence shall not deviate significantly from a given initial position. Observe that constrained position shifting allows for drastic reductions in the number of potential dynamic states to enumerate. However, a better sequence may exist which violates one or more such positional constraints. In contrast, in a recent paper by De Maere and Atkin (2015), no positional constraints are imposed; instead, a number of smart exact fixing/pre-processing ideas are put in place to reduce computational times.

As for MIP approaches, we observe first that RSP can be interpreted as a classical single machine scheduling problem with sequence dependent setup times and earliness/tardiness objective function (see Nogueira et al. (2014)). The runway corresponds to the machine, flights correspond to jobs and time separations between flights on the runway to setup times. In contrast with traditional machine scheduling instances, runway separations do not necessarily satisfy the triangle inequality. Two types of formulations have traditionally competed in the literature on machine scheduling (see Queyranne and Schulz (1994), Nogueira et al. (2014)), namely *big-M* formulations and Time Indexed (TI) formulations. More specifically, in *big-M* formulations for DMAN (AMAN), the departure (arrival) time of a flight is represented by a single continuous variable. *Big-M* formulations for AMAN and DMAN are adopted in Abela et al. (1993), Beasley et al. (2000), Briskorn and Stolletz (2014), Furini et al. (2015). In principle, MIP based approaches can provide the optimal solution to the original problem. However, in order to meet computing time limits for instances of practical interest, *big-M* MIP approaches typically need to resort to some sort of heuristic decomposition of the problem (for instance, *rolling horizon*, see e.g. Furini et al. (2012) and Samà et al. (2013)), and the overall algorithm boils down to a heuristic method.

In TI formulations, the time horizon is discretized in small time periods. The schedule of a given flight (job) is modeled by a set of binary variables, each associated with a feasible departure or arrival time period. Only one of such variables will be 1 in a feasible solution, i.e. the one identifying

the scheduled time. TI formulations return much stronger bounds than *big-M* formulations (see, e.g. Sousa and Wolsey (1992), Van den Akker et al. (2000)), but at the cost of increasing computing times due to the large number of variables and constraints. Besides providing strong lower bounds, it has been observed (e.g. Uma and Wein (1998), Savelsbergh et al. (2005), Masin and Raviv (2014)) that the LP solutions of TI formulations can be effectively exploited in heuristic approaches to generate feasible solutions.

We want to remark that in our approach, time step size is part of the input. Of course, given the original "time-continuous" instances, one may adopt very different step sizes. Larger step sizes correspond to smaller time-indexed formulations and consequently faster computations. However, the final solution may be suboptimal (with respect to the optimal continuous time solution); or, even worse, feasible instances may become infeasible (see, e.g., Harrod (2011)). It is not difficult to see that, if separations are integer (time units), then a time step size of 1 time unit would ensure that the final solution is also optimal for the original problem. However, in many practical contexts, larger sizes can still provide satisfactory solutions, allowing for drastic reductions in computing times. For example, in Furini et al. (2012) the step size is 60 seconds for all instances. In Heidt et al. (2013), in the same instance the authors resort to two distinct step sizes, namely 5 seconds for (the first) 10 airplanes and 75 seconds for the remaining ones. Remarkably the quality of the 10 seconds step-size has been assessed during an official validation campaign (see Kjenstad et al. (2013a)) in the context of SESAR joint undertaking (SESAR (2007)). The solutions from the time-indexed formulation were simulated and compared against the solutions obtained by experienced controllers on the same instances, exhibiting significant improvements for all performance indicators.

TI formulations for AMAN and DMAN have recently been presented, e.g., in Heidt et al. (2013), Kjenstad et al. (2013a,b). One major challenge when adopting TI formulations for scheduling problems consists in limiting the number of variables and constraints. As shown in Kjenstad et al. (2013b) and Lieder et al. (2015), in RSP the time windows associated with the flights naturally limit the size of the resulting formulations. Note that MILP formulations — and in general all exact approaches — can also be a tool of designing heuristics. In particular, this happens when the enumeration search halts before the entire enumeration tree has been visited, in which case the current best incumbent is returned. Moreover, in TI formulations, as discussed above, another source of approximation derives from the choice of the time step size.

Let G be an undirected graph, whose nodes are in one-to-one correspondence with the binary variables of the TI formulation and whose edges corresponds to pairs of conflicting variables (i.e. that cannot be simultaneously at 1 in any feasible solution). The polyhedron associated with a TI formulation is contained in the *stable set* polyhedron (see Wolsey and Nemhauser (2014)), that is the polyhedron associated with the stable sets of G. The polyhedral properties of TI formulations for

the single machine scheduling problems with release dates has deeply been investigated in Waterer et al. (2002). However, such results cannot directly be applied to runway sequencing problems because set-up times are sequence-dependent. A standard way to tighten stable set formulations is by including suitable clique inequalities (Wolsey and Nemhauser (2014)), namely inequalities associated with the cliques of G. Alternative TI formulations may be obtained by considering different cliques. Because of the limited time available for solving the Runway Scheduling Problem, selecting a suitable formulation is a critical issue. On the one hand, one would want to include many clique constraints, so as to obtain tighter bounds. On the other hand, the number of clique constraints should not grow too much, so as to control the computational burden for solving the linear relaxation of the formulation.

In this paper we present a novel technique to mediate between these two contrasting goals. This task is carried out by identifying a TI formulation which compromises between *i*) the quality of the LP bound and *ii*) the "compactness" of the formulation, measured as number of rows and nonzeroes. Our TI formulation is based on a new class of valid inequalities for the single machine scheduling problem with sequence dependent set-up times. This new family generalizes a family recently introduced by Nogueira et al. (2014). The new generalization allowed us to significantly improve the quality of the lower bounds and reduce the number of constraints with respect to Nogueira et al. (2014) on our instances of RSP. The novel TI formulation is then solved by standard column generation techniques. The combination of these factors eventually allowed us to find the solution of difficult instances from large airports in Europe, namely Stockholm Arlanda, Hamburg and Milano Linate, within the stringent time limits dictated by real-time requirements. Our new approach exploits the fact that the number of different set-up times is small with respect to the number of jobs. This feature occurs in many practical contexts in production management. However, the approach is not effective when the number of distinct set-up times grows.

2. Problem statement and basic formulation

We denote by L and D the set of arrival and departure flights, respectively, hereafter simply presented as *arrivals* and *departures*, and we let $F = L \cup D$ be the set of all the flights. Overall in the paper we assume $|F| \geq 2$, otherwise the problem is trivial. Arrivals and departures take place on a single runway during the time horizon T, which we assume to be discretized in equally sized periods, with $T = \{1, \ldots, |T|\}$. Period $t \in T$ starts at time t and ends at time $t+1$ — so period t is the half-open interval $[t, t+1)$. When saying that a flight arrives/lands/departs/takes off at time t we intend that this occurs at the beginning of time period t.

Each arrival (departure) or landing (take-off) may happen only within a given time window — typically narrower for arrivals. Departures may be dropped at very high cost, while arrivals

must always land (within their time windows). Conventionally, we assume that the arrival time of an arrival flight coincides with its landing time. Similarly, the departure time of a departure flight coincides with its take-off time. For each flight $i \in D$ ($i \in L$), let $l_i, u_i \in T$ be, respectively, the earliest, and the latest departure (arrival) time period; also, we let $h_i \in T$ be the requested departure (arrival) time. The set of contiguous time periods $T_i = \{l_i, l_i + 1, \ldots, u_i\} \subseteq T$ is the *time window* for $i \in F$.

For each ordered pair of distinct flights $(i, j) \in F \times F$, a minimum *separation* time $s_{ij} > 0$ is required, that is if i precedes j on the runway and j arrives/departs at time t, i must arrive/depart in $\{0, 1, \ldots, t - s_{ij}\}$. Let $F' \subseteq F$. A (feasible) runway schedule for F' is a time $\gamma_i \in T_i$ for each flight $i \in F'$ such that for each pair (i, j) of distinct flights in F' with $\gamma_i \leq \gamma_j$, we have $\gamma_j - \gamma_i \geq s_{ij}$. For each flight $i \in D$ ($i \in L$) let q_{it} be the cost of departing (landing) i at time $t \in T_i$. For each departure $i \in D$, let c_i be the cost of dropping i.

The *Runway Sequencing Problem* (RSP) for a set of flights $F = D \cup L$ consists in finding a set of dropped departures $\tilde{D}$ and a feasible schedule γ for the remaining flights $F \setminus \tilde{D}$ so that the total cost $\sum_{i \in \tilde{D}} c_i + \sum_{i \in F \setminus \tilde{D}} q_{i\gamma_i}$ is minimized.

Time-indexed formulations. In order to formulate and solve RSP by integer programming, we associate a binary variable x_{it} with every $i \in F$ and every $t \in T_i$, which is 1 if and only if $\gamma_i = t$, i.e. i arrives/departs at time t. Also, with every departure $i \in D$ we associate a binary variable y_i which is 1 if i is dropped and 0 otherwise. Since every flight is assigned (at most) one arrival/departure time in a feasible schedule, for every $i \in F$ and every $k, l \in T_i$, $k \neq l$, we have $x_{ik} + x_{il} \leq 1$.

Consider now two distinct flights $i, j \in F$, and assume that the assignment $\gamma_i = k$ and $\gamma_j = l$ violates the separation requirement between i and j, that is $-s_{ji} < l - k < s_{ij}$ [1]. Then, we have either $x_{ik} = 0$ or $x_{jl} = 0$ in any feasible solution. In turn, this can be expressed by the constraint $x_{ik} + x_{jl} \leq 1$ and we say that the pair (of indices) $\{ik, jl\}$ is an *incompatible pair*. For an instance of RSP, we let I be the set of all incompatible pairs (of indices).

With an instance of RSP, we associate an undirected simple graph $G(V, E)$ called *conflict graph*. The nodes of G are in one-to-one correspondence to the x variables of the formulation and it has an edge between two nodes whenever the associated variables cannot both possess the value of 1. More formally, we let

$$V = \{it : i \in F, t \in T_i\}$$

[1] The schedule $\gamma_i = k$ and $\gamma_j = l$ is unfeasible if i) either $k < l$ (i precedes j on the runway) and $l - k < s_{ij}$; or ii) $l < k$ (j precedes i on the runway) and $k - l < s_{ji}$. Recall that $s_{ij} > 0$ for all $\{i, j\} \subseteq F$.

and

$$E = I \cup \{\{ik, il\} : ik, il \in V, k \neq l\}.$$

From the above discussion, it follows that x represents a feasible schedule, if and only if x satisfy:

$$x_{ik} + x_{jl} \leq 1, \qquad \{ik, jl\} \in E. \tag{1}$$

A *clique* of an undirected graph is a subset of the nodes such that every two nodes in the subset are adjacent. Incidentally, observe that any pair of adjacent nodes is also a clique (of cardinality 2). Let K be a clique of the conflict graph $G(V, E)$, and let x satisfies (1) then it is easy to see that x also satisfies the *clique inequality*:

$$\sum_{it \in K} x_{it} \leq 1. \tag{2}$$

If $K \subseteq V$ is a clique and $u, v \in K$, then the edge (u, v) is said to be *covered* by K. An *I-cover* is a set of cliques $K_1, K_2, \ldots$, such that every edge in I is covered by at least one clique in the set (a trivial I-cover is the one where the cliques correspond to edges in E). Let $\mathcal{K} = \{K_1, K_2, \ldots\}$ be a I-cover. It is not difficult to see that x satisfies (1) if and only if x satisfies the system of inequalities:

$$\begin{aligned} &(i) \quad \sum_{l \in T_i} x_{il} \leq 1, && i \in F, \\ &(ii) \quad \sum_{it \in K} x_{it} \leq 1, && K \in \mathcal{K}. \end{aligned} \tag{3}$$

Let c_i and $q_{it} = |t - h_i|$ be the cancellation cost and the coefficient measuring the earliness/tardiness of the flight i with respect to its expected arrival/departure time h_i, respectively. We are finally able to write a binary linear programming formulation of RSP:

$$\begin{aligned} &\min \sum_{i \in D} c_i y_i + \sum_{i \in F} \sum_{t \in T_i} q_{it} x_{it}, \\ &(i) \qquad \sum_{t \in T_i} x_{it} = 1, \ i \in L, \\ &(ii) \qquad \sum_{t \in T_i} x_{it} + y_i = 1, \ i \in D, \\ &(iii) \qquad \sum_{it \in K} x_{it} \leq 1, \ K \in \mathcal{K}, \\ &(iv) \qquad x_{it} \in \{0, 1\}, \ i \in F, t \in T_i \quad y_i \in \{0, 1\}, \ i \in D. \end{aligned} \tag{4}$$

where $\mathcal{K} = \{K_1, K_2, \ldots\}$ is a I-cover and the $(4.iii)$ define an *I-cover system of inequalities*.

Constraints $(4.i)$ ensure that every arrival is assigned an arrival time from its time window, whereas constraints $(4.ii)$ ensure that every departure is either dropped or assigned a departure

time from its time window. Finally, constraints $(4.iii)$ are the I-cover inequalities which ensure that the schedule respects separation constraints. The objective function represents the overall cost of a solution. Observe that constraints $(3.i)$ are implied by $(4.i)$ and $(4.ii)$, whereas $(3.ii)$ are precisely $(4.iii)$.

The linear relaxation of above time-indexed formulation can be exploited to generate bounds in Branch&Bound schemes. As discussed in the introduction, such bounds are typically quite tight.

In order to keep the number of constraints at bay, it is important to carefully select the cliques in the I-cover[2]. In fact, typically most of the constraints (in real-life instances) of (4) will belong to the I-cover system of constraints $(4.iii)$.

Building "small" clique covers. One heuristic way to obtain I-covers with "few" cliques, is to try to cover the edges of G with large cliques[3].

For single machine scheduling problems with sequence dependent setup times, one of the original and well studied example of I-cover system of inequalities (see, e.g., Nogueira et al. (2014) and Sousa and Wolsey (1992)) is given by the family of inequalities:

$$x_{jt} + \sum_{k \in T_i \cap [t-s_{ij}+1,t]} x_{ik} \leq 1, \qquad i,j \in F, i \neq j, t \in T_j. \tag{5}$$

A pictorial representation of a clique of the conflict graph associated with a generic constraint of type (5) is given in Figure 1, where the nodes of the conflict graph are drawn on a grid. In particular, node ik appears in row i and column k. The clique associated with constraint (5) involves only one node of row j (namely node jt) and the nodes ik of row i whose columns are in the range $k \in \{t - s_{ij} + 1, \ldots, t\}$.

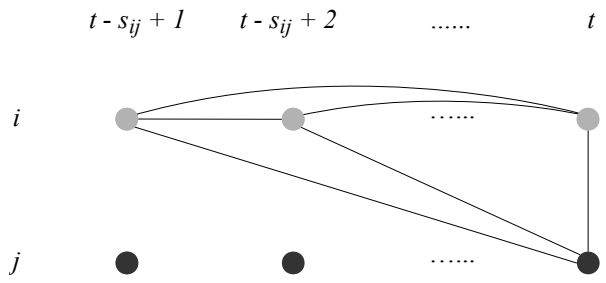

Figure 1 A clique associated with (5)

[2] Note that while an I-cover can contain a polynomial number of cliques, the number of distinct I-covers can grow exponentially with the number of nodes of G

[3] It is important to observe that one may want to have additional cliques in order to further strengthen the formulation, but this is a different perspective and we do not consider it here

Note that each clique inequality of system (5) can be strengthened by lifting in a trivial fashion, giving the following system:

$$\sum_{l \in T_j \cap [t-s_{ji}+1,t]} x_{jl} + \sum_{k \in T_i \cap [t-s_{ij}+1,t]} x_{ik} \leq 1, \qquad i,j \in F, i<j, t \in T_j \cup T_i. \tag{6}$$

Figure 2 shows a clique of type (6).

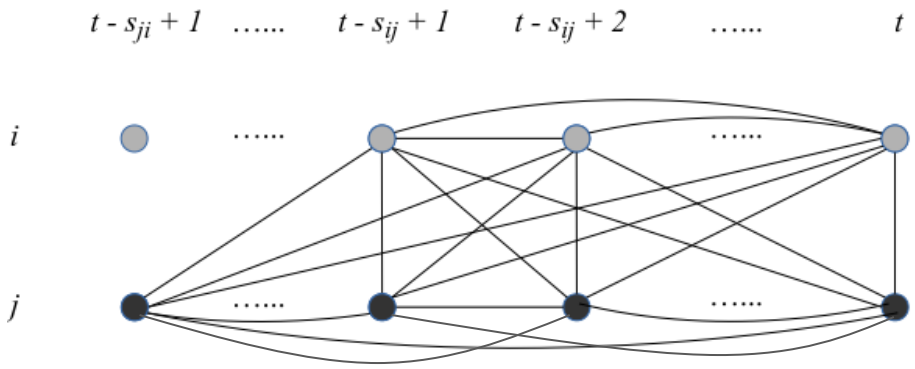

Figure 2 A clique associated with the lifted inequality (6)

Hereinafter we call *Basic RSP formulation* the formulation for RSP defined by (4), where the I-cover constraints (4.*iii*) are of type (6). We will show in Section 5 that, even if the lifted family provides better bounds than (5), yet it is not sufficient to solve all instances in our real-life testbed within the stringent running times imposed by real-time.

3. Stronger families of clique inequalities

A family of clique inequalities valid for (4) has recently been introduced by Nogueira et al. (2014). In what follows, for all $S \subseteq F$, $|S| \geq 2$, and all $i \in S$, we let $s_i(S) = \min\{s_{ij} : j \in S \setminus i\}$.

PROPOSITION 1 **(Nogueira et al. (2014))**. *The following is a valid system of clique inequalities for (4):*

$$\sum_{i \in F} \sum_{l \in [t-s_i(F)+1,t] \cap T_i} x_{il} \leq 1, \qquad t \in T. \tag{7}$$

We refer to the formulation (4) where the I-cover system (4.*iii*) is given by (6) and (7) as the *Nogueira formulation*.

With the aim of defining a stronger but also more compact time-indexed formulation, we introduce a new family of clique inequalities — that we call (S,t)-*clique inequalities* — generalizing (7):

PROPOSITION 2. *Let $t \in T$ and let $S \subseteq F$,with $|S| \geq 2$. The (S,t)-clique inequality:*

$$\sum_{i \in S} \sum_{l \in [t - s_i(S) + 1, t] \cap T_i} x_{il} \leq 1 \tag{8}$$

is valid for (4).

Proof. It follows directly from Proposition 1 and from the trivial observation that any constraint, which is valid for a subset of flights and time periods, is also valid for the larger sets.

Figure 3 shows a (S,t)-clique on the conflict graph. The clique involves the nodes in the range $[t - s_i(S) + 1, t]$ for each row $i \in S$.

It is easy to see that (8) generalize (6) by letting $S = \{i, j\}$ and (7) by letting $S = F$. Also observe that, since for any pair i, j of distinct flights the separation s_{ij} is strictly positive, for any set of flights S with $|S| \geq 2$ and any $i \in S$ we have $s_i(S) \geq 1$.

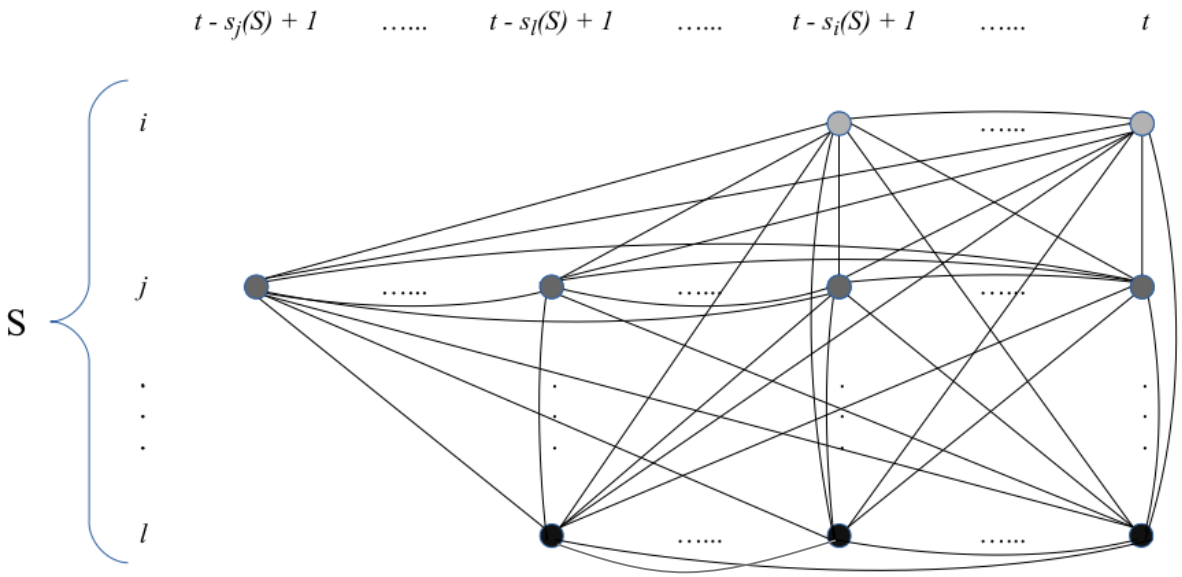

Figure 3 **A (S,t)-clique (8)**

The following proposition shows when the (S,t)-clique inequalities (8) define an I-cover system.

PROPOSITION 3. *Let $\mathcal{S} = \{S^0, S^1, \ldots S^h, \ldots\}$ be a family of non-empty subsets of F, each of cardinality at least 2. The inequalities*

$$\sum_{i \in S} \sum_{l \in [t - s_i(S) + 1, t] \cap T_i} x_{il} \leq 1, \ S \in \mathcal{S}, \ t \in T \tag{9}$$

define an I-cover system (and $\mathcal{S}$ is an I-cover of the conflict graph G) if for each $i, j \in F$ $(i \neq j)$ there exists $S \in \mathcal{S}$ such that $i, j \in S$ and $s_{ij} = s_i(S)$.

Proof. Suppose not. Then there exists an incompatible pair $\{fk, gr\} \in I \subseteq E$ such that the sum $x_{fk} + x_{gr}$ of corresponding time-indexed variables is not contained in any (S,t)-clique inequality of (9). Observe that since $\{fk, gr\} \in E$, then $fk, gr \in V$, which in turn implies that $k \in T_f$ (i.e. k is a feasible time period for the flight f) and $r \in T_g$ (i.e. r is a feasible time period for the flight g). We may assume with no loss of generality that $r \geq k$. Then, by definition of I, $\{fk, gr\} \in I$ implies $r - k < s_{fg}$. Let $\bar{S} \in \mathcal{S}$ be such that $f, g \in \bar{S}$ and $s_{fg} = s_f(\bar{S})$. Let us consider the $(\bar{S}, r)$-clique inequality of type (8), i.e. the (S,t)-clique inequality associated with $\bar{S}$ and time period $t = r$. Since $g \in \bar{S}$, $s_g(\bar{S}) \geq 1$ and $r \in T_g$, we have that $r \in [r - s_g(\bar{S}) + 1, r] \cap T_g$ and the variable x_{gr} appears in the $(\bar{S}, r)$-clique inequality. Since $f \in \bar{S}$, $s_f(\bar{S}) = s_{fg}$, $k \in T_f$, $k > r - s_{fg}$ we have that $k \in [r - s_{fg} + 1, r] \cap T_f$ and the variable x_{fk} appears in the $(\bar{S}, r)$-clique inequality. So both x_{fk} and x_{gr} appear in the $(\bar{S}, r)$-clique inequality, a contradiction.

We will refer to the I-cover system of inequalities (9) as the (S,t)-*I-cover system.*

The next sequential algorithm 1 constructs a (S,t)-I cover system which will represent our I-cover system (4.*iii*). In what follows, we say that an (S,t)-inequality (8) *covers* an incompatible pair $\{ik, jl\} \in I$ if both variables x_{ik} and x_{jl} appear in the inequality. The algorithm keeps at each iteration a family $\mathcal{S} = \{S^0, S^1, \dots\}$ of subsets of F. Let $\mathcal{Q}(\mathcal{S})$ be the corresponding set of (S,t)-inequalities (9). If the constraints in $\mathcal{Q}(\mathcal{S})$ cover all incompatible pairs in I, we are done. Otherwise, let $\bar{I}(\mathcal{S}) \subseteq I$ be the set of the incompatible pairs not covered by any inequality in $\mathcal{Q}(\mathcal{S})$. Then a new subset $R \subseteq F$ is identified, so that at least one uncovered pair in $\bar{I}(\mathcal{S})$ is covered by one of the (R,t)-inequalities, for some t. Then R is included in $\mathcal{S}$, and the algorithm iterates.

Algorithm 1 Greedy (S,t)-I-cover

0: Let $\mathcal{S} := \{F\}$.
1: **while** $\bar{I}(\mathcal{S})$ is non-empty **do**
2: Find a set $R \subseteq F$ with the property that the associated (R,t) inequalities, for $t \in T$, cover some pairs in $\bar{I}(\mathcal{S})$.
3: Let $\mathcal{S} := \mathcal{S} \cup \{R\}$.
4: **end while**
5: **return** The set $\mathcal{S}$.

There are of course many ways to carry out Step 2. Recall that our goal is to generate as few inequalities as possible in order to keep the overall formulation very compact. This is in general a difficult task, so we limit ourselves to find at Step 2 a set R which maximizes the number of freshly covered incompatible pairs. To this purpose we introduce a suitable binary linear program.

Suppose now $\bar{I}(\mathcal{S}) \neq \emptyset$, and let $\{ik, jr\} \in \bar{I}(\mathcal{S})$. Without loss of generality we may assume $r \geq k$. By Proposition 3, the corresponding *uncovered* ordered pair of flights $(i,j) \in F \times F$ is such that $s_{ij} > s_i(S)$ for every $S \in \mathcal{S}$. We will identify a set $R \subseteq F$ such that for at least one uncovered pair (i,j) we have $s_{ij} = s_i(R)$. Let $C(\mathcal{S}) = \{(i,j) \in F \times F : (i,j) \text{ uncovered}\}$. Next, for $i \in F$, we introduce a binary variable z_i, which is 1 if and only if $i \in R$. Also, with each ordered pair of distinct flights $(i,j) \in C(\mathcal{S})$ we associate a binary variable y_{ij} which is 1 only if the ordered pair (i,j) is *covered* by R, namely if $s_{ij} = s_i(R)$.

So, R can be constructed by solving the following MIP:

$$
\begin{aligned}
& \max \sum_{(i,j)\in C} y_{ij} \\
(i) \quad & y_{ij} \leq z_i, \quad (i,j) \in C(\mathcal{S}), \\
(ii) \quad & y_{ij} \leq z_j, \quad (i,j) \in C(\mathcal{S}), \\
(iii) \quad & y_{ij} + z_k \leq 1, \quad (i,j) \in C(\mathcal{S}),\ k \in F :\ s_{ij} > s_{ik}, \\
(iv) \quad & z_i \in \{0,1\}, \ \ i \in F, \ \ y_{ij} \in \{0,1\}, \ \ (i,j) \in C(\mathcal{S}).
\end{aligned} \tag{10}
$$

PROPOSITION 4. *Let $\bar{z}, \bar{y}$ be a feasible solution to (10) and let $\bar{R} \subseteq F$ be the corresponding set of flights. Let $\bar{y}_{pq} = 1$. Then (p,q) is covered by $\bar{R}$.*

Proof. By contradiction, assume that (p,q) is not covered by $\bar{R}$. Since $\bar{y}_{pq} = 1$ we have that $\bar{z}_p = \bar{z}_q = 1$ and then $(p,q) \in \bar{R}$. Since (p,q) is not covered we have $s_p(\bar{R}) < s_{pq}$. So, there exists $r \in \bar{R}$ such that $s_p(\bar{R}) = s_{pr} < s_{pq}$. But then $\bar{z}_r = 1$ and by $(10.iii)$ $\bar{y}_{pq} + \bar{z}_r \leq 1$ and we have $\bar{y}_{pq} = 0$, a contradiction.

In the following we will refer to the formulation defined $(4.i)$, $(4.ii)$ and the (S,t)-clique cover system resulting from the Greedy (S,t)-I cover algorithm 1 as the (S,t)-*clique formulation.*

Detecting dominated cliques. For any $t \in T$, we say that a (S,t)-clique inequality dominates a (H,t)-clique inequality if $H \subseteq S$. Because of the way the cover is generated by the algorithm *Greedy (S,t)-I-cover* by the MIP problem (10) many dominated cliques are likely to be generated. Removing these cliques is crucial to speed up the overall solution algorithm. The following proposition provides a sufficient condition for domination between the constraints associated with a clique cover.

PROPOSITION 5. *Let H and S be two subsets of flights. For any $t \in T$, the (H,t)-clique inequality is dominated by the (S,t)-clique inequality if the following two conditions are satisfied:*

i) $H \subseteq S$,

ii) $s_i(H) \leq s_i(S)$ *for each* $i \in H_t$.

We modify Algorithm Greedy (S,t)-I-Cover by testing the conditions of Proposition 5 in Step 3 before including a new clique in the current clique cover.

Lifting. Observe that (S,t)-clique inequalities are not necessarily maximal. They can be strengthened by lifting each (S,t)-clique in a sequential way. This is done by the simplistic greedy algorithm of Kopf and Ruhe (1987). Namely, given a (S,t)-clique K, first the set $P \subseteq V \setminus K$ of the nodes which are adjacent to every node in K are identified. Then the node of P with maximum degree is added to K and the process iterates until P is empty.

4. The overall algorithm

The algorithm consists of three phases: first a logical presolve attempts to reduce the size of the formulation; then the LP-relaxation of the resulting formulation is solved with the standard column generation, providing a lower bound on the optimal cost; finally a feasible integer solution to (4) is computed over the subset of variables generated in the previous phase. By comparing the upper bound and the lower bound, the optimality of this solution can be assessed.

4.1. Presolve

The presolve phase consists of three fixing procedures to reduce the size of the time windows associated with the flights.

Fixing 1. This fixing mechanism exploits the fact that arrivals typically have quite narrow time windows and they cannot be dropped. So, let $i \in L$ be an arrival and let $[l_i, u_i]$ be the associated landing window. If another flight $j \in F$ precedes i on the runway, then j must clear the runway early enough to let i land; how much earlier, in turn, depends on the minimum time separation s_{ji}, and on the latest time u_i in which i can land. A symmetric argument can be used when a flight j occupies the runway *after* an arrival flight i. Observe that in the following proposition, the real interval $[a,b] \subset R_+$ can be empty (i.e., when $a > b$).

PROPOSITION 6. *Let* $i \in L$ *be an arrival flight and let* $j \in F$ *be any other flight. Let* t_j *be the time when* j *occupies the runway (namely,* j *lands or takes-off). Then* $t_j \notin [u_i - s_{ji} + 1, l_i + s_{ij} - 1]$ *in any feasible solution. Consequently, we can fix* $x_{jt} = 0$*, for each* $t \in [u_i - s_{ji} + 1, l_i + s_{ij} - 1]$.

Proof. By contradiction assume that $t_j \in [u_i - s_{ji} + 1, l_i + s_{ij} - 1]$. Let t_i be the time i lands, then $t_i \in [l_i, u_i]$. We consider two cases:

i) j *occupies the runway before flight* i *lands.* Then, since $t_j \geq u_i - s_{ji} + 1$, we have $t_i \geq t_j + s_{ji} \geq u_i + 1$, and i lands outside its arrival window, a contradiction.

ii) j *occupies the runway after* i *lands.* Then, since $t_j \leq l_i + s_{ij} - 1$, we have $t_i \leq t_j - s_{ij} \leq l_i - 1$ and i lands outside its arrival window, a contradiction.

Note that the above fixing has an impact only if the interval $[u_i - s_{ji} + 1, l_i + s_{ij} - 1]$ is non empty. By simple arithmetics one sees that this happens if $u_i - l_i < s_{ij} + s_{ji} - 1$.

Fixing 2. Next, we also applied the tightening procedure for time windows, proposed in Ascheuer et al. (2001) for TSPTW. To this end, let $i \in F$ be any flight and let $\bar{L}_i = \{j \in L : u_j < l_i + s_{ij}\}$ be a subset of arriving flights which must land before i. Moreover let $\hat{L}_i = \{j \in L : l_j + s_{ji} > u_i\}$ be a subset of arriving flights which must land after i. We can tighten the time window of i by setting:

$$l_i = \max\left\{l_i, \min_{j \in \bar{L}_i}\{l_j + s_{ji}\}\right\} \tag{11}$$

$$u_i = \min\left\{u_i, \max_{j \in \hat{L}_i}\{u_j - s_{ij}\}\right\} \tag{12}$$

Fixing 3. Finally, let $i \in F$, $r \in [l_i, u_i]$ and assume that ir does not appear in (any pair of) the incompatible set I. In other words, flight i can arrive/depart at time r without affecting any other flight. As a consequence, recalling that q_{ir} is the cost of landing/take-off at time r (for flight i), i will not land/take-off in any time $t \in [l_i, u_i]$ such that $q_{it} \geq q_{ir}$, and the corresponding variables x_{it} can be fixed to 0.

4.2. Computing lower and upper bounds

The presolve phase boils down to compute reduced time windows for the flights: in what follows, V will denote the set of residual columns.

We compute a lower bound by solving to optimality the LP-relaxation of the (S,t)-clique formulation (4). However, even with reduced time windows, its size is typically too large to be solved to optimality within the stringent time limits imposed for the instances in our testbed. To overcome this difficulty, we approach the LP-relaxation by a standard column generation scheme (see Wolsey (1998)), where a *master problem* containing only a subset of the variables is solved to optimality, and then a *pricing problem* is solved to detect whether some of the external (i.e. not included into the current master problem) variables have negative reduced costs. External variables with negative reduced costs are added to the master problem and the algorithm iterates until all the external columns have nonnegative reduced costs. At this point, the solution to the current master program is optimal for the LP-relaxation of the entire (S,t)-clique formulation. We now give a generic description of the column generation method. We remark that the method can be applied to any formulation.

The way the initial set $\bar{V}$ is chosen at Step 0 depends on the class of instances and will be discussed in Section 5. The reduced cost c_v at Step 3 can be computed in the standard fashion. In particular, let us denote by μ_l, for $l \in L$, the dual variables associated with constraints $(4.i)$;

Algorithm 2 Column generation scheme

0: Choose a suitable subset of columns $\bar{V} \subseteq V$.

1: Let the current master problem be obtained from the original formulation by dropping the columns in $V \setminus \bar{V}$.

2: Solve the LP-relaxation of the current master problem.

3: *Pricing*: compute the reduced costs $\bar{c}_v$ for all $v \in V \setminus \bar{V}$. If all reduced costs are non-negative STOP.

4: Include in $\bar{V}$ all columns with negative reduced cost and GOTO 1.

by μ_d, for $d \in D$, the dual variables associated with constraints $(4.ii)$; and by δ_K, for $K \in \mathcal{K}$, the dual variables associated with constraints $(4.iii)$. If we let $v = it$ (for all $i \in F$ and all $t \in T_i$), then $\bar{c}_v = q_{it} - \mu_i + \sum_{K: v \in K} \delta_K$.

To speed up the solution process, observe that the above column generation scheme still converges to the optimal solution of the original formulation if, at Step 3, we replace the reduced cost $\bar{c}_v$ with a lower bound $\tilde{c}_v \leq \bar{c}_v$. In particular, since $\delta_K \geq 0$ for all $K \in \mathcal{K}$, we may take $\tilde{c}_v = q_{it} - \mu_i$ for all $v = it$ with $i \in F$, $t \in T_i$. So, in our column generation, Step 3 is replaced by the following

3′: *Pricing*: compute the approximated reduced cost $\tilde{c}_v$ for all $v \in V \setminus \bar{V}$. If all approximated reduced costs are non-negative, STOP.

Computing a feasible solution. The column generation algorithm terminates with the (current) master program defined by the subset $\bar{V}$ of the original columns V. Let $LB(\bar{V})$ be the associated lower bound computed by solving the linear relaxation of the master program. A feasible integer solution $\bar{x}$ to (4) is then computed by solving this program to (integer) optimality by invoking a MIP solver. In other words, $\bar{x}$ is the best possible solution when only the columns in $\bar{V}$ are used.

Trivially, the quantity $UB(\bar{V}) = q^T \bar{x}$ provides an upper bound on the optimal solution value to (4). If $LB(\bar{V}) = UB(\bar{V})$, then $\bar{x}$ is the global optimal solution of the original RSP.

Even when $LB(\bar{V}) < UB(\bar{V})$, the global optimality of $\bar{x}$ can still be proven by reduced cost fixing (see Wolsey (1998)). The argument goes as follows. Suppose $\bar{x}$ is not globally optimal, and let x^* be such that $q^T x^* < q^T \bar{x}$. Let $v \in V \setminus \bar{V}$ be one of the "external" columns. Suppose that $LB(\bar{V}) + \tilde{c}_v \geq UB(\bar{V})$. Then $x^*_v = 0$: in other words, column v cannot belong to a solution which is better than $\bar{x}$. So, if $LB(\bar{V}) + \tilde{c}_v \geq UB(\bar{V})$ for all $v \in V \setminus \bar{V}$, then $\bar{x}$ is globally optimal.

5. Computational results

In this section we demonstrate the quality of the (S,t)-clique formulation over a set of real-life or realistic instances. In the first series of experiments we compare three different versions of

the formulation (4), corresponding to different definitions of the I-cover system (4.*iii*): the *Basic* formulation, where we use the clique inequalities (6); the *Nogueira* formulation, with the clique inequalities (7); and our (S,t)-clique formulation, defined by the (S,t)-clique inequalities (8). In the second series we compare with another approach proposed in the literature.

For our test instances, we tested steps of different size, ranging from 1 to 10 seconds.

In our test-bed, we consider two sets of instances:

i) the set AH, which consists of six real-life or realistic instances from the airports of Arlanda (Stockholm) and Hamburg. Stockholm Arlanda is the largest airport in Sweden. The instances are based on historical data from the Swedish Air Navigation Service Provider (ANSP). Hamburg instances were generated at the University of Salzburg using a realistic simulated environment of the Hamburg airport based on the NAVSIM simulator (Graupl et al. (2012)).

For each instance the time horizon was partitioned into time slots of different size ranging from 0.5 to 10 secs. Increasing the sizes of time slots can make the instances infeasible and we only report for the slot sizes for which the problem admits a feasible solution.

All the benchmark AH instances are available at SINTEF upon request to the authors of this article.

ii) The sets A and C were previously considered in Furini et al. (2015). The set A consists of 12 real instances corresponding to two simulation days at Milano Linate Airport. The set C consists of 15 larger instances generated by merging the instances of the set A. For such instances, time slot was set to one minute, early departures/arrivals as well as cancellation of departures are not allowed and there are no time windows for departures and arrivals. The objective function consists in minimizing the "tardiness" with respect to the expected departure/arrival time. Set A is publicly available at *www.or.deis.unibo.it/research.html*. The test instances of the set C as well as detailed computational results for both sets A and C were kindly provided to us by the authors of Furini (2015).

The computational experiments were carried out on a Intel Core(TM) i7-3770 CPU 3.40 GHz workstation with 32 Gb RAM. The code was written in C++ and the Mixed-Integer Programming solver was Cplex 12.6, used with the default settings.

5.1. Results for the AH instances

In Table 1 we detail the instance data. For each instance, columns $|D|$, $|L|$ and $|F|$ are the number of departures, the number of landings and the total number of flights ($|F| = |D| + |L|$), respectively. Columns *Min Slot* and *Max Slot* are the smallest and the largest time discretization intervals (in secs.) considered in the experiments. Column *Horizon* shows the length of the time horizon (in secs.).

Table 1 Instance data

Name	$\|D\|$	$\|L\|$	$\|F\|$	Min Slot	Max Slot	Horizon
Arlanda1	21	19	40	0.5	4	1,797
Arlanda2	21	19	40	0.5	2	1,950
Arlanda3	17	16	33	0.5	2	1,784
Hamburg1	40	18	58	0.5	42	4,980
Hamburg2	50	22	72	0.5	6	3,180
Hamburg3	39	18	57	0.5	42	3,180

As pointed out in Section 1, size is a critical issue for the practical usability of TI formulations. Table 2 compares the size (before presolve) of *Basic*, *Nogueira* and $(S,t)-Clique$ formulations in terms of the number of rows and nonzeroes. It clearly shows that the (S,t)-clique formulation ensures a drastic reduction both in the number of rows and in the number of nonzeroes.

Table 3 compares for each instance lower bounds provided by the (S,t)-clique formulation with those found with the *Basic* and the *Nogueira* formulations, respectively. Columns Δ_1 and Δ_2 report the percentage of the lower bound increase from *Basic* to *Nogueira* and from *Nogueira* to (S,t)-clique formulation, respectively.

We observe that for all the benchmark instances, the (S,t)-clique formulation provides better bounds than the *Basic* and the *Nogueira* formulations with percentage improvements tending to increase as the size of the time slot decreases.

Table 4 compares the three formulations according to the time spent by the MIP solver to find the best upper bound solution, without using any logical presolve. The results clearly show the superiority of the (S,t)-clique formulation, both in terms of computation times and of the number of instances solved to optimality within the time-limit of 600 seconds.

Table 5 reports on the results of the overall algorithm depicted in Section 4. Columns *ncols*, *nrows* and *nonzeroes* show the number of columns, rows and nozeroes of the (S,t)-clique formulation after presolving, respectively. Columns LP, UB and $Time$ contain the value of the LP-relaxation, the best upper bound found at the end of the algorithm, and the total computation time, respectively. We observe that the lower bounds provided by the (S,t)-clique formulation are very close to the optimum and LP solutions are a good driver for the MIP heuristics embedded into MIP solvers. It follows that a key success factor is the ability to solve the TI formulation efficiently. The computational time of the proposed approach is suitably small and matches the requirements imposed by the real-time needs. It is remarkable that we could even solve to optimality instances with the time slot of 1 second, which is in many practical cases smaller than necessary.

Table 6 aims at putting in evidence the impact of the several components of the algorithm for a particular instance, namely *Arlanda*2 with the time slot set to 1 second. Each line in the table shows the cumulative effect of adding a new component to the previous ones. Line *Naked*

Instance	Slot	$\|I\|$	ncols	nrows			nonzeros		
				Basic	*Nogueira*	$(S,t)-$ *clique*	*Basic*	*Nogueira*	$(S,t)-$ *clique*
Arlanda1	0.5	181,574,132	46,509	1,791,713	199,705	20,801	268,779,333	57,096,854	20,899,113
Arlanda1	1	45,383,564	23,285	894,839	99,946	10,425	67,302,910	14,320,418	5,250,740
Arlanda1	2	11,316,526	11,666	446,066	49,948	5,228	16,847,914	3,588,457	1,319,108
Arlanda1	3	5,042,048	7,791	296,562	33,343	3,497	7,537,018	1,614,261	591,894
Arlanda1	4	2,828,078	5,849	221,875	24,984	2,155	4,283,366	910,904	277,355
Arlanda2	0.5	147,522,574	41,723	1,660,800	128,293	24,812	240,888,589	38,332,381	21,470,575
Arlanda2	1	36,870,728	20,892	829,345	64,237	12,435	60,305,398	9,614,706	5,392,040
Arlanda2	2	9,199,020	10,468	412,290	32,095	6,225	15,096,076	2,410,250	1,352,576
Arlanda3	0.5	107,778,742	33,538	1,095,710	77,464	22,336	161,268,256	23,805,006	17,000,682
Arlanda3	1	26,938,448	16,794	547,160	38,798	11,197	40,371,257	5,973,258	4,270,948
Arlanda3	2	6,722,602	8,414	271,870	19,389	5,601	10,105,801	1,498,533	1,070,665
Hamburg1	1	1,156,734,169	177,218	8,101,513	2,211,176	98,138	837,674,273	383,613,181	41,153,777
Hamburg1	2	289,220,184	88,640	4,047,919	1105,515	49,083	209,488,696	95,955,004	10,328,308
Hamburg1	3	168,677,964	59,120	2,697,312	737,199	32,755	93,835,484	43,114,239	14,990,548
Hamburg1	4	94,187,596	44,353	2,021,701	105,565	24,406	52,848,486	6,400,894	8,161,328
Hamburg1	5	60,087,542	35,497	1,616,416	84,446	19,536	33,808,796	4,088,056	5,224,746
Hamburg1	6	42,484,614	29,595	1,346,538	368,899	16,420	24,002,557	11,130,042	4,118,963
Hamburg1	7	31,161,242	25,352	1,153,382	60,318	13,968	18,081,786	2,180,438	2,799,494
Hamburg1	8	23,267,216	22,197	1,007,845	52,738	12,207	13,222,262	1,610,351	2,047,943
Hamburg1	9	18,458,386	19,735	895,214	46,902	10,869	10,592,170	1,299,714	1,540,514
Hamburg1	10	14,821,140	17,784	805,747	42,239	9,797	8,466,771	1,031,572	1,316,656
Hamburg2	0.5	47,0791,498	92,762	6,754,481	520,123	66,088	1,101,699,822	175,654,238	88,604,827
Hamburg2	1	117,616,984	46,442	3,373,562	260,045	33,090	275,671,594	43,960,047	22,197,749
Hamburg2	2	29,323,510	23,262	1,681,469	130,008	16,562	68,975,911	11,009,717	5,559,176
Hamburg2	3	13,087,568	15,542	1,118,659	86,810	11,067	30,905,529	4,950,610	2,525,971
Hamburg2	4	7,303,906	11,679	836,629	17,289	8,056	17,420,789	1,007,734	1,355,677
Hamburg2	5	4,663,830	9,365	667,636	13,833	6,450	11,154,829	645,712	868,852
Hamburg2	6	3,310,804	7,815	555,420	43,610	5,563	7,906,770	1,280,973	672,180
Hamburg3	0.5	285,897,658	71,736	4,164,976	282,321	25,765	661,173,843	94,790,271	24,358,034
Hamburg3	1	71,437,438	35,916	2,080,172	141,124	12,921	165,442,795	23,724,923	6,118,140
Hamburg3	2	17,815,834	17,989	1,036,669	70,557	6,478	41,394,696	5,943,650	1,536,690
Hamburg3	3	7,950,748	12,019	689,645	47,124	4,342	18,555,362	2,675,057	698,969
Hamburg3	4	4,437,818	9,030	515,637	7,884	3,087	10,456,475	487,932	367,065
Hamburg3	5	2,830,808	7,235	411,189	6,308	2,479	6,691,399	313,067	236,257
Hamburg3	6	2,009,764	6,043	342,327	23,677	2,202	4,752,003	693,500	186,274
Hamburg3	7	1,466,580	5,168	292,413	4,529	1,791	3,570,801	168,214	128,147
Hamburg3	8	1,094,234	4,538	254,295	3,953	1,567	2,616,304	124,196	94,001
Hamburg3	9	867,286	4,038	225,256	3,519	1,400	2,093,635	100,325	74,589
Hamburg3	10	699,782	3,652	202,640	3,175	1,267	1,679,357	80,376	61,196

Table 2 **LP sizes for** *Basic*, *Nogueira* **and** $(S,t)-clique$ **formulation on Arlanda and Hamburg Instances.**

(S,t)-*clique* demonstrates the size (number of columns and number of rows, respectively) and the computation time required by the overall algorithm for the "naked" (S,t)-clique formulation without performing any operation. Line *TW-tightening* contains the total time required by the algorithm when also including the presolve procedures of Section 4.1. Line *Non Dominated Cliques* shows the total time of the algorithm when including also the clique detection procedure of Section 3 to remove dominated cliques. Finally, row *Lifting* shows the effect of adding the sequential lifting procedure to enlarge the cliques introduced in Section 3.

Instance	Slot	*Basic*	*Nogueira*	$\%\Delta_1$	$(S,t)-$ *clique*	$\%\Delta_2$
Arlanda1	0.5	1,535.7	1,969.6	28.3	2,412.5	22.5
Arlanda1	1	1,537.8	1,974.9	28.4	2,429.1	23.0
Arlanda1	2	1,538.5	1,977.8	28.5	2,433.2	23.0
Arlanda1	3	1,555.2	1,993.6	28.2	2,469.0	23.8
Arlanda1	4	1,594.5	2,075.2	30.1	2,645.1	27.5
Arlanda2	0.5	1,148.5	1,495.4	30.2	1,696.1	13.4
Arlanda2	1	1,152.5	1,502.5	30.4	1,905.2	26.8
Arlanda2	2	1,152.0	1,500.8	30.3	1,903.6	26.8
Arlanda3	0.5	758.1	979.6	29.2	1,214.8	24.0
Arlanda3	1	758.6	982.8	29.6	1,219.5	24.1
Arlanda3	2	757.7	981.2	29.5	1,216.0	23.9
Hamburg1	1				3,196.0	
Hamburg1	2	1,612.0	2,962.8	83.8	3,195.0	7.8
Hamburg1	3	1,627.5	3,018.1	85.4	3,262.5	8.1
Hamburg1	4	1,642.0	3,094.5	88.5	3,262.0	5.4
Hamburg1	5	1,647.5	4,014.5	143.7	4,083.3	1.7
Hamburg1	6	1,737.0	3,183.2	83.3	4,335.0	36.2
Hamburg1	7	1,837.5	5,791.3	215.2	5,852.0	1.0
Hamburg1	8	1,742.0	4,100.6	135.4	4,192.0	2.2
Hamburg1	9	1,737.0	4,089.0	135.4	4,198.5	2.7
Hamburg1	10	1,742.5	4,138.1	137.5	4,250.0	2.7
Hamburg2	0.5				43,131.4	
Hamburg2	1	2,142.3	41,538.1	1,839.0	49,766.5	19.8
Hamburg2	2	2,142.5	41,872.7	1,854.4	49,766.0	18.9
Hamburg2	3	2,177.3	43,667.9	1,905.6	54,923.3	25.8
Hamburg2	4	2,159.0	47,192.8	2,085.9	54,862.7	16.3
Hamburg2	5	2,173.8	48,827.7	2,146.2	55,300.0	13.3
Hamburg2	6	2,262.3	49,419.5	2,084.5	67,473.3	36.5
Hamburg3	0.5	1,736.1	3,363.7	93.7	3,672.8	9.2
Hamburg3	1	1,742.0	3,378.2	93.9	3,958.0	17.2
Hamburg3	2	1,749.0	3,389.8	93.8	3,976.0	17.3
Hamburg3	3	1,768.5	3,475.0	96.5	4,302.0	23.8
Hamburg3	4	1,794.0	3,992.0	122.5	4,296.0	7.6
Hamburg3	5	1,837.5	4,190.0	128.0	4,320.0	3.1
Hamburg3	6	1,866.0	3,953.5	111.9	4,656.0	17.8
Hamburg3	7	1,981.0	5,040.0	154.4	5,310.7	5.4
Hamburg3	8	1,900.0	4,408.0	132.0	4,448.0	0.9
Hamburg3	9	1,872.0	4,374.0	133.7	4,509.0	3.1
Hamburg3	10	1,945.0	4,540.0	133.4	4,670.0	2.9

Table 3 Lower bounds for *Basic*, *Nogueira* **and** $(S,t)-clique$ **formulation on Arlanda and Hamburg Instances.**

5.2. Results with the sets A and C

As pointed out before, the instances considered in Furini et al. (2015) do not present time windows and the objective function consists in penalizing the delay with respect to the expected departure/arrival time. For such instances the master problem of the column generation algorithm from Section 4 was initialized by introducing an artificial time window imposing a maximum delay of 300 secs. and taking all the columns falling in this interval.

Table 7 reports on computational experiments with the A and C instances. Columns 3-5 contain the detailed results — best upper bound, best lower bound and computation time, respectively — for the algorithm of Furini et al. (2015). Columns 6-8 present the results obtained by our algorithm.

		Basic		*Nogueira*		$(S,t)-clique$	
Instance	Slot	BUB	Time	BUB	Time	BUB	Time
Arlanda1	0.5	36,679.0	600.0	16,385.5	600.0	**2,416.0**	490.7
Arlanda1	1	4,460.0	600.0	13,184.0	600.0	**2,430.0**	42.7
Arlanda1	2	**2,434.0**	121.9	**2,434.0**	103.8	**2,434.0**	8.9
Arlanda1	3	**2,469.0**	30.4	**2,469.0**	27.6	**2,469.0**	2.3
Arlanda1	4	**2,660.0**	8.1	**2,660.0**	7.2	**2,660.0**	1.9
Arlanda2	0.5	3,573.5	600.0	1,847.0	600.0	**1,697.5**	118.2
Arlanda2	1	2,032.0	600.0	2,480.0	600.0	**1,918.0**	43.8
Arlanda2	2	**1,916.0**	51.9	**1,916.0**	43.7	**1,916.0**	6.5
Arlanda3	0.5	1,256.5	600.0	1,239.0	600.0	**1,218.5**	83.1
Arlanda3	1	**1,223.0**	134.2	**1,223.0**	56.8	**1,223.0**	15.4
Arlanda3	2	**1,220.0**	19.7	**1,220.0**	10.8	**1,220.0**	3.2
Humburg1	1		600.0		600.0	**3,201.0**	73.7
Humburg1	2	390,436.0	600.0	378,836.0	600.0	**3,200.0**	15.3
Humburg1	3	7,731.0	600.0	3,270.0	497.6	**3,270.0**	7.5
Humburg1	4	**3,272.0**	322.8	**3,272.0**	37.8	**3,272.0**	3.4
Humburg1	5	**4,115.0**	139.1	**4,115.0**	24.9	**4,115.0**	2.7
Humburg1	6	**4,398.0**	102.1	**4,398.0**	89.6	**4,398.0**	3.0
Humburg1	7	**5,852.0**	71.6	**5,852.0**	13.5	**5,852.0**	1.3
Humburg1	8	**4,216.0**	50.7	**4,216.0**	11.1	**4,216.0**	1.3
Humburg1	9	**4,221.0**	36.1	**4,221.0**	9.0	**4,221.0**	1.2
Humburg1	10	**4,290.0**	27.1	**4,290.0**	9.3	**4,290.0**	1.0
Humburg2	0.5		600.0		600.0	**49,401.5**	600.0
Humburg2	1	131,411.0	600.0	147,409.0	600.0	**58,596.0**	197.5
Humburg2	2	**58,588.0**	282.6	**58,588.0**	140.6	**58,588.0**	59.1
Humburg2	3	**59,093.0**	98.3	**59,093.0**	65.6	**59,093.0**	28.0
Humburg2	4	**59,120.0**	47.0	**59,120.0**	4.1	**59,120.0**	10.3
Humburg2	5	**59,335.0**	24.3	**59,335.0**	2.2	**59,335.0**	4.5
Humburg2	6	**77,950.0**	17.4	**77,950.0**	8.0	**77,950.0**	6.3
Humburg3	0.5	5,689.0	600.0	4,147.0	600.0	**3,692.0**	36.8
Humburg3	1	4,433.0	600.0	3,976.0	600.0	**3,974.0**	26.1
Humburg3	2	**3,976.0**	43.9	**3,976.0**	25.6	**3,976.0**	5.5
Humburg3	3	**4,302.0**	14.2	**4,302.0**	7.4	**4,302.0**	2.8
Humburg3	4	**4,296.0**	6.2	**4,296.0**	1.4	**4,296.0**	1.5
Humburg3	5	**4,320.0**	3.5	**4,320.0**	0.8	**4,320.0**	1.1
Humburg3	6	**4,656.0**	2.4	**4,656.0**	1.0	**4,656.0**	1.7
Humburg3	7	**5,334.0**	1.7	**5,334.0**	0.4	**5,334.0**	0.9
Humburg3	8	**4,448.0**	1.2	**4,448.0**	0.2	**4,448.0**	0.7
Humburg3	9	**4,509.0**	1.0	**4,509.0**	0.2	**4,509.0**	0.7
Humburg3	10	**4,670.0**	0.8	**4,670.0**	0.1	**4,670.0**	0.6

Table 4 Computation time to solve for *Basic*, *Nogueira* and $(S,t)-clique$ formulation on Arlanda and Hamburg Instances.

The computational tests of Furini et al. ran on an Intel(R) Core(TM)2 Duo CPU clocked at 3.2GHz with 2Gb RAM under the Linux operating system. According to DIMACS benchmarks, their machine is about 5 times slower than ours, so we scaled the computation time reported in Furini et al. (2015) accordingly.

Table 7 demonstrates that all the A and C instances were solved to optimality, with computation times significantly faster than in Furini et al. (2015), and meeting service requirements. We also observe that imposing the artificial time windows with a maximum delay of 300 seconds made

Instance	Slot	$\|I\|$	ncols	nrows	nonzeros	LP	UB	Time
Arlanda1	0.5	145,217,002	39,081	19,019	18,541,786	2,412.5	2,416.0	388.1
Arlanda1	1	36,297,310	19,571	9,532	4,657,210	2,429.1	2,430.0	24.8
Arlanda1	2	9,058,252	9,813	4,778	1,170,258	2,433.2	2,434.0	5.6
Arlanda1	3	4,024,372	6,543	3,191	524,355	2,469.0	2,469.0	1.6
Arlanda1	4	2,228,968	4,856	1,994	249,429	2,645.1	2,660.0	1.3
Arlanda2	0.5	77,955,758	26,705	21,147	15,546,143	1,696.8	1,697.5	62.1
Arlanda2	1	19,480,604	13,380	10,594	3,902,012	1,918.0	1,918.0	10.1
Arlanda2	2	4,859,670	6,712	5,294	978,871	1,916.0	1,916.0	2.2
Arlanda3	0.5	56,802,208	23,319	18,818	12,745,258	1,212.5	1,218.5	44.0
Arlanda3	1	14,204,972	11,686	9,423	3,199,416	1,217.7	1,223.0	8.0
Arlanda3	2	3,555,596	5,871	4,706	803,058	1,214.7	1,220.0	1.7
Hamburg1	1	920,732,865	154,230	94,461	33,711,575	3,196.0	3,201.0	68.8
Hamburg1	2	305,670,900	77,114	47,091	8,458,212	3,195.0	3,200.0	13.2
Hamburg1	3	136,054,212	51,397	31,383	3,807,252	3,262.5	3,270.0	6.1
Hamburg1	4	75,652,094	38,429	23,142	2,112,399	3,262.0	3,272.0	2.9
Hamburg1	5	48,131,612	30,697	18,420	1,353,532	4,083.3	4,115.0	2.3
Hamburg1	6	34,011,280	25,583	15,575	981,995	4,335.0	4,398.0	2.5
Hamburg1	7	24,560,476	21,650	12,957	716,264	5,852.0	5,852.0	1.3
Hamburg1	8	18,541,832	19,122	11,347	532,082	4,192.0	4,216.0	1.2
Hamburg1	9	14,781,860	17,075	10,168	433,429	4,198.5	4,221.0	1.1
Hamburg1	10	11,759,082	15,266	9,007	342,410	4,250.0	4,290.0	1.0
Hamburg2	0.5	231,420,902	61,914	61,485	27,704,251	47,438.7	49,401.5	162.5
Hamburg2	1	57,796,050	31,006	30,769	14,563,167	58,596.0	58,596.0	59.4
Hamburg2	2	14,377,008	15,522	15,384	3,658,471	58,588.0	58,588.0	14.1
Hamburg2	3	6,383,480	10,343	10,268	1650,615	59,032.0	59,093.0	11.1
Hamburg2	4	3,515,204	7,723	7,495	896,821	59,061.3	59,120.0	5.6
Hamburg2	5	2,226,130	6,149	5,835	573,168	59,335.0	59,335.0	2.7
Hamburg2	6	1,558,584	5,088	5,005	427,930	77,950.0	77,950.0	4.2
Hamburg3	0.5	136,263,134	48,860	23,410	17,787,912	3,672.8	3,692.0	83.8
Hamburg3	1	34,045,348	24,468	11,714	4,458,312	3,958.0	3,974.0	14.8
Hamburg3	2	84,73,344	12,243	5,834	1,114,241	3,976.0	3,976.0	3.7
Hamburg3	3	3,759,142	8,159	3,888	500,521	4,302.0	4,302.0	2.3
Hamburg3	4	2,076,462	6,088	2,723	261,343	4,296.0	4,296.0	1.1
Hamburg3	5	1,308,452	4,840	2,168	166,502	4,320.0	4,320.0	0.9
Hamburg3	6	923,292	4,039	1,932	129,015	4,656.0	4,656.0	1.5
Hamburg3	7	638,070	3,344	1,530	85,390	5,310.7	5,334.0	0.7
Hamburg3	8	500,718	3,018	1,342	64,885	4,448.0	4,448.0	0.6
Hamburg3	9	402,780	2,720	1,219	52,772	4,509.0	4,509.0	0.5
Hamburg3	10	308,534	2,376	1,063	41,351	4,670.0	4,670.0	0.5

Table 5 Computational results $(S,t)-clique$ formulation with TW-tightening

	ncols	nrows	non zeros	LP	Time
Naked (S,t)-clique	20,892	19,316	5,720,110	2,429.1	54.8
TW-tightening	13,380	16,073	3,637,348	2,429.1	43.5
Non Dominated Cliques	13,380	10,594	2,670,983	2,429.1	28.0
Lifting	13,380	10,594	3,902,012	2,429.1	10.1

Table 6 Computational results on Arlanda2 with the slot size set to 1 second

the initial master problem already optimal for the LP-relaxation so there was no need to add new external columns.

		Furini et al.			$(S,t)-clique$ (max delay = 300)	
name	$\|F\|$	UB	LB	Time	OPT	Time
FPT01	60	265	265	24.0	265	3.2
FPT02	60	293	293	24.0	293	3.2
FPT03	60	255	255	24.0	255	2.8
FPT04	60	268	268	24.0	268	3.0
FPT05	60	249	249	24.0	249	3.2
FPT06	60	167	167	24.0	167	3.0
FPT07	60	198	198	24.0	198	3.1
FPT08	60	167	167	24.0	167	3.4
FPT09	60	183	183	24.0	183	3.1
FPT10	60	211	211	24.0	211	3.1
FPT11	60	229	229	24.0	229	3.5
FPT12	60	207	207	8.0	207	3.6
FPT_0_69	70	604	261	120.0	604	4.3
FPT_50_119	120	2,272	312	120.0	1,994	4.4
FPT_100_169	170	796	338	120.0	796	4.5
FPT_0_89	90	1,332	331	120.0	1,316	10.0
FPT_40_129	130	2,885	408	120.0	2,368	8.9
FPT_80_169	170	1,800	390	120.0	1,508	8.9
FPT_0_109	109	2,131	408	120.0	2,115	20.3
FPT_30_139	139	3,935	458	120.0	3,055	18.5
FPT_60_169	170	5,019	451	120.0	3,577	20.0
FPT_0_129	130	3,167	509	120.0	2,909	29.4
FPT_20_149	150	4,846	536	120.0	3,649	27.6
FPT_40_169	170	5,159	511	120.0	3,691	26.8
FPT_0_149	150	4,467	593	120.0	3,786	42.1
FPT_10_159	160	5,808	576	120.0	4,142	38.8
FPT_20_169	170	6,434	577	120.0	4,171	40.4

Table 7 Computational results with Furini et al. (2015) instances.

6. Final remarks

We address the integrated arrival/departure management problem on a single runway by a suitably compact time-indexed formulation defined through a special family of clique inequalities that can be handled by MIP solvers even on standard laptops.

The computational experience on a number of real and realistic instances confirms that computational times meet the practical real-time requirements of tower controllers. Reversing the opinion expressed in a recent survey (and generally accepted), we have shown that MIP approaches can effectively be exploited to solve RSP in practical environments. Moreover, and in contrast with alternative heuristic approaches to the problem, the methodology provides tight estimations on the solution quality and the optimality could be proven for all instances in the available testbeds.

Finally, we want to remark that several objective functions can be used to penalize delays and that the TI formulation can easily accommodate other objective functions, like those based on convex piecewise linear costs.

Acknowledgments

We wish to thank Fabio Furini for his kind collaboration in providing instances and detailed results.

References

Abela, J., D. Abramson, M. Krishnamoorthy, A. De Silva, G. Mills. 1993. Computing optimal schedules for landing aircraft. *Proceedings of the 12th National Conference of the Australian Society for Operations Research, Adelaide*.

Ascheuer, N., M. Fischetti, M. Grötschel. 2001. Solving the asymmetric travelling salesman problem with time windows by branch-and-cut. *Mathematical Programming* **90**(3) 475–506.

Atkin, J.A.D., E. Burke, J. Greenwood, D. Reeson. 2007. Hybrid metaheuristics to aid runway scheduling at london heathrow airport. *Transportation Science* **41**(1) 90–106.

Balakrishnan, H., B.G. Chandran. 2010. Algorithms for scheduling runway operations under constrained position shifting. *Operations Research* **58**(6) 1650–1665.

Beasley, J.E., M. Krishnamoorthy, Y.M. Sharaiha, D. Abramson. 2000. Scheduling aircraft landings the static case. *Transportation science* **34**(2) 180–197.

Beasley, J.E., J. Sonander, P. Havelock. 2001. Scheduling aircraft landings at london heathrow using a population heuristic. *Journal of the operational Research Society* 483–493.

Bennell, J.A., M. Mesgarpour, C.N. Potts. 2011. Airport runway scheduling. *4OR* **9**(2) 115–138.

Briskorn, D., R. Stolletz. 2014. Aircraft landing problems with aircraft classes. *Journal of Scheduling* **17**(1) 31–45.

De Maere, G., J.A.D. Atkin. 2015. Pruning rules for optimal runway sequencing with airline preferences. *Lecture Notes in Management Science* **7** 76–82.

Furini, F. 2015. Personal communication.

Furini, F., M.P. Kidd, C.A. Persiani, P. Toth. 2014. State space reduced dynamic programming for the aircraft sequencing problem with constrained position shifting. P. Fouilhoux, L. Gouveia, A. R. Mahjoub, V.T. Paschos, eds., *Combinatorial Optimization*, *Lecture Notes in Computer Science*, vol. 8596. Springer International Publishing, 267–279.

Furini, F., M.P. Kidd, C.A. Persiani, P. Toth. 2015. Improved rolling horizon approaches to the aircraft sequencing problem. *Journal of Scheduling* **18**(5) 435–447.

Furini, F., C.A. Persiani, P. Toth. 2012. Aircraft sequencing problems via a rolling horizon algorithm. *Combinatorial Optimization*, *Lecture Notes in Computer Science*, vol. 7422. Springer International Publishing, 273–284.

Graupl, T., B. Jandl, C.H. Rokitansky. 2012. Simple and efficient integration of aeronautical support tools for human-in-the-loop evaluations. *Integrated Communications, Navigation and Surveillance Conference (ICNS), 2012*. IEEE, F4–1.

Harrod, S. 2011. Modeling network transition constraints with hypergraphs. *Transportation Science* **45**(1) 81–97.

Heidt, A., H. Helmke, F. Liers, A. Martin. 2013. Robust runway scheduling using a time-indexed model. *Schaefer, D. (ed) Proceedings of the SESAR Innovation Days (2014) EUROCONTROL*.

Kjenstad, D., C. Mannino, P. Schittekat, T. Nordlander, M. Smedsrud. 2013a. Optimizing aman-sman-dman at hamburg and arlanda airport. *Schaefer, D. (ed) Proceedings of the SESAR Innovation Days (2013) EUROCONTROL*.

Kjenstad, D., C. Mannino, P. Schittekat, M. Smedsrud. 2013b. Integrated surface and departure management at airports by optimization. *Modeling, Simulation and Applied Optimization (ICMSAO), 2013 5th International Conference on*. IEEE.

Kopf, R., G. Ruhe. 1987. A computational study of the weighted independent set problem for general graphs. *Foundations of Control Engineering* **12**(4) 167–180.

Lieder, A., D. Briskorn, R. Stolletz. 2015. A dynamic programming approach for the aircraft landing problem with aircraft classes. *European Journal of Operational Research* **243**(1) 61 – 69.

Masin, M., T. Raviv. 2014. Linear programming-based algorithms for the minimum makespan high multiplicity jobshop problem. *Journal of Scheduling* **17**(4) 321–338.

Nogueira, T. H., C.R.V. de Carvalho, M.G. Ravetti. 2014. Analysis of mixed integer programming formulations for single machine scheduling problems with sequence dependent setup times and release dates. *Optimization Online* .

Queyranne, M., A.S. Schulz. 1994. *Polyhedral approaches to machine scheduling*. Fachbereich Mathematik: Preprint-Reihe Mathematik, TU, Fachbereich 3.

Samà, M., A. DAriano, D. Pacciarelli. 2013. Rolling horizon approach for aircraft scheduling in the terminal control area of busy airports. *Transportation Research Part E: Logistics and Transportation Review* **60** 140–155.

Savelsbergh, M.W.P., R.N. Uma, J. Wein. 2005. An experimental study of lp-based approximation algorithms for scheduling problems. *INFORMS Journal on Computing* **17**(1) 123–136.

SESAR, Joint undertaking. 2007. High performing aviation for europe, http://www.sesarju.eu/. URL `http://www.sesarju.eu/`.

Sousa, J.P., L.A. Wolsey. 1992. A time indexed formulation of non-preemptive single machine scheduling problems. *Mathematical Programming* **54**(1-3) 353–367.

Uma, R.N., J. Wein. 1998. On the relationship between combinatorial and lp-based approaches to np-hard scheduling problems. *Integer Programming and Combinatorial Optimization*. Springer, 394–408.

Van den Akker, J.M., C.A.J. Hurkens, M.W.P. Savelsbergh. 2000. Time-indexed formulations for machine scheduling problems: Column generation. *INFORMS Journal on Computing* **12**(2) 111–124.

Waterer, H., E.L. Johnson, P. Nobili, M.W.P. Savelsbergh. 2002. The relation of time indexed formulations of single machine scheduling problems to the node packing problem. *Mathematical Programming* **93**(3) 477–494.

Wolsey, L.A. 1998. *Integer programming*, vol. 42. Wiley New York.

Wolsey, L.A., G.L. Nemhauser. 2014. *Integer and combinatorial optimization*. John Wiley & Sons.