Università degli Studi di Roma Tre
Dipartimento di Informatica e Automazione
Via della Vasca Navale, 79 – 00146 Roma, Italy

Real-time scheduling of aircraft arrivals and departures in a terminal maneuvering area

Andrea D'Ariano[1], Dario Pacciarelli[1], Marco Pistelli[1] and Marco Pranzo[2]

RT-DIA-198-2012 **August 2012**

(1) Università degli Studi Roma Tre,
Via della Vasca Navale, 79
00146 Roma, Italy.

(2) Università degli Studi di Siena,
Via Roma 56
53100 Siena, Italy.

This work is partially supported by the Italian Ministry of Research, Grant number RBIP06BZW8, project FIRB "Advanced tracking system in intermodal freight transportation". Preliminary results on a subset of the ASP instances have been presented at the 13th International IEEE Conference on Intelligent Transportation Systems in September 2010 [13].

ABSTRACT

The Aircraft Scheduling Problem (ASP) is the real-time problem of scheduling take-off and landing operations at a congested airport in a given time horizon, taking into account the runways and the air segments in the Terminal Maneuvering Area (TMA). The ASP can be viewed as a job shop scheduling problem with additional real-world constraints. Compared with the current literature based on job shop scheduling applied to solve the ASP, we enrich the existing models by including new formulations of relevant practical constraints. We introduce and analyze three alternative ASP formulations, in which the objective function is the minimization of delay propagation with respect to the off-line timetable. Scheduling rules, heuristic and exact methods are implemented and tested on instances from the Roma Fiumicino (FCO) airport, in Italy. Computational experiments show that practical-size instances are solved to near-optimality by our branch and bound algorithm in a few seconds of computation.

Keywords: Air Traffic Control; Aircraft Delay Minimization; Job Shop Scheduling; Alternative Graph; Branch and Bound.

1 Introduction

The ever increasing air traffic demand of passengers and cargos all over the world is currently limited by the capacity of the airports, which are expected to become a serious bottleneck in Air Traffic Control (ATC) operations in a near future [1]. Aviation authorities are thus seeking methods to better use the existing airport infrastructure and to better manage aircraft movements in the proximity of airports, improving aircraft punctuality while maintaining the required level of safety. Decision Support Systems (DSSs) based on optimization algorithms may help to exploit at most the capacity available in the Terminal Maneuvering Area (TMA) during operations. In this context, the optimization of take-off/landing operations is a key factor to improve the performance of the entire ATC system. However, ATC operations are still mainly performed by human controllers with only a limited aid from automated systems. In most cases, computer support consists of a graphical representation of the current aircraft position and speed.

This paper deals with the development of optimization models and algorithms for real-time aircraft scheduling at a busy airport. The algorithms have been tested on practical size instances from the FCO airport, one of the world's busiest airports by passenger traffic and the main airport in Italy. Further computational results on the second busiest airport in Italy (Milano Malpensa, MXP) can be found in Pistelli et al. [23].

From a logical point of view, ATC decisions in a TMA can be broadly divided into: *Routing decisions*, where a route for each aircraft has to be chosen from its current position to its destination. *Scheduling decisions*, where feasible aircraft sequencing and timing have to be determined in each airway, such that safety regulations are satisfied and a given performance index is optimized. Even if, in practice, routing and scheduling decisions in a TMA are taken simultaneously, the main objective of routing decisions is typically to balance the use of critical resources (e.g., alternative runways, air corridors) while that of scheduling is the delay minimization. This paper focuses on real-time aircraft scheduling decisions with fixed routes, and we refer to this problem as the *Aircraft Scheduling Problem* (ASP). We assume that the routing problem is solved off-line in a preliminary step.

We now briefly recall the general procedure for take-off/landing operations in the proximity of airports. For each TMA, landing aircraft move along predefined routes from an entry fix to a runway following a standard descent profile. During all the approach phases, a minimum separation distance between every pair of consecutive aircraft must be guaranteed for each air segment and runway of the TMA. This standard separation depends on the type and relative positions of the two aircraft (at the same or different altitude). A minimum separation between aircraft is mandatory and prescribed by international aviation regulations. By considering the different aircraft speeds, the safety distance can be translated in a Separation Time Interval (STI). Similarly, departing aircraft leave the runway flying towards the assigned exit fix along an ascent profile, respecting separation standards. The runway can be occupied by only one aircraft at a time, and a safety separation time should be ensured between any pair of aircraft. Once a landing aircraft enters the TMA it should proceed to the runway. However, it is possible to let aircraft wait in flight in holding circles until they can be guided into the landing sequence. Once in the holding circle, an aircraft can leave it only after the traversing of (or a multiple of) half length of the circle.

A landing aircraft is considered to be late when landing after its scheduled arrival time. A departing aircraft is supposed to take-off within its assigned time window and is late

whenever it is not able to accomplish the departing procedure within its assigned time window. A conflict occurs whenever two or more aircraft do not respect the minimum required distance at the same air segment or runway. Any potential conflict must be detected and solved in real-time by human air traffic controllers by rescheduling aircraft movements [6].

In practice, the air traffic controllers of the TMA take their decisions in coordination with the en-route controllers of neighbouring areas. It is therefore worth investigation the extent at which decision support systems should support the collaboration between air traffic controllers of different areas or should support only the decisions taken in the TMA, thus considering fixed the aircraft entrance time. In this paper we investigate two versions of the ASP. In the *non-collaborative* problem, the entrance time of each aircraft in the TMA is given in input and it is therefore a constraint to be satisfied in a feasible solution. In the *collaborative* problem, the entrance time of each aircraft in the TMA is only constrained to be greater than or equal to a minimum value, while the actual value has to be determined in order to optimize the objective function.

Summarizing the above discussion, the ASP can be stated as follows: given a set of aircraft willing to land/take-off, and given for each aircraft its approach/leaving path, its current speed, its runway occupancy time, and a time window to accomplish the landing (departing) procedures, our approach assigns to each aircraft the start time from the fix (runway) and all relevant points in such a way that all aircraft conflicts are solved, all the constraints on safety separation distances are satisfied, and a given system performance index is optimized.

The original contribution of this paper is three-fold:

- With respect to the existing models for the management of aircraft traffic in a TMA, we increase the level of accuracy in modeling the ASP constraints, both in the collaborative and non-collaborative cases. Specifically, we combine the following modeling features: (i) holding circle constraints; (ii) STI constraints for air segments; (iii) blocking constraints for runways; (iv) time windows constraints for the aircraft travel time in air segments.

- We adapt existing exact and heuristic algorithms in order to deal with the constraints of the specific ASP formulation.

- Extensive computational experiments carried out for the FCO airport demonstrate that practical size instances are solved at optimality or near-optimality by these algorithms within a short computation time, compatible with real-time application.

The paper is organized as follows. Section 2 briefly reviews the literature on ASP, Section 3 introduces our three formulations of the ASP, Section 4 describes heuristic and exact solution methods, Section 5 presents the computational results on the FCO airport and, finally, Section 6 gives conclusions and future research directions.

2 Literature review

The ASP has been the subject of many papers. In view of the survey papers of Kuchar and Yang [21], Ball et al. [6] and Bennell et al. [9], this section only reviews the papers

most related to our work. The ASP can be broadly classified as static or dynamic, while existing models can be grouped as basic or detailed.

In the *static* ASP, landing/departing aircraft must be sequenced when all information is known in advance, whereas in the *dynamic* ASP aircraft enter the TMA one at the time, and a new sequence of take-off/landing aircraft has to be recomputed every time a new incoming aircraft is known. We observe that the difference between static and dynamic problems is sometimes artificial, specially if previously scheduled aircraft can be re-sequenced when a new aircraft enters the TMA. In such cases, a static algorithm can be applied every time the next relevant event occurs, provided that it is fast enough to compute a solution during operations. Similarly, a dynamic approach may not be usable on-line if the CPU requirement is too high. In fact, for real-time purposes aircraft require a continuous monitoring of their position and speed, and therefore the computation time of the algorithm is more important than static/dynamic classification.

Basic models include only the runways in the TMA, while *detailed* approaches also model the air segments. Most of the early papers on ASP falls in the former category and the choice is motivated by the fact that the runways are often the bottleneck of the TMA. With a basic model the ASP is typically formulated as a single or a parallel machine scheduling problem, while detailed formulations model the ASP as a job shop scheduling problem. In both cases there can be additional constraints to take into account specific characteristics of the ASP. For example, no-wait constraints are used to model the traversing of air segments [10, 20] and blocking constraints for the traversing of runways [2, 13]. A description of blocking and no-wait resources for general scheduling problems can be found in [17]. In the ASP, basic models are more tractable than detailed models and may lead to useful insights for the problem. At the same time, they are less realistic since bottleneck situations may happen also in air segments of the TMA and in any case a solution that is feasible for a basic model may not be feasible in practice. Basic models may have a limited impact on the practice of air traffic control and the recent trend of research is to incorporate more practical constraints in the model.

We next briefly review basic models for the ASP. Among the earliest papers on the dynamic ASP, Dear [14] proposes an effective solution approach by constraining the set of feasible positions in the sequence for the new aircraft to avoid excessive perturbations to the already scheduled aircraft. With this constraint, known as Constrained Position Shifting (CPS), no aircraft can be sequenced forward or backward more than a specified number of positions with respect to the First In First Out (FIFO) sequence. CPS is also used by Psaraftis [25], who develops a dynamic program for the static ASP, and by Venkatakrishnan et al. [28] that adapt the latter approach to the dynamic ASP among others.

Ernst et al. [15] tackle the static ASP by using a specialized simplex algorithm for the single runway case and extend it to the multiple runway case. Beasley et al. [7] present a mixed-integer zero-one formulation for the static ASP in the single and multiple runways cases. The problem is then solved using exact and heuristic algorithms. Hansen [18] proposes a genetic algorithm for the routing and sequencing of landing aircraft on a multiple runway case.

Beasley et al. [8] formulate a dynamic ASP with multiple runways as a displacement problem. If an aircraft is further delayed with respect to the previous solution then an additional penalty has to be paid. They also adapt to the dynamic case an exact and two heuristic approaches previously developed to solve the static version of the ASP. In [29],

an Ant Colony System combined with a rolling horizon approach is applied to solve the dynamic ASP in a single runway setting.

Artiouchine et al. [4] present a basic model that addresses the problem of scheduling aircraft on a single runway. The objective is to assign landing times to aircraft with a mono-pattern, where all aircraft are identical and where a single holding pattern is only considered.

Soomer and Franx [26] develop an approach for the static ASP with a single runway based on airlines' priorities and collaborative decision making. Each airline owns a set of aircraft and provides for each aircraft the cost function associated to its delay. The authors propose a scaling procedure of the costs of each airline to ensure equity. A mixed integer formulation and a local search procedure are developed to solve the problem.

Sölveling et al. [27] include the impact of environmental costs in the ASP for a multi runway airport, considering a detailed representation of several costs in the objective function, such as taxing, crossing and gates idling operations, as well as CO_2 emissions costs. A mixed integer formulation of the problem is proposed and the problem is then solved by a general MIP solver.

Eun et al. [16] develop a DSS for an airport with a single runway. They face a dynamic ASP in which the controllers are allowed to delay airborne aircraft using only a discrete set of delay values. The objective function is the minimization of aircraft delays. The solution approach is based on a branch-and-bound algorithm with linear programming and Lagrangian dual decomposition. The DSS uses a rolling horizon approach to solve the dynamic ASP and to reduce the size of the instance. The approach is validated by simulation and using real data of the Seoul GMP airport in South Korea.

We next review detailed ASP models. This stream of research strives to develop models adherent to the ATC practice. The TMA is formulated as a job shop scheduling problem in which each machine models a portion of the TMA, either an air segment or a runway, and each aircraft corresponds to a job. The traversing of an air segment or a runway by a specific aircraft is known as an operation, whose processing time is equal to the traversing time of the associated air segment/runway. Additional constraints take into account the practical aspects of the ASP, such as speed constraints for each flying aircraft and sequence-dependent set-up times to model separation time intervals between consecutive aircraft.

Bianco et al. [10] consider a detailed model in which the TMA is formulated as a no-wait job shop scheduling problem with release times and sequence-dependent setup times to model aircraft safety separations. They implement a local search algorithm which compares favourably versus a simple control rule sequencing the aircraft at the runways according to the FIFO rule. Results are shown for the MXP and FCO airports in Italy.

Adacher et al. [2] propose a detailed model for the static ASP problem based on the alternative graph (AG) model of Mascis and Pacciarelli [22], that is a generalization of the disjunctive graph formulation. Differently from previous approaches based on job shop scheduling, the alternative graph enables an accurate representation of the air traffic regulations to be taken into account when solving the ASP. The resulting model enriches the one of Bianco et al. [10] by including additional real-world constraints such as holding circles, time windows of [minimum, maximum] allowed aircraft speeds, multiple capacity of air segments and blocking constraints at runways. Simple heuristic algorithms are proposed to solve the problem.

Artiouchine et al. [5] introduce another abstract model where the routing and schedul-

ing problem for landing operations is formulated through the K King problem. According to the authors, this approach is a strong abstraction of landing operations. For example, STI constraints are not accurately modelled, as well as the range of aircraft dynamics.

Bennell et al. [9] review optimization approaches for scheduling aircraft in a TMA. According to the authors, there are only a few attempts to model the aircraft scheduling problem including all the TMA resources. Many theoretical studies promise solutions which may not be possible to implement in practice, as some critical operational constraints are ignored or relaxed in the model or the required computational resources are unreasonable. Moreover, landing and take-off problems are often solved separately and there may be a very limited forecast of traffic both in time and space. These facts limit the knowledge of the impact of scheduling decisions on the delay propagation to ingoing and outgoing aircraft. In the conclusions, the authors point out that scheduling algorithms can be of practical use only if they can provide near-optimal solutions within seconds of computation.

From the analysis of the ASP literature, we can conclude that the detailed models enable a better representation of the practical problem with respect to basic models. However, further research is needed to develop effective and efficient solution algorithms, as well as to find the right compromise between a number of factors, including the degree of detail in the formulation of relevant practical constraints, the effectiveness of solution algorithms, the time available to compute ASP solutions, the level of collaboration among en-route and TMA controllers, and the time horizon of the prediction. Moreover, even if the delay minimization is typically the main indicator of the solution quality, in practice traffic controllers pay attention to other indicators, such as the total time spent by the aircraft in the TMA. All these issues motivates this paper.

3 Problem Formulation

This section first introduces the alternative graph model and then shows how objective function, variables and constraints of ASP are formulated. Three models are presented for managing traffic in a TMA and an illustrative example follows for the FCO airport.

Alternative graphs have been effectively used to model complex real-world job-shop scheduling problems arising in public transportation and manufacturing [12, 22]. In the ASP, an *operation* is an elementary movement by a specific aircraft, such as the traversing of an air segment, holding circle or runway. As in the usual definition of the job shop scheduling problem, a job (aircraft) must undertake a prescribed sequence of operations on specific machines (TMA resources).

An alternative graph is a triple $\mathcal{G} = (N, F, A)$, where $N = \{0, 1, \ldots, n-1, n\}$ is the set of nodes, F is a set of directed arcs (*fixed*) and A is a set of pairs of directed arcs (*alternative*). In set N, node $i = 1, \ldots, n-1$ is associated to the start time t_i of operation $o_i = o_1, \ldots, o_{n-1}$, while nodes $0/n$ are associated to the start/end of the schedule (i.e., to the dummy operations o_0 and o_n). Arcs in the set F model the sequence of operations to be executed by a job (the route of a landing/departing aircraft in the TMA). Arcs in the set A model the sequencing decisions. If $((i,j),(h,k)) \in A$, arc (i,j) is the alternative of arc (h,k) and defines an order between two operations. Each arc (i,j), either fixed or alternative, has an associated weight $w_{i,j}$.

A *selection* S is a set of alternative arcs, at most one from each pair. Given a selection

S, $l^S(i,j)$ denotes the length of a longest path from node i to node j in the graph $G(S) = (N, F \cup S)$. A selection, in which exactly one arc is chosen from each pair in A, is a *feasible schedule* (i.e., a solution to the ASP) if $G(S)$ has no positive weight cycles. In fact, a positive cycle represents a circular precedence between operations, i.e., an operation which precedes itself. Given a feasible schedule S, the minimum start time of o_i is $t_i = l^S(0,i)$, for each $i \in N$. A feasible schedule S is *optimal* if $l^S(0,n)$ is minimum over all the solutions.

The problem of finding an optimal selection in $\mathcal{G} = (N, F, A)$ can be formulated as a particular *disjunctive program*, i.e., a linear program with logical conditions "and" ($\wedge$, conjunction) and "or" ($\vee$, disjunction):

$$\begin{array}{ll} \min t_n - t_0 & \\ s.t. & \\ t_q - t_p \geq w_{p,q} & (p,q) \in F \\ (t_j - t_i \geq w_{i,j}) \vee (t_k - t_h \geq w_{h,k}) & ((i,j),(h,k)) \in A \end{array}$$

Each variable t_i is the start time of o_i, operation o_0 precedes (o_n follows) all the other operations and t_0 (t_n) represents the start (the end) of traffic prediction. Given a fixed arc (p,q), timing is feasible if $t_q \geq t_p + w_{p,q}$. Given an alternative arc $((i,j),(h,k))$, if the arc (i,j) is selected than $t_j \geq t_i + w_{i,j}$ must be satisfied in a feasible schedule, alternatively if the arc (h,k) is selected than $t_k \geq t_h + w_{h,k}$ must be satisfied.

3.1 Modeling the ASP via alternative graphs

This subsection presents the alternative graph formulation of ASP decision variables, constraints and objective function. A TMA is composed of various types of resources: runways, air segments and holding circles. We next illustrate the formulation of each type of resource. Given an operation o_i, we denote with $o_{\sigma(i)}$ the operation which follows o_i on the same job.

The traversing time of an air segment by an aircraft can range in an interval $[\phi, \gamma]$ of minimum and maximum travel time. In Figure 1, we represent aircraft A traveling in an air segment. The travel time constraint is modelled with a pair of fixed arcs $(i, \sigma(i))$ and $(\sigma(i), i)$. Arc $(i, \sigma(i))$ is weighted with $w_{i,\sigma(i)} = \phi$ while arc $(\sigma(i), i)$ is weighted with $w_{\sigma(i),i} = -\gamma$. These constraints impose that the entrance time in the next air segment $\sigma(i)$ must be $t_i + \phi \leq t_{\sigma(i)} \leq t_i + \gamma$. We observe that when $w_{i,\sigma(i)} = -w_{\sigma(i),i}$ the travel time constraints correspond to a strict no-wait constraint, as in the ASP model of [10].

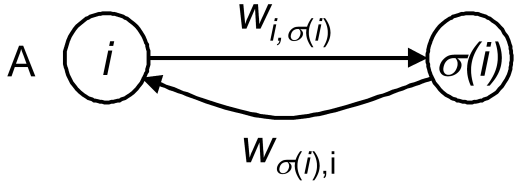

Figure 1: Alternative graph formulation of a [min,max] travel time constraint.

Air traffic regulations impose that all aircraft flying on the same air segment must respect a minimum longitudinal separation depending on their category (heavy, medium or light). This separation translates into a STI between the entrance/exit of consecutive

aircraft in the same air segment. Overtaking is not allowed within an air segment. In this paper the minimum separation time between consecutive aircraft in the same air segment is modelled as a sequence-dependent setup time [3, 10]. We also consider the special case in which two parallel air segments are very close to each other (e.g. a common glide path), and the minimum longitudinal distance between aircraft is not sufficient to satisfy traffic regulations. An additional minimum diagonal distance is therefore imposed between aircraft flying on the parallel air segments. This traffic situation is modelled as a single air segment resource in which setup times do not depend on the aircraft sequence only but on the route chosen for each aircraft also.

Figure 2(a-b) shows how we model the sequencing decisions for two aircraft on an air segment. In the example, two nodes represent the entrance (i) of aircraft A in an air segment and its exit, i.e., the entrance ($\sigma(i)$)in the subsequent air segment. Similarly, nodes j and $\sigma(j)$ are used for aircraft B. The no overtaking constraint and the minimum separation distance between aircraft is obtained by adding two alternative pairs $((i,j),(\sigma(j),\sigma(i)))$ and $((j,i),(\sigma(i),\sigma(j)))$. The weights on the alternative arcs correspond to the minimum STI between A and B in the air segment. The only two ways to select alternative arcs from the two pairs without generating positive weight cycles in the graph are either to select both arcs (i,j) and $(\sigma(i),\sigma(j))$ (i.e., aircraft A precedes B at both the entrance and exit of the air segment), or to select both arcs (j,i) and $(\sigma(j),\sigma(i))$ (i.e., B precedes A at both locations).

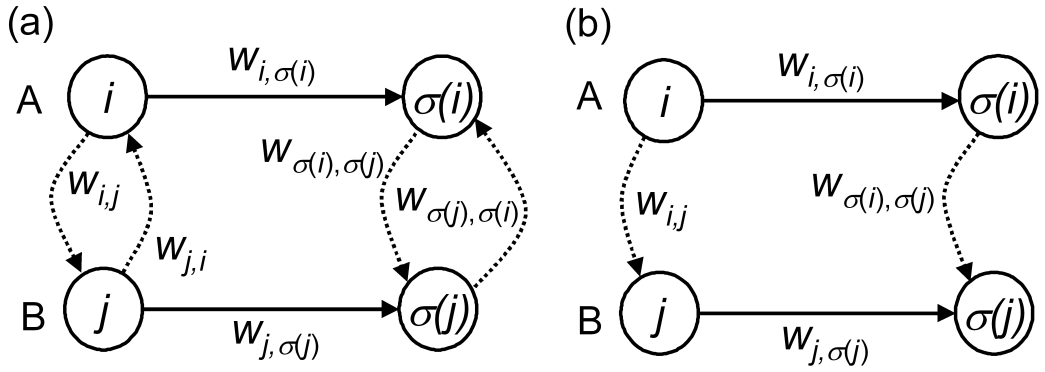

Figure 2: Alternative graph formulation of an air segment.

In Figure 2(b), A is before B and the minimum STI at the entrance will be $w_{i,j}$ while the minimum STI at the exit will be $w_{\sigma(i),\sigma(j)}$.

The aircraft sequencing on a runway is formulated by an alternative pair for each pair of aircraft landing or taking off from that runway. The runway is modelled as a blocking resource [17], since the presence of one aircraft on the runway imposes that no other aircraft can use it. If there are intersecting runways, no two aircraft can use them simultaneously, i.e. intersecting runways behave as a single blocking resource. In Figure 3(a), two aircraft, A and B, require the same runway. Node i and node j represent A and B entering the runway, while nodes $\sigma(i)$ and $\sigma(j)$ represent A and B leaving the runway (i.e., entering the subsequent resource). The weight of the fixed arcs $w_{i,\sigma(i)}$ and $w_{j,\sigma(j)}$ model the time to process the runway by aircraft A and B. The alternative pair $((\sigma(i),j),(\sigma(j),i))$ is used to model the sequencing decision between A and B. If A precedes B then the alternative arc $(\sigma(i),j)$ is selected, otherwise $(\sigma(j),i)$ is selected (and

B precedes A on the runway). The safety separation constraints a first aircraft leaving the runway and a second aircraft entering the runway are represented by the weights $w_{\sigma(i),j}$ (if A precedes B) and $w_{\sigma(j),i}$ (if B precedes A). In Figure 3(b), A precedes B and B can enter the runway only after $t_j \geq t_i + w_{i,\sigma(i)} + w_{\sigma(i),j}$.

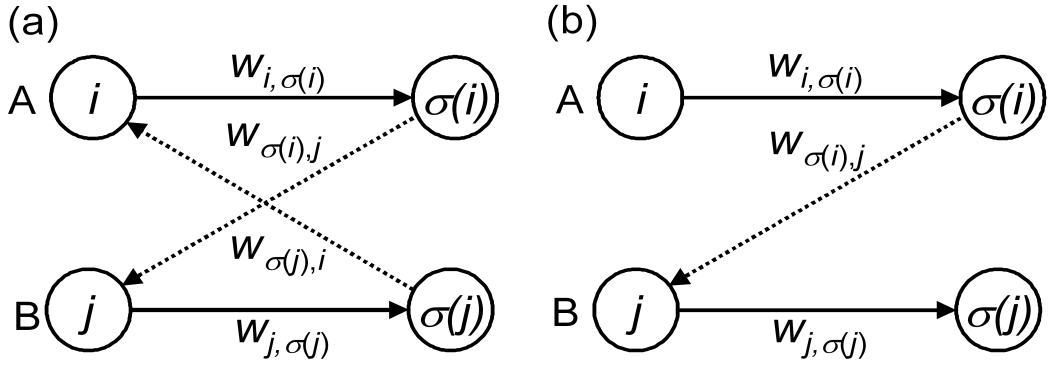

Figure 3: Alternative graph formulation of a runway.

Holding circles are used by traffic controllers to let landing aircraft wait in flight when the TMA is congested. When an aircraft enters the airborne holding in response to a disturbed traffic situation, it must fly at a given speed and can leave the holding circle only after the traversing of (a multiple of) half length of the circle. We model the holding circle decisions via alternative pairs and the holding circle constraints via fixed arcs in the graph. In Figure 4(a-b), the possible entrance of a landing aircraft in a holding resource is modelled. Two possible decisions are modelled: (i) the aircraft travel a half circle of weight β or (ii) the aircraft skip the holding circle.

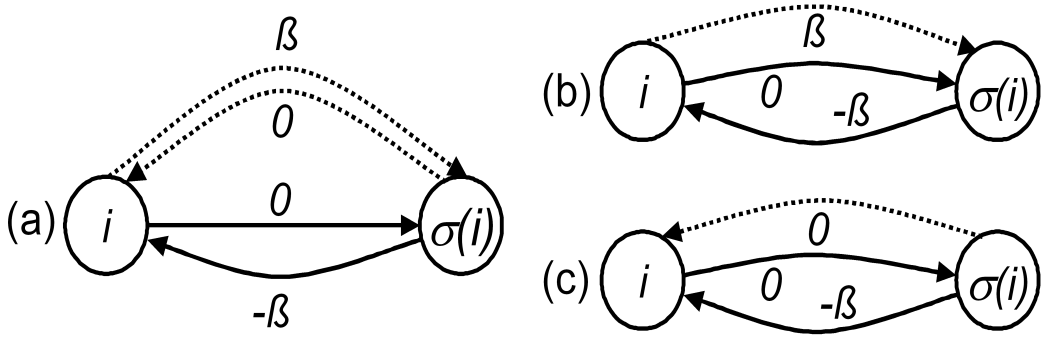

Figure 4: Alternative graph formulation of a holding circle.

In Figure 4(a), the two alternative decisions are formulated with a pair of alternative arcs $((i,\sigma(i)),(\sigma(i),i))$, weighted β and 0. The holding circle constraints are modelled by the two fixed arcs $(i,\sigma(i))$ and $(\sigma(i),i)$ with weights 0 and $-\beta$. In a solution, one of the two alternative arcs must be chosen. Figure 4(b) shows the solution (i) in which the arc $(i,\sigma(i))$ is selected. This forces the aircraft to spend exactly β time units in the holding circle (since it causes a zero weight cycle with the fixed arc $(\sigma(i),i)$, thus imposing $t_{\sigma(i)} = t_i + \beta$). Figure 4(c) shows the solution (ii) in which the arc $(\sigma(i),i)$ is selected. This forces the aircraft to skip the holding cycle (since it causes a zero weight cycle with

the fixed arc $(i, \sigma(i))$, thus imposing $t_{\sigma(i)} = t_i$). A description of how to extend our model to multiple circles can be found in Pistelli et al. [23], in which the alternative graph formulation of a given number of half circles is shown, with suitable arc weights. In this work, we performed experiments by considering multiple (half) circles at the airborne holding resources.

Figure 5 shows release, due date and deadline arcs. The *release time* is the minimum time at which the aircraft must enter the TMA. The fixed arc $(0, i)$ models the release time r_i of o_i with weight $w_{0,i} = r_i$. The *deadline time* is the maximum time at which the aircraft can enter the TMA. The fixed arc $(i, 0)$ models the deadline time $\bar{d}_i$ of o_i with weight $w_{i,0} = -\bar{d}_i$ If o_i is late (i.e., $l^S_{0,i} + w_{i,0} > 0$) there is a positive cycle in the graph and the resulting schedule is infeasible.

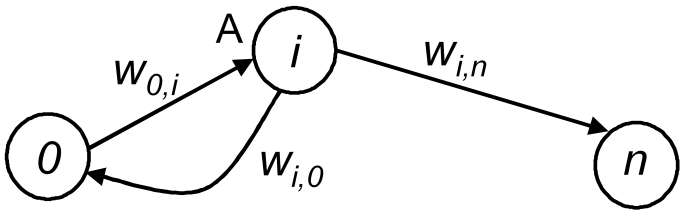

Figure 5: Alternative graph formulation of a release, due date and deadline constraints.

Arc (i, n) in Figure 5 is used to model the objective function of the ASP. The *due date time* d_i is the scheduled time of an aircraft on a specific resource, e.g., the landing/take-off time shown in the timetable. The fixed arc (i, n), with weight $w_{i,n} = -d_i$, is a due date arc that we use to measure the delay of the associated aircraft at the starting of operation i. If in a feasible schedule $t_i > d_i$, then the delay of the solution must be $t_n \geq t_i - d_i$. Therefore, given a feasible schedule S and since $t_i = l^S_{0,i}$, we have that $l^S_{0,n}$ corresponds to the maximum delay associated to S. The choice of the due date d_i allows to compute several components of the aircraft delay. Specifically, let α_i be the scheduled starting time of o_i in the timetable, and let τ_i be its earliest possible starting time compatible with the position of the associated aircraft at the beginning of traffic prediction, and disregarding the presence of other aircraft. If $\tau_i > \alpha_i$, the difference $\tau_i - \alpha_i$ is an *unavoidable delay* that cannot be recovered by rescheduling the aircraft.

Given a feasible schedule S such that $t_i > \alpha_i$, the quantity $t_i - \alpha_i$ denotes the *total delay* of o_i in S, and $t_i - \max\{\tau_i, \alpha_i\}$ denotes the *maximum consecutive delay* of o_i in S, since the unavoidable delay $\tau_i - \alpha_i$ is omitted [12]. Hence, if we choose a due date $d_i = \alpha_i$ for each operation with a scheduled starting time, then $l^S_{0,n}$ is the *maximum total delay* of S. Similarly, if we choose $d_i = \max\{\alpha_i, \tau_i\}$ for each operation with a scheduled starting time, then $l^S_{0,n}$ is the *maximum consecutive delay* of S. In this paper we minimize the latter quantity. The ASP objective is thus the minimization of the maximum consecutive delay over all aircraft.

For take-off operations, we follow the procedure commonly adopted by air traffic controllers that consider a time window for take-off between 5 minutes before and 10 minutes after the Scheduled Take-off Time (STT) [9]. Therefore, a departing aircraft is late if leaving the airport after 10 minutes from its STT, i.e., the value α associated to the take-off operation is STT plus 10 minutes. For each landing aircraft we consider a value

α equal to its Scheduled Landing Time (SLT).

3.2 Alternative graph models of the ASP

We next describe three models for the collaborative and non-collaborative ASP.

Model 1 is a non-collaborative model in which the entrance time of each aircraft in the TMA is a fixed value r_i received in input from neighboring areas. The entrance time constraint is modelled by adding a release arc with weight r_i and a deadline arc with weight $-r_i$ to the ASP model for each aircraft.

Model 2 is a collaborative model in which the entrance time of each aircraft in the TMA can be defined in cooperation with the traffic controllers of neighboring areas. An optimal solution for this model will fix the "best" entrance times, i.e., such that the maximum consecutive delay of the aircraft schedule is minimized. Clearly, delaying the entrance of a landing aircraft in the TMA has an impact on neighboring areas. In order to limit this impact, a late entrance is penalized in the objective function. This model is also a simplification of Model 1, since the holding circle decisions are not included. In other words, the entrance in the TMA is located at the exit from the holding circles, so that the time spent in a holding circle is penalized as a late entrance.

Model 3 is also a collaborative model. The difference with Model 2 is that the entrance in the TMA is located at the entrance in the holding circles, i.e., holding circle decisions are now included in the ASP model and only the delay at the entrance of the holding circles is penalized in the model. In other words, with this model a late entrance better corresponds to the impact on neighboring areas than in Model 2.

We observe that, deadline arcs increase the complexity of Model 1 significantly with respect to Models 2 and 3, since the ASP formulation is severely constrained. On the other hand, in order to implement in practice the solutions provided by the latter models, a larger degree of flexibility is necessary to set the entrance time of each aircraft in the TMA. Changing the aircraft entrance time requires not only a coordination action between the controllers of the TMA and the controllers of neighboring areas. The change is physically possible only if the schedules are found sufficiently in advance with respect to the start time t_0 of traffic prediction, to let enough time to the en-route controllers to delay the approaching time of incoming aircraft according to the ASP solution.

3.3 Illustrative example of the FCO airport

Figure 6 shows a traffic situation with 2 landing aircraft (A and B) and 1 departing aircraft (C). At the FCO airport, there are three runways (16L, 16R, 25), two of which (16R, 25) are intersecting and cannot be used simultaneously. The TMA of this airport is divided into 3 holding circles (TAQ, CMP, CIA; numbered from 1 to 3), 7 air segments for landing procedures (4 to 10), 3 air segments for take-off procedures (14 to 16) leading to the exit points of the TMA (ELVIN, RAVAL, BOL), two runways (12 and 13) and a common glide path (11).

In the traffic situation of Figure 6, there are potential conflicts at two air segments (4 and 11) and at one runway (12), since these resources are used by more than one aircraft.

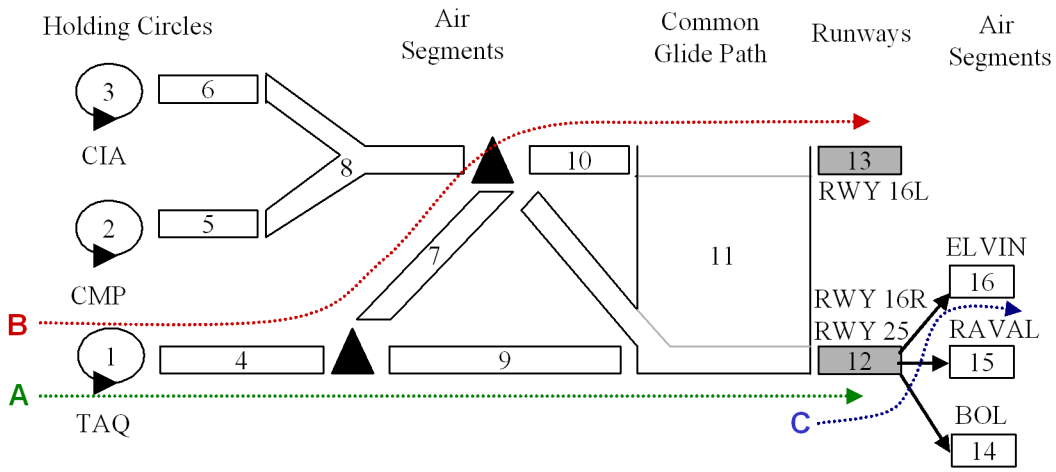

Figure 6: A traffic situation of the FCO airport with landing (A,B) and departing (C) aircraft.

For this example situation Figures 7, 8 and 9 present the AG formulation of the three models of Section 3.2. Each node of the graph represents an operation, e.g., $A9$ is aircraft A entering air segment 9. Fixed (alternative) arcs are depicted with solid (dotted) arrows. For the sake of simplicity, the weight of fixed and alternative arcs is not shown in the figures. For a detailed description of all arc weights, we refer the reader to Section 3.1.

The departing aircraft C is constrained to enter the runway after its minimum departure time, which is formulated with a release arc $(0, C12)$. The landing aircraft A and B are constrained to enter the TMA after their minimum departure time.

Two holding circle decisions are modelled for each landing aircraft: travel or not half circle of weight β. Two alternative arcs model these decisions: $((A1, A4), (A4, A1))$ and $((B1, B4), (B4, B1))$.

Each aircraft must traverse each air segment along its route within a window of [minimum, maximum] traversing times, e.g., arcs $(A9, A11)$ and $(A11, A9)$ for the air segment 9. The potential conflicts on air segments are modelled by four alternative pairs: $((A4, B4), (B7, A9))$ and $((B4, A4), (A9, B7))$ for air segment 4, $((A11, B11), (B13, A12))$ and $((B11, A11), (A12, B13))$ for the common glide path 11.

Aircraft sequencing decisions on runway 12 are modelled by the pair $((C16, A12), (Aout, C12))$. In order to minimize the maximum consecutive delay on the runways, the due date arcs $(Aout, n)$, $(Bout, n)$ and $(C16, n)$ are weighted with $w_{(Aout,n)} = -\max\{\tau_{Aout},\ \alpha_{Aout}\}$, $w_{(Bout,n)} = -\max\{\ \tau_{Bout},\ \alpha_{Bout}\}$ and $w_{(C16,n)} = -\max\{\ \tau_{C16},\ \alpha_{C16}\}$, where α and τ are defined in Section 3.1.

Figure 7(a-b) shows Model 1, in which the entrance time of each landing aircraft is taken as a constraint for the problem. This formulation is shown in Figure 7(a), with release and deadline arcs at the first operation of landing aircraft in the TMA: for aircraft A the arcs $(0, A1)$ and $(A1, 0)$ with weight $w_{0,A1}$ and $-w_{0,A1}$; for B $(0, B1)$ and $(B1, 0)$ with weight $w_{0,B1}$ and $-w_{0,B1}$. With these constraints, the entrance time of the two aircraft must be equal to $w_{0,A1}$ and $w_{0,B1}$.

Figure 7(b) shows a feasible schedule for Model 1. This solution is obtained by scheduling first aircraft A on the air segment 4 (selecting the arcs $(A4, B4)$ and $(A9, B7)$), on the common glide path 11 (selecting the arcs $(A11, B11)$ and $(A12, B13)$), and on the

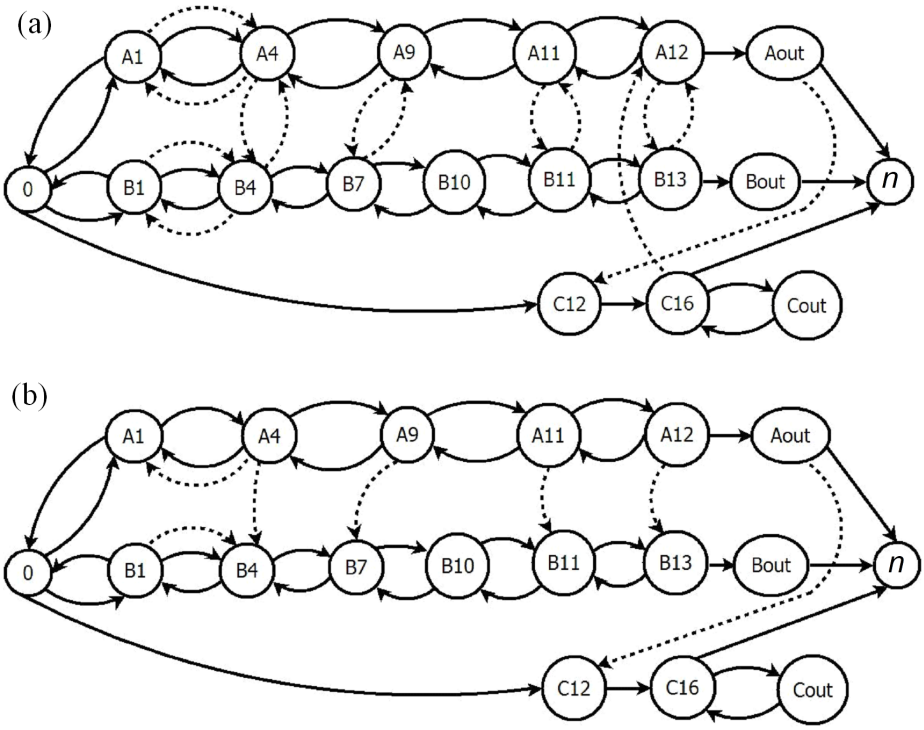

Figure 7: The Model 1 formulation (a) and solution (b) for the example in Figure 6.

runway 12 (selecting the arc $(Aout, C12)$). Aircraft A skips the holding circle (selecting the arc $(A4, A1)$ with weight 0). Aircraft B runs a half circle of weight β (selecting the arc $(B1, B4)$) in order to avoid the potential conflict with aircraft A on the air segment 4 that would cause a positive cycle in the graph due to the deadline constraint $(B1, 0)$.

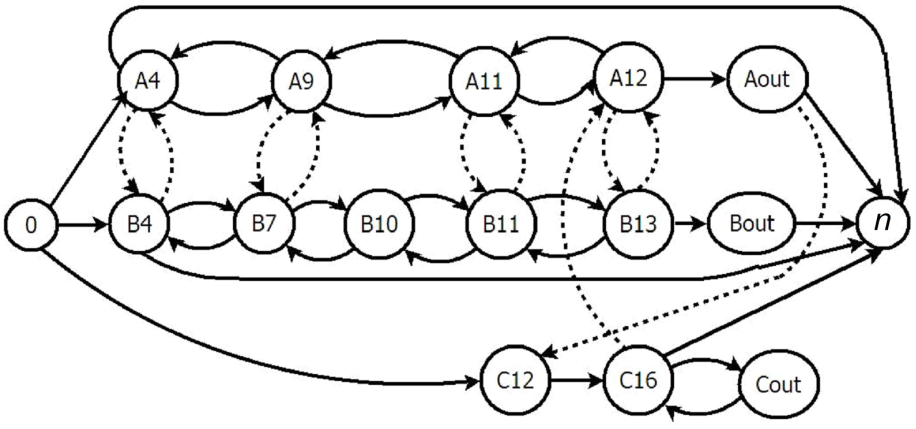

Figure 8: The Model 2 formulation for the example in Figure 6.

In Model 2, holding circle and deadline constraints are replaced with respect to Model 1 by the due date arcs $(A4, n)$ and $(B4, n)$ with weights $w_{(A4,n)} = -\max\{\tau_{A4},\ \alpha_{A4}\}$ and $w_{(B4,n)} = -\max\{\tau_{B4},\ \alpha_{B4}\}$ (see Figure 8). As a result, each aircraft can spend an arbitrary amount of time in the holding circle. However, the time spent in the holding circle is penalized in the objective function. In other words, with Model 2 the delay of landing aircraft is computed both at the entry fix and at the runway, while the delay of

departing aircraft is only computed at the runway.

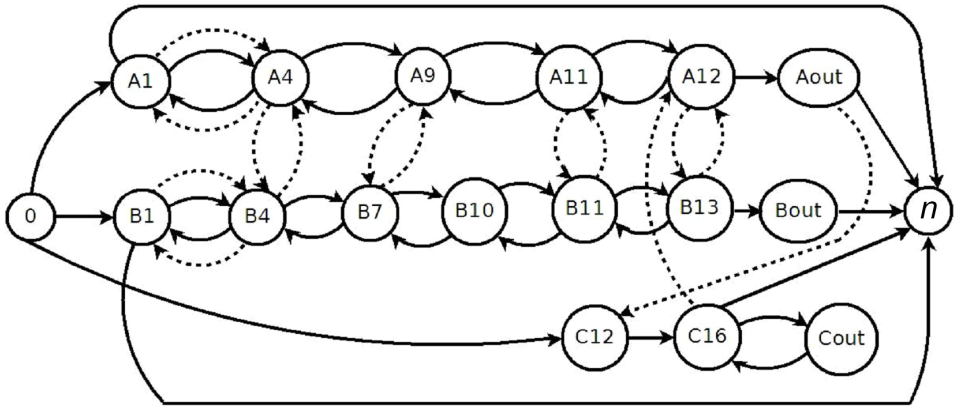

Figure 9: The Model 3 formulation for the example in Figure 6.

In Model 3, the holding circle decisions are modelled as for Model 1. Differently from Model 1, deadline constraints are replaced by the due date arcs $(A1, n)$ and $(B1, n)$ with weights $w_{(A1,n)} = -\max\{\tau_{A1}, \alpha_{A1}\}$ and $w_{(B1,n)} = -\max\{\tau_{B1}, \alpha_{B1}\}$ (see Figure 9). With this formulation, the delay with respect to the minimum entrance time in the TMA is penalized in the objective function. The aircraft delay of landing and departing aircraft is computed as for Model 2.

4 Scheduling algorithms

This section presents the scheduling algorithms used to solve the ASP. Note that even the restricted version of the ASP in which the aircraft must be sequenced only on a single runway is strongly NP-hard since it includes the open traveling salesman problem as a special case [7]. Therefore, in the following we focus on heuristic and branch and bound algorithms. We first describe a constraint propagation technique to speed-up the computation of feasible schedules and then describe five scheduling algorithms, ranging from simple dispatching rules to advanced exact approaches.

4.1 Static implications

In the development of our algorithms we extend to the ASP the concept of *static implications* introduced in [12] for blocking resources. A static implication occurs when, given two unselected pair of alternative arcs $((i, j), (h, k))$ and $((a, b), (c, d))$, it is possible to prove that the selection of (i, j) and (a, b) is infeasible. Therefore, in a feasible schedule the selection of (i, j) *implies* the selection of (c, d). In [12], static implications are shown to be very effective for train scheduling problems. The following proposition extends the concept of static implications to air segments (see the alternative graph formulation of two aircraft traversing an air segment in Figure 2).

Proposition 4.1 *Consider two unselected alternative pairs for an air segment* $((j, i), (\sigma(i), \sigma(j)))$ *and* $((i, j), (\sigma(j), \sigma(i)))$. *If* $w_{i,j} + w_{j,i} > 0$ *then arc* $(\sigma(i), \sigma(j))$ *is implied by the selection of* (i, j) *and arc* $(\sigma(j), \sigma(i))$ *is implied by the selection of* (j, i). *If* $w_{\sigma(i),\sigma(j)} + w_{\sigma(j),\sigma(i)} > 0$

then arc (i, j) *is implied by the selection of* $(\sigma(i), \sigma(j))$ *and arc* (j, i) *is implied by the selection of* $(\sigma(j), \sigma(i))$.

Proof. The result follows from the observation that the selections $\{(i, j), (j, i)\}$ and $\{(\sigma(i),\sigma(j)), \sigma(j), \sigma(i)\}$ contain a positive weight cycle and are therefore infeasible. □

The intuition behind Proposition 4.1 is that aircraft overtaking is not allowed in the proximity of the airport. Therefore, once a corridor is detected all the alternative pairs between two jobs passing in the corridor are implied and the sequencing decision of an aircraft preceding another aircraft can be propagated to all their air segments if the two aircraft follow the same route. This constraint propagation technique is utilized during the execution of the algorithms to speed-up the search process.

4.2 Practical scheduling rule

The *First In First Out* (FIFO) rule solves the aircraft conflicts one at a time by assigning each conflicting resource to the first aircraft requiring it. This local rule requires no look-ahead control action since aircraft run in the airport area on the basis of their actual order of arrival at each air segment or runway. According to [9], the FIFO rule is close to the ATC control practice, even if human controllers may adjust the FIFO sequence in order to recover possible infeasibilities.

4.3 Greedy heuristics based on arc selection

The *Arc Greedy Heuristic* (AGH) is a family of heuristic algorithms which takes decisions based on global information available from the alternative graph formulations [24]. A sketch of the general structure of these algorithms is given in Figure 10.

The family is based on the idea of repeatedly enlarging a selection by choosing an unselected pair at a time from set A and by selecting one of the two arcs until a feasible schedule is found or an infeasibility (i.e., positive weight cycle in the graph) is detected. The heuristics in the AGH family differ from each other for the *evaluation criteria* used to choose the next unselected arc from A. Every time an alternative arc (i, j) is selected, all alternative arcs implied by Propositions 4.1 are also selected. This set is called $Stat(i, j)$ in the algorithm sketched in Figure 10. If for a given selection S there is an arc $(a, b) \in Stat(i, j)$ such that $((a, b), (c, d)) \in A$ and $(c, d) \in S$, we say that (a, b) is *forbidden*, since the addition of (i, j) to S leads to an infeasible solution.

Several evaluation criteria have been introduced in [24]. Here we present only two evaluation criteria, which from preliminary tests turn out to be the best performing for this problem. The AMCC (*Avoid Most Critical Completion time*) criterion chooses an unselected alternative pair $((i, j), (h, k))$ maximizing the quantity $\max\{l^S(0, i) + w_{i,j} + l^S(j, n), l^S(0, h) + w_{h,k} + l^S(k, n)\}$ and selects the arc causing the minimum delay [22]. In other words, this criterion tries to avoid the arc that would cause the largest increase of $l^{S'}(0, n)$ in $G(S')$. The AMSP (*Avoid Most Similar Pair*) criterion chooses an unselected alternative pair $((i, j), (h, k))$ maximizing the quantity $l^S(0, i) + w_{i,j} + l^S(j, n) + l^S(0, h) + w_{h,k} + l^S(k, n)$ and selects the arc causing the minimum delay [24]. In other words, this criterion chooses a pair with the largest sum of consecutive delays and selects the arc causing the smallest delay.

Algorithm AGH

begin
Set $S := \emptyset$
while $A \neq \emptyset$ **do**
begin
Choose a pair $((i,j),(h,k)) \in A$ according to a pre-defined evaluation criterion,
Let $S' := S \cup \{(i,j)\}$, $A' := A \setminus \{((i,j),(h,k))\}$,
$S' := S' \cup Stat(i,j)$, remove from A' the pairs associated to the arcs in $Stat(i,j)$,
if (there is a cycle in $G(S')$) **or** (an arc from $Stat(i,j)$ is forbidden) **then**
begin
Let $S' := S \cup \{(h,k)\}$, $A' := A \setminus \{((i,j),(h,k))\}$,
$S' := S' \cup Stat(h,k)$, remove from A' the pairs associated to the arcs in $Stat(h,k)$,
if (there is a cycle in $G(S')$) **or** (an arc from $Stat(h,k)$ is forbidden) **then**
begin
The procedure fails in computing a feasible schedule, STOP
end
end
Let $S := S'$, $A := A'$.
end
end

Figure 10: Sketch of the AGH heuristics

4.4 Algorithms based on job selection

The *Job Greedy Heuristic* (JGH) is a new family of constructive algorithms which combines different AGH algorithms. JGH iteratively extends a partial schedule until a complete graph selection is found or an infeasibility is detected. The decision taken at each step consists in the insertion of a complete job in the current partial schedule, keeping fixed the sequence of the previously scheduled jobs. A sketch of the general JGH structure is given in Figure 11, in which we use the following notation. We use J to the denote the set of jobs, j_i to denote the set of nodes associated to the operations of the i-th job in J, $\bar{N}$ to denote the set of nodes of scheduled jobs. Finally, y denotes the number of AGH criteria.

As shown in Figure 11, at each iteration all the unscheduled jobs are evaluated for insertion as candidate by using y AGH criteria and the best criterion for each job $i \in U$ is stored with the associated selection S^i. The best candidate job *bestjob* is inserted in the current partial schedule S according to the associated selection $S^{bestjob}$. Note that, at each iteration, the relative order among the previously selected jobs remains fixed. The procedure is repeated for at most $|U|$ times until a feasible schedule S is obtained or infeasibility is detected.

4.5 Branch and bound algorithm

Branch and bound algorithms implicitly explore the whole set of solutions to find a globally optimal schedule. In order to speed up the search, lower bound and implication techniques

Algorithm JGH

begin
Set $\bar{N} := \emptyset$ (set of nodes of scheduled jobs)
Set $U := J$ (set of unscheduled jobs)
Set $S := \emptyset$ (current selection)
while $U \neq \emptyset$ **do**
  begin
  $bestjob := null$, $f(S^{bestjob}) = \infty$,
  forall $i \in U$ **do**
    begin
    Let $\bar{A} := \{((a,b),(c,d)) \in A : (a \in \bar{N} \wedge b \in j_i) \vee (b \in \bar{N} \wedge a \in j_i)\}$
    $f(S^i) = \infty$, (objective for the best insertion of job j_i)
    for $h = 1, \ldots, y$ **do**
      begin
      Starting from the alternative graph $\mathcal{G}(S) = (N, F \cup S, \bar{A})$,
      Compute a schedule by repeatedly applying the h-th AGH criterion,
      Denote with S^i_h the resulting selection,
      if $f(S^i) > l^{S \cup S^i_h}(0,n)$ **then** $f(S^i) = l^{S \cup S^i_h}(0,n)$ and $S^i = S^i_h$, (choose best AGH)
      end
    if $f(S^i) < f(S^{bestjob})$ **then** $bestjob := i$, $S^{bestjob} = S^i$,
    end
  if $bestjob = null$ **then** no feasible schedule found, STOP.
  else $\bar{N} = \bar{N} \cup j_{bestjob}$, $U := U \setminus \{bestjob\}$, $S := S \cup S^{bestjob}$ (insert $bestjob$ in the schedule)
  end
end

Figure 11: Sketch of the JGH heuristic.

are applied. Whenever the lower bound on the objective function for a subset of all solutions becomes larger than or equal to a known upper bound on the global optimum, the whole subset can be pruned.

Implication techniques allow to enlarge a partial selection S whenever it can be proved that for some unselected pair of alternative arcs $((i,j),(h,k))$ the selection $S \cup (i,j)$ is infeasible. In this case (h,k) must be selected and we say that S *implies* (h,k) and, from proposition 4.1, also the set $Stat(h,k)$.

When all the search space is explored, the branch and bound algorithm stops returning the proven optimum. If the procedure is terminated before completing the search, the procedure returns a lower bound and an upper bound on the global optimum.

The exact algorithms we consider are adaptations of the *Branch and Bound* (denoted as BB) algorithm described in [12], developed for a train scheduling problem and now modified to solve the ASP. In the following, we limit ourselves to briefly describe the main components of the algorithm and the modification with respect to the algorithm described in [12].

The initial upper bound is obtained by running the four heuristics proposed above (FIFO, AMCC, AMSP, JGH). At each node of the enumeration tree (associated to a selection S), the BB algorithm computes a lower bound with an adaptation of the single machine Jackson Preemptive Schedule $JPS(S)$ described in [19]. The JPS is computed for each air segment and each runway. Implication techniques described in [12] are also utilized to speed up the computation, even if static implications are adapted to deal with air segment resources by using Proposition 4.1. The search strategy we use alternates four repetitions of the depth-first visit with the choice of a selection S with the smallest value of $JPS(S)$ among the last five generated selections (i.e., best-first visit limited to the last selections enumerated). The choice of this hybrid strategy is motivated by the need for searching new feasible solutions (by using the depth-first strategy) and promising selections (by using the best-first strategy), the numerical values being the result of a tuning carried out on 24 ASP instances.

We use a binary branching scheme in order to choose an unselected pair $((i,j),(h,k))$ for branching. We explored two branching strategies: in the first case we branch with priority on alternative pairs associated to runway resources, in the second case all alternative pairs have the same priority. The first strategy is motivated by the fact that the runways are often the bottleneck resources of the TMA. For both strategies, we use the AMSP criterion to choose the next unselected pair for branching, since this criterion has been the most successful for a preliminary set of experiments on the 24 instances.

In Section 5 we study the two branching strategies in order to select the best one for each of the three ASP models.

5 Computational experiments

The tested instances are based on observations during a busy hour of national and international aircraft arrivals and departures at the FCO airport. We consider an entry fix for each aircraft in the TMA. We also assume that the routing problem is solved off-line, so that the workload of the runways is well balanced and there is no conflict at runways when aircraft are on time. In our models we fix a maximum of three round trips for each aircraft in a holding circle. The experiments are executed on a processor Intel i7

(2.84 GHz), 8 GB Ram and Linux operating system. The AGLibrary, developed by the Operations Research Group of Roma Tre University, is used to implement and test the alternative graph models and algorithms.

This section is organized as follows. We first describe our set of instances based on the FCO airport and randomly generated disturbed traffic conditions. We then assess a number of versions of our branch and bound algorithm and compare the branch and bound algorithm with the heuristic approaches in terms of feasibility rate, computation time and solution quality. Finally, we study the different alternative graph models of the ASP via the branch and bound algorithm in terms of its objective function and other performance indicators, and explore its computational limit for large time horizons of traffic prediction. We also discuss the results and possible improvements.

5.1 Description of the ASP instances

Table 1 presents the main characteristics of 240 instances we use to test the scheduling algorithms. For each model of Section 3, we generate 80 disturbed traffic conditions based on random entrance delays for the landing and departing aircraft. Each row of Table 1 reports average data over 20 instances for a given model (Column 1). Column 2 gives four time horizons of traffic prediction, ranging from 15 minutes to 60 minutes. Column 3 shows the number of arriving/departing aircraft. The resulting graphs are described in Columns 4–6 in terms of the number of nodes ($|N|$), the fixed arcs ($|F|$) and the pairs of alternative arcs ($|A|$).

Table 1: Perturbed traffic situations and prediction horizons

Graph Model	Time Horiz	Arr/Dep Aircraft	$\|N\|$	$\|F\|$	$\|A\|$	Max Delay	Avg Delay	Delayed Aircraft
1	15	6/1	44	290	103	450	170.4	4
1	30	16/4	118	1605	688	900	234.7	10
1	45	22/7	166	2906	1292	1350	345.6	14
1	60	32/16	259	6425	2963	1800	465.1	24
2	15	6/1	38	218	67	450	170.4	4
2	30	16/4	102	1413	592	900	234.7	10
2	45	22/7	144	2642	1160	1350	345.6	14
2	60	32/16	227	6041	2771	1800	465.1	24
3	15	6/1	44	290	103	450	170.4	4
3	30	16/4	118	1605	688	900	234.7	10
3	45	22/7	166	2906	1292	1350	345.6	14
3	60	32/16	259	6425	2963	1800	465.1	24

In Table 1, we introduce four time horizons of traffic prediction. The two shortest time horizons (15 and 30 minutes) are interesting for real-time traffic control, since landing aircraft enter the landing planner's radar range around 30 minutes before touch down, as reported e.g. by Bennell et al. [9]. The two longest time horizons (45 and 60 minutes) are studied in order to evaluate the computation time and solution quality of our exact algorithm for challenging instances. For each time horizon of traffic prediction, we generate 10 Uniform and 10 Gaussian (randomly generated) entrance delays in the TMA. Columns 7–8 of Table 1 report on the perturbation characteristics as the maximum and

average entrance delays (in seconds) and Column 9 shows the number of aircraft delayed at their entrance in the area under study, before solving the ASP problem (see e.g. Figure 6(a)). We only delay aircraft that enter the network in the first half of each time horizon in order to study the delay propagation in the second half of the time horizon.

Models 1 and 3 have the same number of fixed arcs, since each landing aircraft has one deadline arc and one due date arc in Model 1 while each landing aircraft has two due date arcs in Model 3. The number of nodes and alternative arcs is also the same, since Models 1 and 3 have the same number of decision aircraft timing and ordering variables. Differently from Models 1 and 3, Model 2 has less nodes, fixed and alternative arcs since the fixed and alternative arcs of the holding circle constraints are not included in the corresponding graph. The three models are tested on the same entrance delay configurations.

5.2 Assessment of branching strategies

In this subsection we assess the performance of the branch and bound algorithm (BB) and the two branching strategies described in Section 4.5 with the three ASP models. We name BB1 the branch and bound algorithm when the branching strategy gives priority to the runway resources, and BB2 the strategy with no priority.

Table 2: Impact of scheduling the runways first on the three models

Graph Model	BB Version	BB Time Tot	Feas Sched	Opt Solut	Max Delay	BB Time Best	Iter Best
1	BB1	86.50	56	27	341.5	30.3	105862
	BB2	76.83	80	35	323.7	18.5	58451
2	**BB1**	40.36	80	56	429.7	1.1	1031
	BB2	55.15	80	49	430.0	2.4	5615
3	**BB1**	40.07	80	56	435.5	0.6	775
	BB2	68.96	80	37	439.1	4.0	12577

Table 2 shows the performance of two BB versions, each truncated after 120 seconds of computation. Each row presents the average behavior over the 80 instances of Model 1 (with holding circles and deadlines), or Model 2 (without holding circles and deadlines), or Model 3 (without deadlines). Columns 1 and 2 report the type of model and the BB version. Column 3 presents the average over all the instances of the total BB computation time (in seconds), i.e., the time to either prove optimality or to reach the time limit, excluding the computation time of the initial heuristics. Columns 4–5 present the average over all the instances of the number of feasible schedules and the number of times BB proves optimality. For the instances solved by both BB versions, Column 6 reports the average value of the objective function (i.e., the maximum consecutive delay in seconds). Column 7 presents the BB computation time to reach the best solution found (in seconds), excluding the time of the initial heuristics. The last column shows the number of iterations to find the best solution.

From Table 2 we conclude that Model 2 is easier to solve than the other models. This finding is somewhat expected, since in Model 2 the holding circle and deadline constraints are relaxed, therefore the number of alternative pairs is smaller than in the other models. Nevertheless, Models 2 and 3 perform similarly, thus showing that the complexity of

Model 1 is more related to the deadline constraints than to the presence of holding circle constraints or on the number of alternative pairs.

For Model 1, BB1 found an infeasibility in more than 20% of the instances. None of the two BB versions outperforms the other, since the best versions (in bold in Table 2) depend on the model being used. BB2 solves best Model 1, whereas BB1 solves best Models 2 and 3.

The different behavior of BB with different branching schemes is motivated by the presence of a deadline constraint at the entry fix for each landing aircraft in Model 1. These additional constraints make it difficult to predict the feasibility of a partial selection containing alternative arcs related to the runway resources. This difficulty reduces the quality of the lower bound when branching with priority on the runway resources, thus slowing down BB1 with respect to BB2. In what follows, we report under BB the performance of the best branch and bound version for each model (BB2 for Model 1 and BB1 for Models 2 and 3).

5.3 Assessment of static implications

Table 3 shows the performance of the two BB versions: with ('On') or without ('Off') static implications. Both versions are truncated after 120 seconds of computation. Each row presents the average behavior over the 80 instances for each model. Columns 1–2 report the type of model and the BB version. The other columns report the same type of information as in Table 2.

Table 3: Effects of static implications on the three models

Graph Model	Static Implic	BB Time Tot	Feas Sched	Opt Solut	Max Delay	BB Time Best	Iter Best
1	**On**	76.83	80	35	552.8	17.8	46324
	Off	76.98	80	35	553.7	18.6	55114
2	**On**	40.36	80	56	429.7	1.1	1031
	Off	40.48	80	56	429.7	1.2	1042
3	**On**	40.07	80	56	435.5	0.6	777
	Off	40.26	80	56	435.7	0.8	791

From Table 3 we conclude that, on average, the branch and bound version that uses static implications is the best configuration in terms of the objective function value (i.e., the maximum consecutive delay). The gap between the two BB configurations is small since static implications are related to the propagation of aircraft ordering decisions between consecutive air segments and the TMA under study presents a limited number of air segment for each aircraft route. We expect a greater impact of static implications when managing a larger airspace, as it occurs, e.g., in en-route traffic management.

In the next subsections, when referring to BB we mean the branch and bound version that uses static implications for all air segments.

5.4 Feasibility rate of the scheduling algorithms

Table 4 shows the number of feasible schedules computed by each scheduling algorithm. Each row shows the results on the 20 instances of a specific model and time horizon.

Table 4: Feasibility rate of the various algorithms

Graph Model	Time Horiz	AMCC	AMSP	JGH	FIFO	BB
1	15	7	7	0	5	20
1	30	0	0	0	0	20
1	45	0	0	0	0	20
1	60	0	0	0	0	20
2	15	20	20	20	20	20
2	30	20	20	20	18	20
2	45	20	20	20	17	20
2	60	19	20	20	13	20
3	15	20	20	20	20	20
3	30	20	20	20	18	20
3	45	20	20	20	17	20
3	60	20	20	18	12	20

BB is the only algorithm able to find a feasible schedule for all instances. The four heuristics fail very often to compute a solution regarding Model 1, while their feasibility rate is much larger for the other models. This result is motivated by the presence of deadline constraints in Model 1 that are hard constraints for the heuristic approaches presented in this paper. When comparing the four heuristics for Models 2 and 3, we have that AMSP is the best heuristic, while FIFO still frequently fails. In fact, the stand-alone FIFO rule is a too local and myopic strategy to compute a feasible schedule and only the experience of air traffic controllers makes it possible to produce feasible solutions starting from the FIFO rule. For an automated decision support system to be effective in practice, more sophisticated optimization algorithms are required to manage disturbed traffic conditions in a busy TMA.

The next two subsections present a detailed comparison between BB and the heuristics for Model 2 and 3 only, due to the too low feasibility rate of the heuristics regarding Model 1.

5.5 BB versus FIFO

Tables 5 and 6 report on the average results of BB and FIFO for Models 2 and 3, respectively. For both tables, Column 1 shows the time horizon of traffic prediction, Column 2 the scheduling algorithm. Columns 3–5 present the average results obtained for all instances of each time horizon (20 instances per row): the computation time (in seconds) in Column 3, the number of feasible schedules in Column 4 and the number of optimal solutions (proven optimal by BB within 120 seconds) in Column 5. For BB, the computation time refers to the time to compute its best solution (excluding the computation time of the initial heuristics).

In order to obtain a fair comparison of the computational results, the values of the last three columns (Columns 6–8) show the average on the instances for which a solution is found by FIFO. Columns 6 in Tables 5 and 6 shows the maximum consecutive delays of the solutions (in seconds), which is the objective function of the ASP. Since practitioners are also interested in comparing alternative solutions on the basis of different indicators, Columns 7 and 8 show the average consecutive delays and the total increase of travel

Table 5: Model 2: FIFO vs BB for each time horizon

Time Horiz	Sched Algo	Comp Time	Feas Sched	Opt Solut	Max Delay	Avg Delay	Total DTTS
15 min	BB	0.01	20	20	70.9	10.6	169
15 min	FIFO	0.01	20	17	74.4	9.9	151
30 min	BB	0.17	20	20	303.9	91.9	1130
30 min	FIFO	0.02	18	0	584.8	178.3	1431
45 min	BB	0.96	20	20	265.6	78.5	1531
45 min	FIFO	0.07	17	0	752.7	254.4	2181
60 min	BB	3.23	20	0	1068.2	362.4	2673
60 min	FIFO	0.49	13	0	1911.1	872.5	2918

Table 6: Model 3: FIFO vs BB for each time horizon

Time Horiz	Sched Algo	Comp Time	Feas Sched	Opt Solut	Max Delay	Avg Delay	Total DTTS
15 min	BB	0.01	20	20	70.9	10.2	182
15 min	FIFO	0.01	20	16	73.0	7.6	181
30 min	BB	0.29	20	20	303.2	75.1	1730
30 min	FIFO	0.02	18	0	583.7	96.5	4378
45 min	BB	1.18	20	20	264.9	64.9	2294
45 min	FIFO	0.08	17	0	752.7	137.2	8156
60 min	BB	0.98	20	0	1085.1	313.7	5358
60 min	FIFO	0.47	12	0	1918.3	447.7	37306

time spent by all aircraft in the TMA besides their minimum traversing time (Delta Travel Time Spent, or DTTS), both in seconds. DTTS is computed as follows. For an aircraft a landing at a runway r, let t_{ar} and t_{ae} denote the actual landing time at r and the actual entrance time in the TMA, respectively. Let also τ_{ar} and τ_{ae} denote the minimum landing time at r and the minimum entrance time in the TMA, computed disregarding the presence of the other aircraft. Then, the DTTS of aircraft a is $(t_{ar} - t_{ae}) - (\tau_{ar} - \tau_{ae})$. The computation is similar for departing aircraft. The DTTS indicator is interesting for energy consumption reasons, since the smaller is the DTTS of aircraft in the TMA, the smaller is the overall energy consumption. From the computational results of Tables 5 and 6, the DTTS highlights a different behavior of FIFO with Model 2 and Model 3, since with Model 3 the aircraft spend much more time in the holding circles with respect to BB, especially for the largest time horizons.

From the computational results of Tables 5 and 6, we conclude that both algorithms are very fast. However, FIFO often fails in finding a feasible schedule. For the 15-minute time horizon, a small trade-off exists between minimizing the maximum consecutive delay and the other performance indicators. For time horizons of 30, 45 and 60 minutes, BB outperforms FIFO in terms of all performance indicators. These results thus demonstrate the potential benefit of introducing optimization-based DSSs to support the activity of air traffic controllers.

5.6 BB versus the AG heuristics

We next compare the branch and bound algorithm with the arc greedy and job greedy heuristic approaches in terms of feasibility rate, computation time and solution quality. These results are presented in Tables 7 and 8 for Models 2 and 3, respectively. In the first five columns, we show the same kind of information reported in Tables 5 and 6. In order to obtain a fair comparison of the computational results, the values of the last three columns report the average over those instances for which a solution is found by all the three heuristics (i.e., over the instances successfully solved by AMCC for 60-minute traffic predictions in Table 7 and over the instances successfully solved by JGH for 60-minute traffic predictions in Table 8).

From the results of Tables 7 and 8, BB outperforms the other algorithms in terms of maximum consecutive delay minimization and total DTTS. As for the heuristics, AMSP is the most robust in terms of feasible schedules found while AMCC and JGH are the most performing in terms of maximum consecutive delay minimization. In particular, AMCC is the most performing for the smallest time horizons, while JGH is the most performing for the largest time horizon.

We observe that minimizing the maximum consecutive delay (the objective function of BB) has also a positive effect on the other two performance indicators, even if BB does not always produce the best values of the secondary performance indicators. For some instances, DTTS is better reduced by BB while the average consecutive delay is not. This is due to the fact that the average consecutive delay is collected for the departing aircraft only if they are late, i.e. after 10 minutes from their scheduled departure time.

5.7 Study of BB solutions for the three ASP models

Table 9 presents the BB results for all instances of each time horizon (Column 1) and model (Column 2). Columns 3-5 present the maximum consecutive delay at two locations (entry fix and runway) plus their maximum value. Columns 6–8 the average consecutive delay at two locations (entry fix and runway) plus their average value. Column 9 shows the total DTTS (in seconds).

From the information on Table 9, we can analyze the solution quality of the different models in terms of various performance indicators. In general, Model 1 presents the largest consecutive delays and total DTTS, due to the deadline constraints.

Comparing Models 2 and 3, we recall that the time spent in the holding circle is penalized at the entrance of the TMA for Model 2 while it is not penalized for Model 3. As a consequence, the maximum and average consecutive delays at the entrance of the TMA are significantly larger for Model 2. Moreover, in Model 3 the maximum consecutive delay is almost always measured in output (except for the 15 minutes instances), while in Model 2 it may happen that the maximum consecutive delay occurs at the entrance of the TMA (in particular for the 45 minutes instances). In general, Model 2 exhibits a larger average consecutive delay but a smaller DTTS compared to Model 3. This is for the same reason above, i.e., the time spent by an aircraft in the holding circle increases the entrance delay for Model 2 and the DTTS for Model 3.

Table 7: Model 2: AG heuristics vs BB for each time horizon

Time Horiz	Sched Algo	Comp Time	Feas Sched	Opt Solut	Max Delay	Avg Delay	Total DTTS
15 min	BB	0.01	20	20	70.9	10.6	169
15 min	AMCC	0.01	20	20	70.9	10.6	169
15 min	AMSP	0.01	20	18	76.3	13.9	181
15 min	JGH	0.01	20	18	71.5	11.2	179
30 min	BB	0.17	20	20	302.9	92.1	1128
30 min	AMCC	0.01	20	4	364.1	119.6	1241
30 min	AMSP	0.01	20	1	426.1	144.9	1223
30 min	JGH	0.01	20	0	368.7	118.6	1204
45 min	BB	0.96	20	20	257.9	76.1	1537
45 min	AMCC	0.01	20	3	314.8	96.8	1709
45 min	AMSP	0.01	20	1	339.4	102.6	1697
45 min	JGH	0.04	20	1	316.6	104.9	1739
60 min	BB	3.23	20	0	1086.4	396.5	2755
60 min	AMCC	0.07	19	0	1131.1	354.9	2781
60 min	AMSP	0.08	20	0	1130.9	406.0	2871
60 min	JGH	0.19	20	0	1093.6	431.5	2803

Table 8: Model 3: AG heuristics vs BB for each time horizon

Time Horiz	Sched Algo	Comp Time	Feas Sched	Opt Solut	Max Delay	Avg Delay	Total DTTS
15 min	BB	0.01	20	20	70.9	10.2	182
15 min	AMCC	0.01	20	20	70.9	10.2	182
15 min	AMSP	0.01	20	13	72.9	9.6	249
15 min	JGH	0.01	20	13	72.9	9.6	249
30 min	BB	0.29	20	20	302.2	74.9	1745
30 min	AMCC	0.01	20	5	353.7	69.3	3141
30 min	AMSP	0.01	20	3	396.5	74.5	3274
30 min	JGH	0.01	20	2	361.5	65.3	3092
45 min	BB	1.18	20	20	257.2	63.1	2238
45 min	AMCC	0.01	20	2	316.8	62.5	3591
45 min	AMSP	0.01	20	2	349.9	66.5	3832
45 min	JGH	0.03	20	1	314.1	63.6	3913
60 min	BB	0.98	20	0	1105.2	357.4	6252
60 min	AMCC	0.07	20	0	1150.9	252.8	12710
60 min	AMSP	0.07	20	0	1147.4	275.2	14078
60 min	JGH	0.19	18	0	1105.2	275.0	16456

Table 9: Comparison of the three models for each time horizon

Time	Graph	Max Cons Delay			Avg Cons Delay			Total
Horiz	Model	Max	In	Out	Avg	In	Out	DTTS
15 min	1	117.5	-	117.5	51.4	-	51.4	558
15 min	2	70.9	27.5	69.5	10.6	6.2	14.4	169
15 min	3	70.9	21.4	69.5	10.2	4.8	14.9	182
30 min	1	350.8	-	350.8	125.9	-	125.9	3102
30 min	2	302.9	293.1	302.2	92.1	88.3	95.0	1128
30 min	3	302.2	172.4	302.2	74.9	49.6	95.1	1745
45 min	1	562.8	-	562.8	168.4	-	168.4	5290
45 min	2	257.9	231.0	301.1	76.1	63.2	85.7	1537
45 min	3	257.2	137.3	257.2	63.1	32.5	86.3	2238
60 min	1	1180.2	-	1180.2	402.4	-	402.4	10046
60 min	2	1087.3	931	1087.3	398.3	365.6	420.1	2778
60 min	3	1111.7	649.1	1111.7	343.7	227.3	421.4	6085

5.8 Computational aspects of the BB algorithm

This subsection analyzes the optimality gap of BB for different time limits of computation. Table 10 presents the average BB results on all instances of a specific time horizon (Column 1) and model (Column 2). For BB with a time limit of 120 seconds: Columns 3–4 report the average and maximum times to compute the best solution, Column 5 the total computation time, Column 6 the objective function value, and Column 8 the number of proven-optimal solutions. For BB with a time limit of 3600 seconds, Column 7 report the objective function value and Column 9 the number of proven-optimal solutions. An asterisk in the table indicates the all the 20 corresponding instances are solved to optimality within the time limit.

Table 10: Comparison of the three models for each time horizon

Time	Graph	Time Best (BB 120 s)		Avg Time Tot	BB Solution		Opt Solutions	
Horiz	Model	Avg	Max	(BB 120 s)	120 s	3600 s	120 s	3600 s
15 min	1	0.01	0.01	0.01	117.5*	117.5*	20	20
15 min	2	0.01	0.01	0.01	70.9*	70.9*	20	20
15 min	3	0.01	0.01	0.01	70.9*	70.9*	20	20
30 min	1	41.05	95.28	72.5	350.8	338.6*	13	20
30 min	2	0.17	0.66	1.95	302.9*	302.9*	20	20
30 min	3	0.29	1.08	2.69	302.2*	302.2*	20	20
45 min	1	17.23	91.56	114.8	562.8	419.7	2	11
45 min	2	0.96	9.34	39.48	257.9	257.9*	16	20
45 min	3	1.18	10.26	37.59	257.2	257.2*	16	20
60 min	1	12.88	88.41	120	1180.2	1180.2	0	0
60 min	2	3.23	50.83	120	1087.3	1087.3	0	0
60 min	3	0.98	2.42	120	1111.7	1111.7	0	0

From Table 10 it can be observed that, due to the deadline constraints, Model 1 is the most time consuming for the branch and bound algorithm. In fact, for the instances up

to 30 minutes (the time horizon of practical applications), BB proves optimality for all instances within a few seconds for Models 2 and 3, while for Model 1 only 33 instances out of 40 are solved to optimality within 120 seconds and the time to find the best solution may increase up to 95 seconds. The 7 open instances can be solved to optimality by enlarging the computation time of BB up to 3600 seconds. Regarding the 45-minute time horizon instances, an optimal solution is always quickly computed for Models 2 and 3, even if for 4 instances proving optimality requires more than 120 seconds. The best solution for Model 1 is found on average in 17 seconds of computation but optimality can be proved for only 3 instances out of 20 within 3600 seconds of computation. For the 60-minute time horizon, we cannot prove optimality for any instance, even if a feasible schedule is always found within a few seconds and no solution is improved after 89 seconds.

Table 11: Open instances

Perturbation Identifier	Lower Bound	Solut Init	Solut 30sec	Solut 60sec	Solut 120s	Solut Final	Time Best	Time Tot
10-U-231.5-30-1	323	-	491	430	398	398*	67.7	129.6
10-U-247.1-30-1	351	-	537	429	403	403*	91.1	395.2
10-U-263.1-30-1	414	-	528	528	480	480*	95.3	328.9
10-U-267.5-30-1	351	-	537	528	525	427*	233.9	373.4
10-U-229.6-30-1	334	-	590	528	528	383*	344.1	404.1
10-G-207.5-30-1	199	-	289	289	287	287*	89.2	201.7
10-G-224.3-30-1	211	-	464	376	307	307*	83.4	202.0
14-U-358.4-45-1	238	-	836	836	836	791	2257.0	3600.0
14-U-369.6-45-1	255	-	610	610	610	382	1603.8	3600.0
14-U-384.3-45-1	278	-	655	655	655	340*	1079.7	2406.0
14-U-384.5-45-1	262	-	725	725	725	601	3002.0	3600.0
14-U-370.4-45-1	251	-	847	847	847	740	3384.0	3600.0
14-U-352.0-45-1	255	-	-	742	742	480	3490.0	3600.0
14-U-347.7-45-1	342	-	709	709	709	709	0.9	3600.0
14-U-382.6-45-1	275	-	705	705	705	660	2838.0	3600.0
14-U-349.5-45-1	207	-	438	284	284	284*	32.9	146.3
14-U-347.6-45-1	266	-	800	800	800	670	1639.6	3600.0
14-G-333.0-45-1	134	-	534	534	534	253*	180.1	1594.5
14-G-326.4-45-1	105	-	537	537	537	537	0.8	3600.0
14-G-330.5-45-1	117	-	392	392	392	275*	1394.7	2424.0
14-G-325.9-45-1	147	-	322	322	322	240*	581.4	637.2
14-G-339.4-45-1	225	-	597	597	597	257*	436.4	536.2
14-G-328.2-45-1	185	-	563	563	563	262*	641.1	3182.0
14-G-311.9-45-1	190	-	451	451	451	243*	824.7	845.1
14-G-340.4-45-1	186	-	-	537	537	261*	3348.0	3375.0
14-U-384.3-45-2	278	341	325	325	325	325*	0.4	255.5
14-U-384.5-45-2	262	356	328	328	328	328*	0.8	444.6
14-U-347.7-45-2	342	459	389	389	389	389*	1.4	3402.0
14-U-382.6-45-2	275	380	345	345	345	345*	0.6	647.5
14-U-384.3-45-3	278	341	325	325	325	325*	0.4	257.9
14-U-384.5-45-3	262	356	328	328	328	328*	0.6	459.9
14-U-347.7-45-3	342	459	389	389	389	389*	1.5	3441.0
14-U-382.6-45-3	275	380	345	345	345	345*	0.6	606.8

We next focus on the 33 instances that remained open at 120 seconds for the time horizon of 30 and 45 minutes. Table 11 reports the performance of BB on these instances. Each instance is coded in Column 1 with the following *A-B-C-D-E* fields: Under *A* we denote the number of delayed aircraft at the entrance of the TMA; *B* indicates the distribution type: Uniform (U) or Gaussian (G); *C* represents the average entrance delay in seconds; *D* indicates the time horizon of traffic prediction, from 15 to 60 minutes; *E* represents the model used for the ASP, where 1–3 are Models 1–3. Column 2 shows the final lower bound computed by BB, while Column 3 the best solution found by AMCC, AMSP, JGH and FIFO (if any). Column 4–7 present the solution found by BB after 30, 60, 120 and 3600 seconds of computation. Each asterisk in Column 7 indicates the proven optimality of the corresponding solution. Columns 8–9 report the computation time (in seconds) to reach the best solution and to terminate the algorithmic search.

From the results of Table 11, it turns out that for all tested instances BB improves the initial solution found by the heuristics. For the 7 open instances with time horizon equal to 30 minutes, BB is able to close all instances within around 400 seconds of computation, in 5 cases the optimal solution is the same solution found at 120 seconds. Regarding the 26 instances with time horizon of 45 minutes, BB is able to close 17 instances within the largest time limit and to improve the solutions obtained at 120 seconds for 7 of the other 9 instances. For 9 out of 17 instances solved to optimality, the optimal solution is the same found at 60 seconds.

The analysis of Tables 10 and 11 confirms that Models 2 and 3 are easier to solve than Model 1. In fact, all the instances associated to the former models are solved to optimality. Regarding the 45-minute time horizon instances, an optimal solution is always quickly computed for Models 2 and 3, while a feasible schedule is always computed within 1 minute for Model 1.

As a concluding remark on our computational results, we observe that our branch and bound algorithm is quite effective for time horizons of practical interest. For time horizons larger than 45 minutes for Models 2 and 3 and larger than 30 minutes for Model 1, the BB algorithm is still effective in finding feasible solutions but their quality cannot be proved.

6 Conclusions and future work

This paper presents heuristic and exact algorithms for computing feasible schedules for three alternative graph formulations of the ASP. The proposed models generalize previous work on job shop scheduling application to solve the ASP. Several solution algorithms are proposed, ranging from simple heuristics to exact algorithms. Test cases for the FCO airport demonstrate that the branch and bound algorithm is able to solve at optimality instances of practical size within a time limit compatible with real-time application. Also when taking into account other performance indicators, BB often outperforms the heuristics.

Test experiments show that the support provided by our optimization models and algorithms has a significant potential for improving the performance of the TMA as well as to reduce the workload of air traffic controllers. Our approach could be used both in real-time and off-line, to assess the quality of alternative plans. However, since off-line analysis frequently focus on larger time horizons than real-time control, further research

efforts are worth to develop good lower bounds or advanced meta-heuristic algorithms.

Ongoing research is dedicated to the development of on-line decision support systems for air traffic control at TMAs. The scheduling models and algorithms presented in this work should be part of the system core. Other research directions may be considered: the design of an integrated approach to deal with a dynamic setting of the decision support system, the inclusion of rerouting decisions in the TMA to dynamically balance the load of each runway, the extension of our methodology to better deal with speed variations in the landing procedure, the development of combined aircraft scheduling and routing algorithms to solve the ASP, the study of closed-loop control measures for handling disrupted traffic situations [11].

Acknowledgments

This work is partially supported by the Italian Ministry of Research, Grant number RBIP06BZW8, project FIRB "Advanced tracking system in intermodal freight transportation". Preliminary results on a subset of the ASP instances have been presented at the 13th International IEEE Conference on Intelligent Transportation Systems in September 2010 [13].

References

[1] Airports Council International (ACI, 2005). Worldwide and Regional Forecasts, Airport Traffic 2005-2020.

[2] Adacher, L., Pacciarelli, D., Paluzzi, D., Pranzo, M. (2004) Scheduling arrivals and departures in a busy airport. Preprints of the 5th Triennial Symposium on Transportation Analysis, Le Gosier, Guadaloupe, French West Indies, pp. 1–4.

[3] Allahverdi, A., Ng, C.T., Cheng, T.C.E., Kovalyov, M.Y. (2008) A survey of scheduling problems with setup times or costs. *European Journal of Operational Research* **187** 985–1032.

[4] Artiouchine, K., Baptiste, Ph., Durr, C. (2008) Runway Sequencing with Holding Patterns. *European Journal of Operational Research* **189(3)**, 1254–1266.

[5] Artiouchine, K., Baptiste, Ph., Mattioli, J. (2008) The K King Problem, an Abstract Model for Computing Aircraft Landing Trajectories: On Modeling a Dynamic Hybrid System with Constraints. *INFORMS Journal on Computing* **20(2)**, 222–233.

[6] Ball, M., Barnhart, C., Nemhauser, G., Odoni, A. (2007) Air Transportation: Irregular Operations and Control. In: G. Laporte and C. Barnhart (Eds.), *Handbooks in Operations Research and Management Science* **14** 1-67.

[7] Beasley, J.E., Krishnamoorthy, M., Sharaiha, Y.M., Abramson, D. (2000) Scheduling aircraft landings – The static case. *Transportation Science* **34 (2)** 180–197.

[8] Beasley, J.E., Krishnamoorthy, M., Sharaiha, Y.M., Abramson, D. (2004) Displacement problem and dynamically scheduling aircraft landings. *Journal of Operational Research Society* **55** 54–64.

[9] Bennell, J.A., Mesgarpour, M., Potts, C.N. (2011) Airport runway scheduling. *4OR - Quarterly Journal of Operations Research* **9(2)**, 115–138.

[10] Bianco, L., Dell'Olmo, P., Giordani, S. (2006) Scheduling models for air traffic control in terminal areas. *Journal of Scheduling* **9 (3)** 180–197.

[11] Clausen, J. (2007) Disruption Management in Passenger Transportation - from Air to Tracks. *Proceedings of the 7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS 2007).* In C. Liebchen, R.K. Ahuja (Eds), Sevilla, Spain.

[12] D'Ariano, A., Pacciarelli, D., Pranzo, M. (2007) A branch and bound algorithm for scheduling trains in a railway network. *European Journal of Operational Research* **183 (2)** 643-657.

[13] D'Ariano, A., D'Urgolo, P., Pacciarelli, D., Pranzo, M. (2010). Optimal Sequencing of Aircrafts Take-Off and Landing at a Busy Airport. *Proceedings of the 13th International IEEE Annual Conference on Intelligent Transportation Systems*, September 19-22, 2010, Madeira Island, Portugal, pp. 1569–1574.

[14] Dear, R.G. (1976) The dynamic scheduling of aircraft in the near terminal area. Report R76-9, Flight Transportation Laboratory, MIT, Cambridge, MA, USA.

[15] Ernst, A.T., Krishnamoorthy, M., Storer, R.H. (1999) Heuristic and exact algorithms for scheduling aircraft landings. *Networks* **34 (3)** 229–241.

[16] Eun, Y., Hwang, I., Bang, H. (2010) Optimal Arrival Flight Sequencing and Scheduling Using Discrete Airborne Delays. *IEEE Trans. on Intelligent Transportation Systems* **11 (2)** 359–373.

[17] Hall, N.G., Sriskandarajah, C. (1996) A Survey of Machine Scheduling Problems with Blocking and No-Wait in Process. *Operations Research* **44 (3)** 510–525.

[18] Hansen, V.J. (2004) Genetic search methods in air traffic control. *Computers and Operations Research* **31 (3)** 445–459.

[19] Jackson, J.R. (1955) Scheduling a production line to minimize maximum tardiness. Report **43**, Management Science Research Project, University of California, Los Angeles, USA.

[20] Kim, J., Kröller, A., Mitchell, J.S.B. and Sabhnani, G.R. (2009) Scheduling Aircraft to Reduce Controller Workload. *Proceedings of the 9th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS 2009).* In J. Clausen, G. Di Stefano (Eds), Copenhagen, Denmark.

[21] Kuchar, J.K., Yang, L.C. (2000) A Review of Conflict Detection and Resolution Modeling Methods. *IEEE Trans. on Intelligent Transportation Systems* **4 (1)** 179–189.

[22] Mascis, A., Pacciarelli, D. (2002) Job shop scheduling with blocking and no-wait constraints. *European Journal of Operational Research* **143 (3)** 498–517.

[23] Pistelli, M., D'Ariano, A., Pacciarelli, D. (2011) Optimization models and algorithms for air traffic control in the airspace of busy airports. *Tech. Rep. RT-DIA-185-11* pp. 1–26, Dipartimento di Informatica e Automazione, Università degli Studi Roma Tre.

[24] Pranzo, M., Meloni, C., Pacciarelli, D. (2003) A new class of greedy heuristics for job shop scheduling problems. *Lecture Notes in Computer Science* **2647** 223–236.

[25] Psaraftis, H.N. (1980) A dynamic programming approach for sequencing identical groups of jobs. *Operations Research* **28 (6)** 1347–1359.

[26] Soomer, M.J., Franx, G.J. (2008) Scheduling aircraft landings using airlines' preferences. *European Journal of Operational Research* **190 (1)** 277–291.

[27] Sölveling, G., Solak, S., Clarke, J.B., Johnson, E.L. (2010) Scheduling of runway operations for reduced environmental impact. *Transportation Research - Part D* **16 (2)** 110–120.

[28] Venkatakrishnan, C.S., Barnett, A., Odoni, A.M. (1993) Landings at Logan airport: describing and increasing airport capacity. *Transportation Science* **27 (3)** 211–227.

[29] Zhan, Z-H., Zhang, J., Li, Y., Liu O., Kwok, S.K., Ip, W.H., Kaynak, O. (2010) An efficient Ant Colony System based on receding horizon control for the Aircraft Arrival Sequencing and Scheduling Problem. *IEEE Trans. on Intelligent Transportation Systems* **11 (2)** 399–412.