Metaheuristics for efficient aircraft scheduling and re-routing at busy terminal control areas

Marcella Samà [1], Andrea D'Ariano [1*], Francesco Corman[2], Dario Pacciarelli [1]

June 30, 2016

[1] Roma Tre University, Department of Engineering, Section of Computer Science and Automation, via della Vasca Navale, 79 – 00146 Rome, Italy.

* Corresponding author email: dariano@ing.uniroma3.it

[2] Delft University of Technology, Department of Maritime and Transport Technology, Section of Transport Engineering and Logistics, Mekelweg, 2 – 2628CD Delft, The Netherlands.

Abstract

Intelligent decision support systems for the real-time management of landing and take-off operations can be very effective in helping air traffic controllers to limit airport congestion at busy terminal control areas. The key optimization problem to be solved regards the assignment of airport resources to take-off and landing aircraft and the aircraft sequencing on them. The problem can be formulated as a mixed integer linear program. However, since this problem is strongly NP-hard, heuristic algorithms are typically adopted in practice to compute good quality solutions in a short computation time. This paper presents a number of algorithmic improvements implemented in the AGLIBRARY solver (a state-of-the-art optimization solver to deal with complex routing and scheduling problems) in order to improve the possibility of finding good quality solutions quickly. The proposed framework starts from a good initial solution for the aircraft scheduling problem with fixed routes (given the resources to be traversed by each aircraft), computed via a truncated branch-and-bound algorithm. A metaheuristic is then applied to improve the solution by re-routing some aircraft in the terminal control area. New metaheuristics, based on variable neighbourhood search, tabu search and hybrid schemes, are introduced. Computational experiments are performed on an Italian terminal control area under various types of disturbances, including multiple aircraft delays and a temporarily disrupted runway. The metaheuristics achieve solutions of remarkable quality, within a small computation time, compared with a commercial solver and with the previous versions of AGLIBRARY.

Keywords: Optimal Air Traffic Control; Landing and Take-Off Operations; Disruption Management; Disjunctive Programming; Variable Neighbourhood Search; Tabu Search; Hybrid Algorithms.

Acknowledgements: *We acknowledge support from Ing. G. Zaninotto and Ing. A. Toli. We also would like to thank the editors and anonymous reviewers for their helpful, accurate and constructive remarks. All the tested ATC-TCA instances are available upon request by sending an email to the corresponding author of this paper.*

1 Introduction

1.1 The investigated problem

The ever increasing level of air traffic flows poses a difficult challenge for air traffic controllers, that work to ensure the safety and efficiency of operational schedules. This is a difficult task, particularly in bottleneck areas, and the SESAR project and CDM compliance [14, 40] are pushing for the implementation and use of automated traffic control systems. One typical bottleneck of the entire air traffic system is the Terminal Control Area (TCA). During operations, aircraft delays are considered to cause a substantial cost from both airlines and passengers' points of view. The computation of optimal aircraft landing and take-off schedules is thus one of the most relevant operational problems. These facts stimulated the interest for effective intelligent transport system solutions that can show how to better use the existing resources [44].

This paper aims to improve substantially the quality of solutions, in terms of value of the objective function and computation time, for the Air Traffic Control in a Terminal Control Area (ATC-TCA) problem. The investigated problem consists of simultaneously determining the routing (i.e. the resources to be traversed), sequencing (i.e. the orders between aircraft in each resource) and timing of landing and take-off aircraft on the TCA resources, which may include several runways and air segments. While there is no generally recognized objective function in the literature [39], this paper considers a relevant objective function that aims to minimize the maximum positive deviation from the target landing and take-off times.

The resolution of air traffic control problems requires to consider several factors related to safety, efficiency and equity. In this work, safety requires the careful modelling of practical TCA constraints, efficiency consists of reducing aircraft delays with global conflict detection and resolution approaches, equity is achieved by minimizing the largest delay due to the resolution of conflicts. These factors require to consider the routing, sequencing and timing of all aircraft moving in the network during the studied traffic horizon. Further relevant safety, efficiency, equity, and even environmental impacts are addressed, e.g., in [41, 42, 43].

1.2 The related literature

In the aircraft scheduling literature, it is often mentioned a big gap existing between the level of sophistication of published results and algorithms, and the simple methods that are employed in practice. One motivation reported for this gap is that the theory typically addresses very simplified problems for which (near-)optimal performance can be achieved, while the practice must face all the complexity of real-time operations. However, poorly performing aircraft scheduling and routing methods that are used in practice directly impact the quality of service offered to the passengers, the effect being more evident as traffic density gets close to saturation. In fact, any small disturbance related to few aircraft may propagate to the other aircraft, altering the regularity of air traffic even some hours after the end of the original disturbance.

There is a recent trend of research to incorporate more practical objectives and constraints in the detailed (microscopic) models that have not been adequately captured in published models, since too simplified (macroscopic) models may have a limited impact on the practice of air traffic control. In view of the extensive reviews reported in [3, 5, 7, 24, 25, 31], we limit our review of the recent related literature to two streams of research: (i) the development of microscopic models for the management of aircraft flows in

terminal control areas, (ii) the development of algorithmic methods for solving the models of stream (i).

Regarding stream (i), there are a few microscopic models that incorporate the detailed characteristics of the airport infrastructure and of the individual flight paths. Such a level of detail is required to safely detect and solve potential conflicts at the level of runways, ground and air segments of the TCA. The most detailed model used in this context is the job shop scheduling model in which each *operation* denotes the traversal of an air/ground segment, runway (*resource*) by an aircraft (*job*). The variables are the start time of each operation to be performed by an aircraft on a specific resource. A *no-wait* version of this model has been firstly proposed in [8] and successively extended in [11, 12, 26, 38] as a *blocking and no-wait* version. In the latter approach, air segment resources are treated as no-wait resources with time windows for modelling minimum/maximum aircraft travel times, while runway resources are treated as blocking resources which can host at most one aircraft at a time. Objective functions are based on a makespan minimization.

Regarding stream (ii), exact and heuristic algorithms have been proposed for the ATC-TCA problem. Among the literature on exact algorithms, Psaraftis [33] and Balakrishnan and Chandran [4] propose a combination of the constrained position shifting approach (originally introduced by Dear [13]) and of the dynamic programming approach to solve aircraft sequencing and runway scheduling problems, D'Ariano et al. [11] describe a branch and bound algorithm for the aircraft scheduling problem with fixed routes, Faye [15] present a dynamic constraint generation algorithm for an aircraft landing problem. However, exact algorithms can quickly compute near-optimal solutions only for quite small instances or simplified problems. Consequently, numerous metaheuristics have been recently proposed to search for good quality solutions in a short computation time, the most used being the following: genetic algorithms [6, 20, 21, 22], scatter search [32], tabu search [2, 12, 16, 38], ant colony [23, 45], simulated annealing [18, 34], variable neighbourhood search [1, 34, 35]. Several of the proposed algorithms have also been hybridized in order to combine interesting properties and to take the best from each of them. Furthermore, some approaches (e.g., [16, 20, 21, 22, 28, 38, 45]) have been implemented in a rolling horizon (or receding horizon) control framework in order to solve large instances in a short computation time compatible with real-time applications, and to deal with the dynamic and uncertain nature of the ATC-TCA problem. All these approaches have proposed significant improvements compared to the commonly used air traffic control rules, such as the first-in-first-out rule. In fact, the usual control rules take a few sequencing and routing decisions at a time in a myopic fashion, ignoring the propagation of delays to other aircraft in the network [21, 29]. The proposed neighbourhood research methods are well focused on the minimization of the propagation of aircraft delays or other relevant factors. However, the majority of the works focuses on the development of good neighbourhood search capabilities for aircraft sequencing problems and simplified networks, while there is still a lack of fast and effective algorithmic contributions on the simultaneous aircraft scheduling and routing problem on the TCA resources. The latter problem is the main subject of this paper.

In view of the above discussion of the recent literature regarding the management of landing and take-off operations, there is a clear need to incorporate an increasing level of detail and realism in the models while keeping the computation time and quality of the algorithms at an acceptable level. Furthermore, the ATC-TCA problem is well known to be NP-hard, requiring the implementation of advanced heuristics, especially when solving complex instances with multiple delayed aircraft and severe resource capacity deficiencies.

This paper deals with the real-world instances of Samà et al. [38], with up to more than 200,000 scheduling variables and more than 400 routing variables. Since it is not possible to solve these instances with an exact method in a reasonable amount of time, we focus on the development of hybrid metaheuristics (based on tabu search and variable neighbourhood search schemes) to derive good quality solutions in a short time.

1.3 The paper contribution

A recent stream of research on detailed ATC-TCA problem formulations focuses on the Alternative Graph (AG) of Mascis and Pacciarelli [26]. The alternative graph has been first successfully applied to manage other transportation and production problems [9, 10, 30]. In this paper, the ATC-TCA problem is modelled as a generalized job shop scheduling problem via alternative graphs, enriching the model of [8] by addressing the real-world constraints and the objective function proposed in [11, 12, 37, 38]. This graph allows a more accurate modelling of relevant TCA aspects and safety constraints, such as holding circles, waiting in flight before landing, travelling in feasible time windows, hosting multiple aircraft simultaneously in air segments and individual aircraft in runways. Specifically, the alternative graph can model any 4-dimensional route for the aircraft in the TCA, while most of the related work done assumes 3-D routes are fixed and only optimizes the timing of runway operations. In order to include the routing flexibility in the AG model, we use the Mixed Integer Linear Programming (MILP) formulation of [38]. The MILP formulation can be efficiently solved by the rolling horizon framework of [37]. However, the latter approach requires a large computation time when dealing with complex ATC-TCA instances.

This paper presents a number of algorithmic improvements implemented in the AGLIBRARY solver, a set of optimization models and algorithms for complex routing and scheduling problems developed at Roma Tre University. The solver is based on the following framework: a good initial solution for the scheduling problem with fixed routes is computed by the (truncated) branch-and-bound algorithm of [11]. Metaheuristics are then applied to improve the solution by re-routing some aircraft. This action corresponds to the concept of a move, from a metaheuristics perspective. In [12], a tabu search algorithm has been applied to solve practical-size instances for small disturbances. Previous research left open the following two relevant issues. The first issue concerns the extent at which different solution methods might outperform the tabu search algorithm and the rolling horizon framework. A second issue is to study algorithmic improvements, in order to reduce the time to compute good quality solutions. Both these issues motivate the development of the new metaheuristics proposed in this paper. The paper contributions are next outlined:

- We present new routing neighbourhoods that differ from each other in terms of the aircraft that are re-routed in each move and for the set of candidate aircraft routes;
- We alternate the search for promising moves in neighbourhoods of different size, similarly to [27], adopt fast re-scheduling heuristics for the neighbour evaluation, and present strategies for searching within these neighbourhoods based on variable neighbourhood search, tabu search and hybrid schemes;
- We apply the proposed algorithms to the management of complex disturbed situations, including multiple delayed landing and/or take-off aircraft and a temporarily disrupted runway. The situations tested are the most complex instances in [38]. The new metaheuristics are compared with the other

existing methods based on the AG model, and with solutions computed with a commercial MILP solver. Significantly better results are obtained in terms of an improved solution quality and/or a reduced computation time with respect to both the MILP solver and the previous versions of AGLIBRARY.

Section 2 formally describes the ATC-TCA problem and the MILP formulation. Section 3 presents the metaheuristic algorithms proposed in this paper. Section 4 reports the performance of the algorithms on the Milan Malpensa terminal control area (MXP) instances of Samà et al. [38]. Section 5 summarizes the paper results and outlines future research directions.

2 Problem definition and formulation

In this section, a description of the ATC-TCA problem is provided to the reader, together with the formalization of the alternative graph used to model the problem when a pre-defined route is fixed for each aircraft and an MILP formulation describing its extension when the routing alternatives are also considered.

2.1 The ATC-TCA problem

Landing aircraft move in the landing air segments of the TCA, following a standard descent profile, from an air entry point to a common glide path, that is the final landing air segment before the runway. Take-off aircraft move in the ground resources until they get access to the runway and finally fly toward their assigned exit point via take-off air segments.

A minimum longitudinal and diagonal safety separation distance between every pair of consecutive aircraft must be always respected, depending on their type, altitude and relative positions. This minimum distance can be translated into a *minimum separation time* that is sequence-dependent, since it depends on the aircraft sequencing of the common resources by considering the different aircraft categories, the required wake vortex separation based upon wake turbulence categories and the temporal spacing separation standards.

Each aircraft has a *processing time* on each TCA resource, according to its landing/take-off profile. On the air segments, the processing time varies between minimum and maximum feasible values.

Each landing/take-off aircraft has a minimum entrance time into the TCA, *release time*, according to its current position and speed. Landing aircraft can also be constrained to have a maximum entrance time, *deadline time*, into the TCA, e.g., due to limited fuel availability.

All aircraft have scheduled times, *due date times*, to start processing some TCA resources. 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. Following the procedure commonly adopted by air traffic controllers, we consider a time window for take-off between 5 minutes before and 10 minutes after the *Scheduled Take-off Time* (STT). A departing aircraft is considered delayed in exiting the TCA if leaving the runway after 10 minutes from its STT. Arriving aircraft are late if landing after their *Scheduled Landing Time* (SLT).

Before entering the TCA, landing aircraft can fly in *holding circles* that are air segments dedicated to accumulating aircraft delays during the flight. In each holding circle, landing aircraft must fly at a fixed

speed for a number of half circles, as prescribed by the air traffic controller. Departing aircraft instead can be delayed in entering the TCA at ground level, i.e. before entering the runway.

We use the following notation for aircraft delays. *Entrance delay* (*exit delay*) is the delay of an aircraft on the entrance to (the exit from) the TCA. The exit value is partly a consequence of a possible late entrance, which causes an *unavoidable delay*, and partly due to additional delays caused by the resolution of potential aircraft conflicts in the TCA, which is the *consecutive delay*. In this paper, we minimize the maximum consecutive delay that is an equitable approach for the minimization of aircraft delay propagation. The problem variables are the timing, ordering and routing of each aircraft in the TCA resources.

The next section will show a model of ATC-TCA problem in which a route is assigned to each aircraft. This assumption will then be relaxed in order to deliver a general optimization model.

2.2 The AG model

This subsection presents the alternative graph for the ATC-TCA problem with pre-defined routes. This particular graph is a triple $G(N, F, A)$: $N = \{s, 1, ..., n, t\}$ is the set of *nodes*, where nodes are associated to the following events: s and t represent the start and the end of the schedule, while the other n nodes are related to the start of the n ATC-TCA operations; F is the set of *fixed directed arcs* that model the sequence of operations regarding the pre-defined route of each aircraft; A is the set of *alternative pairs* that model the aircraft sequencing and holding circle decisions. Each pair is composed of two alternative directed arcs.

Each node, except s and t, is associated with an operation labelled with the triple krj, where k indicates the aircraft, r the route chosen and j the resource it traverses. The start time h_{krj} of operation krj is the entrance time of aircraft k in resource j when using route r. Each fixed directed arc $(krp, krj) \in F$ connects the two nodes (operations) krp and krj, and has associated the arc weight $w^F_{krp\_krj}$. With our notation, $w^F_{krp\_krj}$ represents a minimum time constraint between h_{krp} and h_{krj} (i.e. $h_{krj} - h_{krp} \geq w^F_{krp\_krj}$). If krj follows krp on the route r of aircraft k, the fixed arc (krp, krj) $[(krj, krp)]$ has a weight $w^F_{krp\_krj}$ $[w^F_{krj\_krp}]$ equal to the minimum [− maximum] time required by aircraft k to process resource r. In this way, the air segment, runway, and holding circle constraints can be modelled. A fixed direct arc (s, krp), (krp, s) or (krp, t) models a release, deadline or due date constraint regarding operation krp. A detailed description of the various sets of fixed directed arcs is provided, e.g., in [11, 38].

Each alternative pair $((kro, uim), (uig, krl)) \in A$ models an aircraft holding circle (when aircraft k = aircraft u and route r = route i) or sequencing (when aircraft $k \neq$ aircraft u) decision. The two arcs of the pair have associated the weights $w^A_{kro\_uim}$ and $w^A_{uig\_krl}$. In any solution, only one arc of each pair in the set A can be selected. If alternative arc (kro, uim) $[(uig, krl)]$ is selected in a solution, the constraint $h_{uim} - h_{kro} \geq w^A_{kro\_uim}$ $[h_{krl} - h_{uig} \geq w^A_{uig\_krl}]$ has to be satisfied. The set A is composed of the subsets: A_{HC} for holding circle decisions, A_{AS} and A_{RW} for sequencing decisions at air segments and runways. For taking a decision on the holding circles to be performed by a landing aircraft, an alternative arc is selected in A_{HC}. For taking an entrance/exit sequencing decision between two aircraft on a shared air segment, an alternative arc is selected in A_{AS}. For taking a sequencing decision between two aircraft on a shared runway, an alternative arc is selected in A_{RW}. With our notation, the weight of an alternative arc in A_{HC} represents a timing in the holding circle, while the weight of an alternative arc in A_{AS} or A_{RW} represents a minimum

separation time between two ordered aircraft on a shared air segment or runway. A detailed description of A_{HC}, A_{AS} and A_{RW} can be found in [38].

A *selection* S is a set of alternative arcs obtained by selecting exactly one arc from each alternative pair in A and such that the resulting graph $\mathcal{G}(F,S) : (N, F \cup S)$ does not contain positive-weight cycles. A selection S is a solution for the ATC-TCA problem with pre-defined routes. Orders and times for all operations are easily identified given a selection S. The minimization of the maximum consecutive delay is measured as a makespan minimization. Given a selection S and any two nodes krp and uml, we let $l^S(krp, uml)$ be the weight of the longest path from krp to uml in $\mathcal{G}(F,S)$. By definition, the start time h_{krp} of $krp \in N$ is the quantity $l^S(s, krp)$, which implies $h_s = 0$ and $h_t = l^S(s,t)$.

2.3 The MILP formulation

The ATC-TCA problem with flexible routes is formulated as a particular *disjunctive program* [38]. This is achieved via an MILP formulation in which the scheduling and routing decisions are considered simultaneously. The starting point is the alternative graph model for the ATC-TCA problem with pre-defined routes. The graph is formulated via a *big-M* formulation enlarging the sets F and A in order to include the fixed and alternative arcs related to all possible aircraft routes. The advantage of this *big-M* formulation is the exact correspondence between arcs and constraints: each fixed directed arc translates into a fixed constraint, while each alternative pair into a pair of alternative constraints. However, we observe that computing the optimal solution of *big-M* formulations can be a time-consuming task for any solver.

We next give a compact *big-M* formulation, while a detailed formulation is given in Samà et al. [38]. For each operation krp there is a non-negative real variable h_{krp} modelling its start time. Regarding operations s and t, h_s is the given start time of traffic prediction (this can be set equal to 0), while h_t is a non-negative real variable indicating the value of the objective function. For each alternative pair $((krp, dij), (uml, vnw)) \in A$ there is a binary variable $x^{uml,vnw}_{krp,dij}$ modelling the sequencing/holding decision. For each aircraft k and each route r, there is a binary variable y_{kr} modelling the route selection. We observe that the number of binary variables increases quadratically with the number of aircraft, while the number of air segments and routing alternatives for each aircraft is usually quite limited in a terminal control area.

$$\min \; h_t \tag{1}$$

$$\sum_{r=1}^{R_k} y_{kr} = 1 \qquad k = 1, ..., Z \tag{2}$$

$$h_{krj} - h_{krp} + M(1 - y_{kr}) \geq w^F_{krp\_krj} \qquad \forall (krp, krj) \in F \tag{3}$$

$$h_{krp} - h_{krj} + M(1 - y_{kr}) \geq w^F_{krj\_krp} \qquad \forall (krj, krp) \in F \tag{4}$$

$$h_{krp} - h_s + M(1 - y_{kr}) \geq w^F_{s\_krp} \qquad \forall (s, krp) \in F \tag{5}$$

$$h_s - h_{krp} + M(1 - y_{kr}) \geq w^F_{krp\_s} \qquad \forall (krp, s) \in F \tag{6}$$

$$h_t - h_{krj} + M(1 - y_{kr}) \geq w^F_{krj\_t} \qquad \forall (krj, t) \in F \tag{7}$$

$$\begin{aligned} &h_{uim} - h_{kro} + M(2 + x^{uig\_krl}_{kro\_uim} - y_{kr} - y_{ui}) \geq w^{A}_{kro\_uim} \\ &h_{krl} - h_{uig} + M(3 - x^{uig\_krl}_{kro\_uim} - y_{kr} - y_{ui}) \geq w^{A}_{uig\_krl} \qquad \forall((kro, uim), (uig, krl)) \in A \setminus A_{HC} \end{aligned} \tag{8}$$

$$\begin{aligned} &h_{krj} - h_{krp} + M(1 + x^{krj\_krp}_{krp\_krj} - y_{kr}) \geq w^{A}_{krp\_krj} \\ &h_{krp} - h_{krj} + M(2 - x^{krj\_krp}_{krp\_krj} - y_{kr}) \geq w^{A}_{krj\_krp} \qquad \forall((krp, krj), (krj, krp)) \in A_{HC} \end{aligned} \tag{9}$$

$$h_{krp} \geq 0 \qquad \forall krp \in N \tag{10}$$

$$x^{uig\_krl}_{kro\_uim} \in \{0, 1\} \qquad \forall((kro, uim), (uig, krl)) \in A \setminus A_{HC} \tag{11}$$

$$x^{krj\_krp}_{krp\_krj} \in \{0, 1\} \qquad \forall((krp, krj), (krj, krp)) \in A_{HC} \tag{12}$$

$$y_{kr} \in \{0, 1\} \qquad k = 1, ..., Z \;\; ; \;\; r = 1, ..., R_k \tag{13}$$

The objective function is reported in Equation (1). We next describe the ATC-TCA problem constraints.

Constraints (2) model the routing decision for each aircraft k among its set of R_k routes. The route r is chosen for aircraft k if and only if $y_{kr} = 1$. In total, there are Z aircraft.

Constraints (3) and (4) model the fixed directed arcs (krp, krj) and $(krj, krp) \in F$, that represent respectively the minimum and $-$ maximum processing times related to operation krp. An arc (krp, krj) is active (i.e. enforces $h_{krj} - h_{krp} \geq w^{F}_{krp\_krj}$) when the route r is chosen for aircraft k (i.e. $y_{kr} = 1$).

Constraints (5), (6) and (7) model the fixed directed arcs (s, krp), (krp, s) and $(krp, t) \in F$, that represent respectively the release, deadline and due date constraints related to operation krp.

Constraints (8) model the alternative pairs $((kro, uim), (uig, krl)) \in A \setminus A_{HC}$. Each of these alternative pairs model the two possible sequencing decisions between a pair of aircraft at a shared air segment (if the alternative pair belongs to A_{AS}) or at a shared runway (if the alternative pair belongs to A_{RW}). Indeed, the arcs of each alternative pair in $A \setminus A_{HC}$ connect two operations of different jobs (aircraft). An alternative pair $((kro, uim), (uig, krl)) \in A \setminus A_{HC}$ is active in this MILP formulation when both the following conditions hold: (i) the route r is chosen for aircraft k (i.e. $y_{kr} = 1$) and (ii) the route i is chosen for aircraft u (i.e. $y_{ui} = 1$). When an alternative pair $((kro, uim), (uig, krl)) \in A \setminus A_{HC}$ is active, only one of its two arcs must be active in any ATC-TCA solution, enforcing a particular sequencing decision between k and u on a shared resource: the alternative arc (kro, uim) is active (i.e. enforces $h_{uim} - h_{kro} \geq w^{A}_{kro\_uim}$) when $x^{uig\_krl}_{kro\_uim} = 0$, while the alternative arc (uig, krl) is active (i.e. enforces $h_{krl} - h_{uig} \geq w^{A}_{uig\_kro}$) when $x^{uig\_krl}_{kro\_uim} = 1$.

Constraints (9) model the alternative pairs $((krp, krj), (krj, krp)) \in A_{HC}$. These alternative pairs model holding circle decisions regarding a particular aircraft. Differently than in the previous case, this implies that the arcs of each pair connect two operations of the same job. An alternative pair $((krp, krj), (krj, krp)) \in A_{HC}$ is active when the route r is chosen for aircraft k (i.e. $y_{kr} = 1$). When an alternative pair in A_{HC} is active, only one of its two arcs must be active in any ATC-TCA solution. The activation of one arc for each alternative pair is modeled as for Constraints (8). Selecting which alternative arc is active in each alternative pair in A_{HC} corresponds to fixing the number of holding circles to be performed by each landing aircraft.

Constraints (10) set the timing variables h as non-negative real variables, while Constraints (11), (12) and (13) set the sequencing variables x and the routing variables y as binary variables.

3 Scheduling and re-routing algorithms

This section describes the algorithmic approaches proposed in this paper to compute effective solutions for the ATC-TCA problem in a short computation time. Section 3.1 presents the general framework of the solver, that is based on a combination of aircraft scheduling and re-routing algorithms. Section 3.2 illustrates the aircraft routing neighbourhoods, which are a main component of the proposed heuristic search procedures. Section 3.3 details the scheduling heuristic procedure used to evaluate the neighbours (the new routing combinations). The routing neighbourhoods and the scheduling algorithms are used in Sections 3.4, 3.5, 3.6 that describe the metaheuristic algorithms developed and tested in this paper.

3.1 Solution framework

Figure 1 illustrates the general scheme of the solver. Since the ATC-TCA problem is an NP-hard problem, we adopt a temporal decomposition and a decomposition in routing and scheduling variables. The former is solved via the rolling horizon procedure in [37], while the latter is solved via the scheduling and re-routing algorithms of the AGLIBRARY solver. Specifically, we use the scheduling algorithms in [11], the re-routing algorithms in [12], the new scheduling and re-routing algorithms developed in this paper. The two decomposition frameworks can be further combined together.

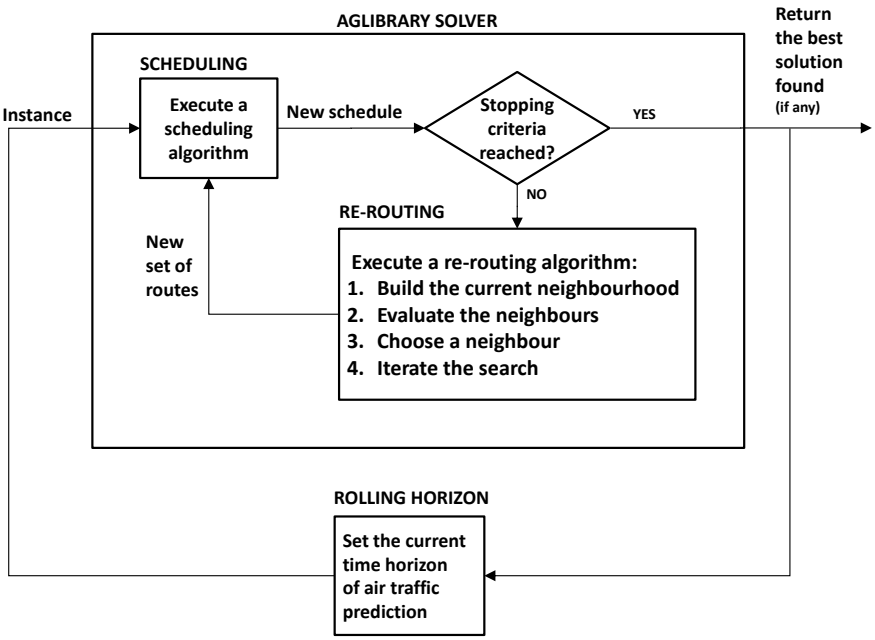

Figure 1: A general scheme of the solver

The rolling horizon decomposition framework divides the ATC-TCA problem into time horizons of traffic predictions. Each time horizon is a sub-problem instance to be solved by the AGLIBRARY solver. We assume that all aircraft information is known at the start time t_0 of the traffic prediction. This rolling horizon framework corresponds to a centralized framework when the overall problem is solved with a single time horizon (i.e. when no temporal decomposition is performed).

The decomposition framework into routing and scheduling works instead as follows. The AGLIBRARY solver iterates between the computation of a new aircraft schedule for a set of routes, and the selection of a new set of routes. The basic idea is to first compute an aircraft scheduling solution given fixed (default) routes, and then search for better aircraft routes. The latter procedure is based on a local search for routing alternatives starting from the scheduling solution, and an iterative scheduling and re-routing technique to continue the search. The iterative procedure returns the best aircraft schedule and the best set of routes after a stopping criteria is reached. In this paper, the maximum computation time is a stopping criteria.

The overall framework returns a feasible aircraft schedule in which a route is fixed for each aircraft and all potential routing conflicts are solved. In case no feasible schedule is computed, the solver reports the conflicting routes via a detailed time-space diagram. Based on the information provided by the solver, the en-route/ground human traffic controllers could take suitable re-scheduling actions on the potential conflicts that are not allowed by the automated decision support system, including re-routing some aircraft to other resources in the same or other airports.

3.2 Routing neighbourhoods

This subsection describes the neighbourhood structures used in this paper. To this aim, we need to introduce the following notation. Let $S(F)$ be a ATC-TCA solution with the routes defined in F and the sequencing decisions defined in S, and let $\mathcal{G}(F,S)$ be the graph of this solution. The search for a better solution is based on the computation of a new graph $\mathcal{G}'(F',S')$. This graph differs from the former $\mathcal{G}(F,S)$ by a different route for some aircraft, and different orders and times of operations. This corresponds to a neighbour, in metaheuristics terms. The longest path in $\mathcal{G}'(F',S')$ is denoted as $l^{S'(F')}(s,t)$. Our intuition is to shorten the weight of the longest path in $\mathcal{G}(F,S)$, i.e. the critical path, by re-routing some aircraft. We observe that F' improves over F in terms of the objective function value if $l^{S'(F')}(s,t) < l^{S(F)}(s,t)$.

The routing neighbourhoods studied in this paper are based on observations on the graph $\mathcal{G}(F,S)$ regarding the nodes that represent operations involved in the resolution of potential aircraft conflicts. To this aim, we need to introduce the following concepts. A *critical node* is a node on the longest path from the start node s to the end node t in $\mathcal{G}(F,S)$, that is called the *critical path set* $\mathcal{C}(F,S)$. A *waiting node* is a critical node in $\mathcal{C}(F,S)$ representing an aircraft k traversing a shared resource with a consecutive delay caused by the resolution of a conflicting request between aircraft k and another aircraft u, while a *hindering node* is another critical node in $\mathcal{C}(F,S)$ related to aircraft u scheduled before aircraft k on the shared resource. Formally, for given a solution $S(F)$, $krp \in N(F) \setminus \{s,t\}$ is a critical node of aircraft k with route r if $l^{S(F)}(s,krp) + l^{S(F)}(krp,t) = l^{S(F)}(s,t)$. A critical node krp is a waiting node if $l^{S(F)}(s,krp) > l^{S(F)}(s,\nu(krp)) + w^F_{\nu(krp),krp}$, where the node $\nu(krp)$ precedes the node krp on route r. For each waiting node krp, there is at least one hindering node $\eta(krp)$ in $\mathcal{G}(F,S)$, different from node $\nu(krp)$, such that $l^{S(F)}(s,krp) = l^{S(F)}(s,\eta(krp)) + w^F_{\eta(krp),krp}$.

We investigate strategies for the selection of alternative routes for some aircraft based on the identification of *backward and forward ramifications* of the critical path in $\mathcal{G}(F,S)$. Intuitively, a backward (forward) ramification is an extension of the critical path that incorporates all the nodes proceeding (following) the critical nodes. Formally, for a given node $krp \in N(F)\setminus\{s,t\}$, we recursively define the backward ramification

$R_B(krp)$ as follows. If krp is a waiting node, then $R_B(krp) = R_B(\nu(krp)) \cup R_B(\eta(krp) \cup \{krp\}$, otherwise $R_B(krp) = R_B(\nu(krp)) \cup \{krp\}$. Similarly, we recursively define the forward ramification $R_F(krp)$ as follows. If krp is the hindering node of a waiting node dij, then $R_F(krp) = R_F(\sigma(krp)) \cup R_F(dij) \cup \{krp\}$, where node $\sigma(krp)$ follows node krp on route r. Otherwise, $R_F(krp) = R_F(\sigma(krp)) \cup \{krp\}$. By definition, $R_B(s) = R_F(s) = \{s\}$ and $R_B(t) = R_F(t) = \{t\}$. Given $\mathcal{C}(F, S)$, we define a *ramified critical path* set as $\mathcal{F}(F, S) = \bigcup_{krp \in \mathcal{C}(F,S)} [R_B(krp) \cup R_F(krp)]$, and a *backward ramified critical path* set as $\mathcal{B}(F, S) = \bigcup_{krp \in \mathcal{C}(F,S)} [R_B(krp)]$.

We study the five neighbourhood structures listed below.

- *Complete K-Route neighbourhood* $\mathcal{N}_{CKR}$: contains all the feasible solutions to the ATC-TCA problem in which K aircraft follows a different route compared to the incumbent solution. To limit the number of neighbours to be evaluated, $\mathcal{N}_{CKR}$ is only partially explored as follows. A move is obtained by choosing K routes different from the ones of the current solution at random (i.e. all alternative routes having the same probability), until a number ψ (parameter) of alternative routing solutions is obtained:

- *Ramified Critical Path Operations neighbourhood* $\mathcal{N}_{RCPO}$ considers only the routing alternatives for the aircraft associated to the nodes in $\mathcal{B}(F, S)$ plus $\mathcal{F}(F, S)$. The idea is that the maximum consecutive delay of an optimal solution to the ATC-TCA problem can be reduced by removing aircraft conflicts causing it. This requires either removing, anticipating or postponing some operations from the critical path set (i.e. re-routing the aircraft associated to the critical path on the graph $\mathcal{G}(F, S)$ of the incumbent solution). The latter result can be obtained by re-routing some aircraft represented by jobs with nodes in $\mathcal{B}(F, S)$ or $\mathcal{F}(F, S)$ and then re-scheduling aircraft movements;

- *Waiting Operations Critical Path neighbourhood* $\mathcal{N}_{WOCP}$ is a restriction of $\mathcal{N}_{RCPO}$ that considers the routing alternatives for the aircraft associated to the waiting nodes in $\mathcal{C}(F, S)$;

- *Delayed Jobs neighbourhood* $\mathcal{N}_{DJ}$ considers only the aircraft (jobs) that have a consecutive delay on some due date arcs in the graph $\mathcal{G}(F, S)$ of the incumbent solution;

- *Free-Net Waiting Operations Jobs neighbourhood* $\mathcal{N}_{FNWJ}$ considers only the aircraft (jobs) that have some waiting nodes in the alternative graph $G(N, F, A)$ of the incumbent solution. A waiting node is identified by computing the consecutive delay that would be associated in $G(N, F, A)$ by selecting an alternative arc (i.e. the one generating the waiting node) and by disregarding all the other arcs in A.

3.3 Heuristic evaluation of routing neighbours

The choice of a best neighbour in the neighbourhood requires the computation of a new ATC-TCA solution $S'(F')$ starting for an incumbent solution $S(F)$, that is characterized by the routing decisions in F' and the sequencing decisions in S'. To this aim, we use fast heuristics based on a two-step graph building procedure in which the graph $\mathcal{G}(F, S)$ is translated into the graph $\mathcal{G}'(F', S')$. In the first step, a sub-graph of $\mathcal{G}'(F', S')$ is generated by considering all the nodes in $N(F^I)$ associated to the routes modelled by the arcs in $F^I = F \bigcap F'$, all the fixed directed arcs in F^I and all the alternative arcs in $S(F)$ incident in a node in $N(F^I)$. This corresponds to keeping a subset of decisions from the incumbent solution into the neighbour solution.

In the second step, the fixed directed arcs in $F^R = F' \setminus F^I$ and the nodes in $N(F^R)$ are added to the sub-graph. Finally, $\mathcal{G}'(F', S')$ is obtained by adding a selection of alternative arcs $S'(F^R)$ to the sub-graph. The selection $S'(F^R)$ is computed by taking the best solution among those computed via two greedy algorithms (i.e. the AMSP and AMCC algorithms) described in D'Ariano et al. [11].

3.4 Tabu search re-routing algorithm

The *Tabu Search* (TS) is a deterministic metaheuristic based on local search, which makes extensive use of memory for guiding the search [17]. A basic ingredient is the *tabu list*, that is used to avoid being trapped in local optima and revisiting the same solution. From the incumbent solution, non-tabu moves define a set of solutions, named the *incumbent solution neighbourhood*. At each step, the best solution in this set is chosen as the new incumbent solution. Some attributes of the former incumbent are then stored in the tabu list. The moves in the tabu list are forbidden as long as these are in the list, unless an *aspiration criterion* is satisfied. The tabu list length can remain constant or be dynamically modified during the search.

The Tabu Search (TS) of D'Ariano et al. [12] is used in the iterative scheduling and re-routing framework. The neighbourhood strategy used by TS explores candidate solutions in $\mathcal{N}_{RCPO}$ unless this neighbourhood is empty. In the latter case, ψ (parameter) consecutive moves are performed in $\mathcal{N}_{CKR}$ with $K = 1$ before searching again in $\mathcal{N}_{RCPO}$. All neighbours are evaluated via the scheduling heuristics of Section 3.3. The best neighbour is set as the move to be made, and evaluated via the branch-and-bound algorithm of [11]; the resulting best solution is set as the new incumbent solution. The inverse of the chosen move is stored in a tabu list of length λ (parameter). The moves in the tabu list are forbidden for λ iterations and no aspiration criteria is used. When no potentially better solution is found on the incumbent solution neighbourhood, the search alternates the above neighbourhood strategy with a diversification strategy, which consists of changing at random the route of μ (parameter) aircraft at the same time. From the tuning performed in [12], the best overall exploration strategy has the following parameter values $\psi = 10$, $\lambda = 32$ and $\mu = 5$.

3.5 Variable neighbourhood search re-routing algorithm

A *Variable Neighbourhood Search* (VNS) is proposed to efficiently solve the ATC-TCA problem. This metaheuristic is based on the combination of different neighbourhoods. Neighbourhood changes are proposed both in a local search phase in order to compute a local minimum, and in a perturbation phase in order to escape from a local minimum [19]. The intuition behind the choice of this search method for solving the ATC-TCA problem is as follows. The metaheuristic search starts from a reference ATC-TCA solution with fixed routes and needs to explore various possibilities to generate new solutions in the local search phase. Multiple aircraft are simultaneously re-routed and new aircraft scheduling solutions are computed in order to evaluate a different load of the TCA resources and to investigate the sequence flexibility when solving the potential aircraft conflicts. The choice of re-routing multiple simultaneous aircraft differs from the local search used in TS, in which a new solution is generated by changing the route of a single aircraft and by considering the aircraft on the ramified critical path only. The proposed VNS makes use of neighbourhoods based on different candidate aircraft and routing alternatives in the runway and/or air segment resources.

The VNS algorithm of Figure 2 is an adaptation of the basic VNS described in Hansen et al. [19]. This algorithm combines the classic ingredients of the VNS algorithm with the routing neighbourhood structures of Section 3.2 and new sophisticated neighbourhood search strategies to search for better aircraft routes.

The general structure of the VNS is the following. The algorithm starts from an incumbent solution of the ATC-TCA problem, named $IncSol$ $\mathcal{G}(F,S)$, computed via the branch-and-bound algorithm of [11] given a default (off-line) route to each aircraft. A counter K is adopted to fix the number of aircraft that are re-routed in each move. The initial value of K is set to 1, i.e. a single aircraft is re-routed in $IncSol$ $\mathcal{G}(F,S)$. The metaheuristic iterates the search for better solutions starting from $IncSol$ until a maximum computation time T_{max} is reached or until the maximum consecutive delay is larger than 0. At each iteration, a neighbourhood of $IncSol$ is generated and a new solution $IncSol'$ is selected via a *shaking* procedure (see below). The search continues with the generation of a neighbourhood of $IncSol'$ and a local search based on best-improvement is performed in a restricted neighbourhood. Then, a *Move Or Not* function is performed as follows. In case an improving move $IncSol''$ is obtained via the local search (i.e. $f(IncSol'') < f(IncSol)$), a new iteration is performed by setting $IncSol''$ as the new incumbent solution and K is set to 1. Otherwise, the parameter K is set to $K+1$ and a new iteration is performed until $K \leq K_{max}$. When $K = K_{max}$ the algorithm diversifies the search with a change of neighbourhood structure (if the algorithm works with a single neighbourhood structure this step is not performed). The metaheuristic returns the best ATC-TCA solution ($IncSol$) and the objective function value ($f(IncSol)$). The pseudo-code of the VNS is reported in Figure 2. We next describe the specific features of the VNS.

Build Neighbourhood. Starting from an incumbent solution, the $\mathcal{N}_{CKR}$ neighbourhood is generated, in which exactly K aircraft are re-routed in the graph $\mathcal{G}(F,S)$ of the incumbent solution.

Shake. This is a typical diversification procedure that consists here in changing the route of K aircraft randomly in the $\mathcal{N}_{CKR}$ neighbourhood of the incumbent solution ($IncSol$), and in computing a new incumbent solution ($IncSol'$) via the scheduling heuristics of Section 3.3 and the new set of routes.

Neighbourhood Search Strategy. This procedure is proposed in order to reduce the local search to the evaluation of up to L neighbours in the current neighbourhood. Starting from an incumbent solution and the $\mathcal{N}_{CKR}$ neighbourhood of this solution, a restricted neighbourhood is generated by using a given neighbourhood structure $\mathcal{N}_i$. The selection of L neighbours is achieved in the following steps:

1. *aircraft ranking* : Each aircraft gets a score based on the criterion specified in a neighbourhood structure $\mathcal{N}$. The score is used to decide how many times the aircraft has to be re-routed in the L neighbours; This ranking is based on one of the neighbourhood structures of Section 3.2. In $\mathcal{N}_{RCPO}$, each aircraft gets a score based on the maximum value $l^{S(F)}(s,krp) + l^{S(F)}(krp,t)\ \forall\ (krp)$ in the ramified critical path of the graph $\mathcal{G}(F,S)$ of the incumbent solution. In $\mathcal{N}_{WOCP}$, each aircraft gets a score based on the sum of the consecutive delays associated at each critical node in the graph $\mathcal{G}(F,S)$. In $\mathcal{N}_{DJ}$, each aircraft gets a score based on the maximum consecutive delay associated at its due date arcs of the graph $\mathcal{G}(F,S)$. In $\mathcal{N}_{FNWJ}$, each aircraft gets a score based on the sum of the consecutive delays associated at a restricted set of waiting nodes in the graph $G(N,F,A)$ of the incumbent solution. We only consider the alternative pairs in which both the alternative arcs generate a consecutive delay. For those alternative pairs, we take the alternative arc generating the largest consecutive delay in

$G(N, F, A)$ and the corresponding waiting node.

2. *route ranking* : The routes of each aircraft get a score based on the distance from the route of the incumbent solution. The larger is the difference between the routes, the higher is the score. The route ranking thus suggests for each aircraft to select the most different routes. Among the selected routes, the ranking gives precedence to the routing alternatives in which there is a change of runway, since this is often the conflicting resource of the TCA aircraft routes;

3. *neighbour generation* : This is the assignment of the routes to the aircraft in each neighbour. This is done by selecting the aircraft based on the aircraft ranking and by selecting the routes based on the route ranking. A combinatorial combination of the routes is used in order to generate L different neighbours. In each neighbour, exactly K aircraft are re-routed compared to the incumbent solution. The neighbours are ordered based on the aircraft ranking, and in case of tie on the route ranking.

Algorithm VNS
Input: $IncSol$ $\mathcal{G}(F, S)$, K_{max}, T_{max}, L, $\mathcal{N}_1$, $\mathcal{N}_2$
$\mathcal{N}_i \leftarrow \mathcal{N}_1$,
While $(T < T_{max})$ & $(f(IncSol) > 0)$ **do**
  Begin
  $K \leftarrow 1$,
  While $(K \leq K_{max})$ **do**
    Begin
    $BuildNeighbourhood(IncSol, K)$,
    $IncSol' \leftarrow Shake(IncSol, K)$,
    $BuildNeighbourhood(IncSol', K)$,
    $NeighbourhoodSearchStrategy(IncSol', K, L, \mathcal{N}_i)$,
    $IncSol'' \leftarrow FirstImprovement(IncSol')$,
    $(IncSol, K) \leftarrow MoveOrNot(IncSol, IncSol'', K)$,
    If $(K = K_{max})$ **do**
      Begin
      $\mathcal{N}_i \leftarrow NeighbourhoodChange(IncSol, \mathcal{N}_i, \mathcal{N}_1, \mathcal{N}_2)$,
      End
    $T \leftarrow$ CPU time()
    End
  End

Figure 2: Sketch of the VNS algorithm

Figure 3 presents a numerical example of the neighbourhood search strategy, in which four aircraft ($J = \{J1, J2, J3, J4\}$) can be re-routed in a TCA. J1 and J4 have four alternative routes (e.g., the routes of $J1$ are: $J1$-1, $J1$-2, $J1$-3, $J1$-4), while J2 and J3 have two alternative routes each. In the incumbent solution, all aircraft use the first route ($J1$-1, $J2$-1, $J3$-1, $J4$-1). The parameters of the procedure are set to

the following values: $L = 4$ and $K = 2$ (i.e. the neighbourhood is restricted to 4 neighbours and 2 aircraft are re-routed in each neighbour).

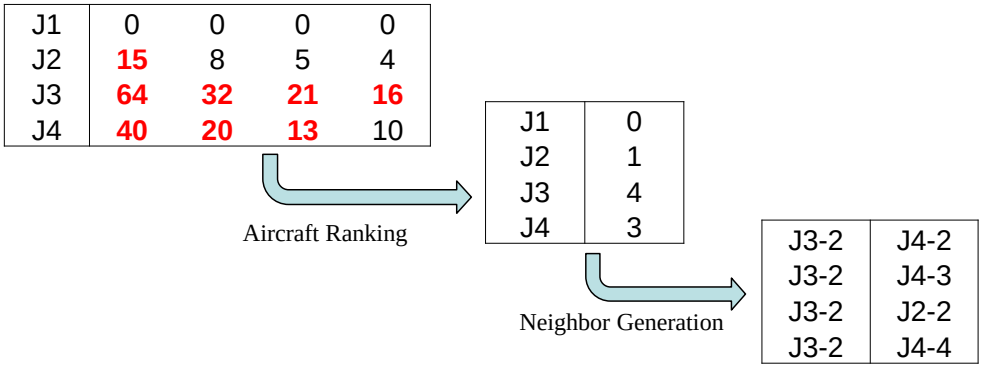

Figure 3: Example of neighbourhood search strategy with $|J| = 4$, $L = 4$ and $K = 2$

The aircraft ranking determines a score matrix in which each row represents an aircraft and each column the number of times each aircraft can be re-routed. This score matrix is depicted in the left-hand side of Figure 3. Specifically, the first column reports the score for each aircraft based on the neighbourhood structure $\mathcal{N}_{DJ}$ (e.g., the value 40 in this example is the maximum consecutive delay associated at aircraft $J4$). The other columns report the score of the first column divided by the column number, e.g., $40/2 = 20$, $40/3 = 13$, $40/4 = 10$. By taking the highest KL scores in the score matrix (these values are reported in bold in Figure 3), we get the number of times each aircraft has to be re-routed in the four neighbours (e.g., $J4$ is re-routed in three neighbours). This is reported in the center table of Figure 3.

The route ranking orders the list of re-routing alternative of each aircraft based on maximizing the difference with the incumbent route and giving precedence to the routing alternatives in which there is a change of runway. In this case, the route ranking of $J4$ is $J4$-2, $J4$-3, $J4$-4 and the most different route with a change of runway is $J4$-2.

The neighbour generation procedure assigns the routes to the aircraft in each neighbour. In this example, $J4$ appears in the neighbours with its three different routes, while $J2$ and $J3$ have only a route to be chosen. Every row of the table in the right-hand side of Figure 3 corresponds to a candidate move.

First Improvement. This is a local search procedure in the restricted neighbourhood in which the candidate moves are ordered based on the neighbour generation procedure. At each step of the procedure, the neighbour with the highest ranking in the restricted neighbourhood is considered, a new graph is build with the new routes of the neighbour, and a new solution is computed via the scheduling heuristics of Section 3.3 for the new set of routes. This procedure lasts until a better solution is obtained compared to the incumbent solution, or until all L neighbours in the restricted neighbourhood have been evaluated.

Move Or Not. This procedure is responsible for possibly making a move. In the case where the best solution found in the neighbourhood is better than the incumbent, the resulting graph is solved by the branch-and-bound algorithm of [11], and the best solution is set as the new incumbent solution. Otherwise, the best solution in the neighbourhood is chosen as incumbent, or some diversification strategy is employed.

Neighbourhood Change. This procedure diversifies the search by alternating K iterations of the neighbourhood search strategy with $\mathcal{N}_1$, and K iterations of the neighbourhood search strategy with $\mathcal{N}_2$.

3.6 Hybrid search re-routing algorithm

We introduce a hybrid metaheuristic, named Variable Neighbourhood Tabu Search (VNTS), for solving the ATC-TCA problem. This algorithm consists of a combination of VNS and TS, as proposed earlier in other contexts by Moreno Pérez et al. [27]. The hybridization is proposed in order to take promising ideas from both the metaheuristics. The VNS is used in order to explore different neighbourhoods of the incumbent solution, so that the reference aircraft scheduling solution can be improved in terms of multiple routing modifications. The TS is used to avoid cycling and is combined with aircraft re-routing strategies to escape from local minimum. Specifically, the ingredients of the proposed hybrid metaheuristic are an intensification strategy based on the exact exploration of a restricted aircraft re-routing neighbourhood, and various diversification strategies for aircraft re-routing in different runway and air segment resources and on a restart technique based on a list of potentially useful routing alternatives for each aircraft.

Figure 4 introduces the pseudo-code of the hybrid metaheuristic. Starting from an incumbent solution *IncSol* $\mathcal{G}(F, S)$, a given neighbourhood structure $\mathcal{N}_i$ and a counter Q (initialized as 0), the metaheuristic iterates the search for better solutions in various neighbourhoods until a maximum computation time T_{max} is reached, as far as the maximum consecutive delay is larger than 0. Each iteration evaluates all the neighbours in a restricted neighbourhood of *IncSol* (i.e. L non-tabu neighbours, determined via a neighbourhood search strategy, analogously to the VNS procedure) and the best neighbour is implemented via the branch-and-bound algorithm of [11]. The *Move Or Not* function is performed as for the VNS algorithm of Figure 2. However, when $K = K_{max}$ the algorithm diversifies the search as follows. If $Q < Q_{max}$, a change of neighbourhood structure is implemented, and Q is set to $Q + 1$; otherwise a restart strategy based on a problem-specific memory structure is applied and Q is set to 0. Before a new iteration is performed, a tabu search memory and a time counter are updated. We next describe key VNTS features in more detail.

Best Improvement. Given a restricted neighbourhood of an incumbent solution, this procedure evaluates all neighbours via the scheduling heuristics of Section 3.3. The best neighbour is set as the move to be made, and evaluated via the branch-and-bound algorithm of [11]; the resulting best solution is set as the new incumbent solution.

Neighbourhood Change. This procedure diversifies the search by alternating K iterations of the neighbourhood search strategy with $\mathcal{N}_1$, with K iterations of the neighbourhood search strategy with $\mathcal{N}_2$. In particular, when $Q < Q_{max}$, $K = K_{max}$ and the current neighbourhood structure $\mathcal{N}_i = \mathcal{N}_1$, the procedure sets $\mathcal{N}_i = \mathcal{N}_2$ and the search continues with K neighbourhood search strategy iterations with $\mathcal{N}_2$.

Restart Strategy. This is a typical diversification strategy that we implement as follows. A restart memory structure is used to store the number of times each aircraft re-routing has been evaluated in any neighbour so far. From this structure, we compute an ordered list of routing alternatives for each aircraft from the less frequently used to the most frequently used. The restart strategy consists of the implementation of a move (disregarding the tabu memory) in which the less frequently used route is set for each aircraft. In case of tie, a random decision is taken for each aircraft among its least-used routes. The restart memory

Algorithm VNTS
Input: $IncSol$ $\mathcal{G}(F,S)$, K_{max}, T_{max}, L, TL_{max}, Q_{max}, $\mathcal{N}_1$, $\mathcal{N}_2$
$Q \leftarrow 0$,
$\mathcal{N}_i \leftarrow \mathcal{N}_1$,
While $(T < T_{max})$ & $(f(IncSol) > 0)$ **do**
Begin
$K \leftarrow 1$,
While $(K \leq K_{max})$ **do**
Begin
$BuildNeighbourhood(IncSol, K)$,
$TabuMemoryFilter(TL)$,
$NeighbourhoodSearchStrategy(IncSol, K, L, \mathcal{N}_i)$,
$IncSol' \leftarrow BestImprovement(IncSol)$,
$(IncSol, K) \leftarrow MoveOrNot(IncSol, IncSol', K)$,
If $(K = K_{max})$ **do**
Begin
If $(Q < Q_{max})$ **do**
Begin
$\mathcal{N}_i \leftarrow NeighbourhoodChange(IncSol, \mathcal{N}_i, \mathcal{N}_1, \mathcal{N}_2)$,
$Q \leftarrow Q + 1$,
End
Else
Begin
$RestartStrategy(Q)$,
$Q \leftarrow 0$,
End
End
$TL \leftarrow TabuMemoryUpdate(TL, TL_{max}, Q, Q_{max})$,
$T \leftarrow$ CPU time()
End
End

Figure 4: Sketch of the VNTS algorithm

structure is reset each time the restart strategy is used.

Tabu Memory. This technique is used to avoid cycling and revisiting the same solution. This is achieved by two functions named *tabu memory filter* and *tabu memory update*. The former function removes from the current neighbourhood all the tabu moves (no aspiration criteria is used), while the latter function updates the list of tabu moves during the search. The tabu list (TL) used in the VNTS is made by up to TL_{max} moves. For each move that has been implemented in a local search step, the following key information is stored: the re-routed aircraft of the move, and the old route chosen for each re-routed aircraft. For

example, consider an instance with three aircraft and three routes for each aircraft. An incumbent solution is $J1-1, J2-1, J3-1$ and the move performed by a local search is $J1-1, J2-2, J2-3$. The tabu memory update will store in the tabu list the following information: $J2$ and $J3$ have been re-routed and the tabu-routes are $J2-1$ and $J3-1$. A move is tabu when the two routes $J2-1$ and $J3-1$ are chosen as the new routes of $J2$ and $J3$. The tabu list is reset when a restart move is performed (i.e. when $Q = Q_{max}$).

4 Computational experiments

This section presents the experimental assessment of the various metaheuristics of Section 3. The test bed is the Milan Malpensa terminal control area (MXP) and the instances are taken from Samà et al. [38]. In particular, we evaluate the metaheuristics on the most complex instances with strong disturbances and a large number of aircraft. The objective of this evaluation is to report the marginal improvement achieved by the new metaheuristics compared with the approaches in [38]. To this end, the experiments are executed on the same processor used in [38], that is Intel Core 2 Duo E6550 (2.33 GHz), 2 GB of RAM, Windows XP. For all the metaheuristics tested in this paper, we adopt the deadline implications and the pre-processing procedure developed in [38]. For all the approaches based on the rolling horizon framework, we use the same parameter setting in [38] (i.e. we fix the roll period to 10 min and the look-ahead period to 15 min).

4.1 Description of the TCA

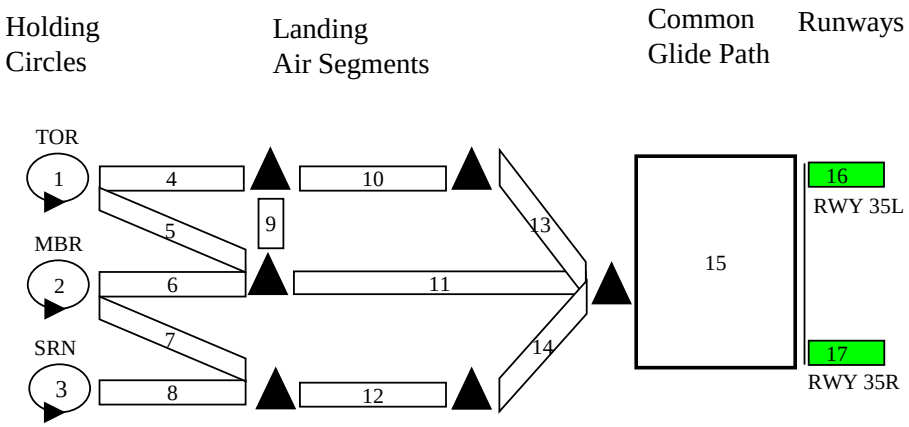

Figure 5: Malpensa (MXP) Terminal Control Area

Figure 5 shows the TCA scheme of Malpensa (MXP) TCA. There are two runways (RWY 35L, RWY 35R), used both for departing and arriving procedures. The MXP resources are: three airborne holding circles (resources 1-3 in Figure 5, named TOR [Torino], MBR [Mebur], SRN [Saronno]), eleven air segments for arriving procedures (resources 4-14), a common glide path (resource 15) and two runways (resources 16-17). The common glide path resource includes two parallel air segments before the runways for which traffic regulations impose a minimum diagonal/longitudinal distance between landing aircraft. Note that

only air segment 9 can be traversed in both directions, while all the other air segments can only be traversed from the entrance of the TCA towards the runways.

4.2 Tested instances

The ATC-TCA instances are based on real data collected for the Milan Malpensa TCA. We consider a subset of the instances generated in [38], that are obtained by varying the following parameters: (i) the model variant, (ii) the time horizon of traffic prediction, (iii) the aircraft delays, (iv) the disruption.

Model variant. Three variants are investigated to model objective functions and user requirements. Model 1 (M1) measures the delay of landing and take-off aircraft at the runways, which are the most used TCA resources. Model 2 (M2) modifies the objective function for landing aircraft by measuring their delay both at the runways and at the entrance of the TCA, penalizing a late entrance in the TCA. The latter model takes into consideration the extra work required to coordinate the solutions of TCA with en-route traffic controllers. Model 3 (M3) extends M2 with additional deadline constraints that limit the maximum possible entrance time of each landing aircraft in the TCA. These constraints are inserted in order to consider the limited possibility of airborne holding, which are more expensive and constrained than ground holding.

Time horizon of traffic prediction. Two time horizons of different length are considered: 60 and 180 minutes. The shorter time horizon can be considered of practical interest for the management of light disturbances, while the longer time horizon can better assess the traffic control measures in terms of aircraft delay propagation. The latter time horizon is particularly relevant to study in case of disruptions.

Aircraft delays. Disturbed traffic conditions are generated by delaying the entrance time of some aircraft in the TCA. The entrance delays are randomly generated according to a uniform distribution, and are applied at some aircraft entering the TCA during the first half of the time horizon under examination. For each time horizon of traffic prediction, we consider 10 delay instances in which random delays are up to 5 minutes and other 10 delay instances in which random delays are up to 15 minutes.

Disruption. We consider the case in which one of the two runways of the TCA is unavailable in a time window, and all landing and take-off aircraft have to be scheduled on the only available runway during the disruption. The disrupted case is only studied for the 180-minute traffic predictions with a disrupted runway between the first and the second hour of traffic prediction.

In total, there are three groups of ATC-TCA instances: two groups regard undisrupted operations with different time horizons (60 and 180 minutes), for a total of 120 delayed instances (i.e. for the three model variants, the two time horizons, and the 20 aircraft delays); one group of 60 disrupted instances (i.e. the three model variants, the 180-minute time horizon, and the 20 aircraft delays).

Table 1 gives average information on the aircraft delay instances regarding the MILP formulation; every row is an average over the 20 delay instances. Moreover, Column 1 reports the time horizon of traffic prediction, Column 2 the model variant, Column 3 the number of fixed constraints, Column 4 the number of alternative constraints, Column 5–7 the number of MILP variables, Column 8 the number of aircraft. The model variants differ in terms of the fixed constraints, since different due date and deadline arcs are used. Disrupted instances have the same characteristics (concerning the variables in Table 1) as the corresponding undisrupted instances.

Table 1: Size of the MILP formulation for the various ATC-TCA instances

Time Horizon	Model Variant	Num of Fixed Constraints	Num of Alternative Constraints	MILP Variables			Num of Aircraft
				h	x	y	
60 min	M1	728	11526	264	5763	62	40
	M2	751	11526	264	5763	62	40
	M3	774	11526	264	5763	62	40
180 min	M1	4331	456476	871	228238	414	117
	M2	4649	456476	871	228238	414	117
	M3	4967	456476	871	228238	414	117

4.3 Assessment of the VNS and VNTS parameters

This section discusses the choice of alternative configurations for the VNS and VNTS algorithms, while the TS algorithm configurations are evaluated in previous works [12, 38]. The computational assessment is based on 20 pilot ATC-TCA instances with aircraft delays. Disrupted instances have not been considered since these present a reduced number of alternative routes. The assessment is based on the evaluation of the metaheuristics in the centralized framework with $T_{max} = 180$ seconds.

Table 2: Experimental setting of algorithmic parameters

T_{max} (sec)	BB Time (sec)	K_{max}	L	TL_{max}	Q_{max}
180	4/**10**/20/30	2/**3**/4/5	5/**10**/20	0/8/15/**32**	2/**3**/4

$\mathcal{N}$	$\mathcal{N}_1 + \mathcal{N}_2$
DJ / RCPO / WOCP / FNWJ	**WOCP+FNWJ** / FNWJ+WOCP / DJ+WOCP / WOCP+DJ

Table 2 shows the parameters considered in the algorithm assessment. The parameters used in both algorithms are the time given to the Branch-and-Bound algorithm (BB Time, in seconds), the number of aircraft that are re-routed in the current neighbourhood (K_{max}), the size of the restricted neighbourhood (L), and the choice of the neighbourhood structure ($\mathcal{N}_1$ + $\mathcal{N}_2$ or $\mathcal{N}$ when the two coincide). Additional parameters of the VNTS are the tabu list length (TL_{max}) and the counter used for the diversification strategies (Q_{max}). The value of each parameter that yielded the best results is reported in bold and used in all the presented experiments.

4.4 Assessment of the solution quality

This section presents the results obtained for the best metaheuristics developed in this paper and compares them with the results obtained in Samà et al. [38], that is used as a benchmark comparison for the newly developed algorithms. Table 3 gives an overview of the best ALGOrithm in [38] (ALGO, Column 4), the best CEntralized MetaHeuristic (CE MH, Column 5) and the best Rolling Horizon MetaHeuristic (RH MH, Column 6) for the instances grouped by the time horizon of traffic prediction (Column 1), the type of disturbance (Column 2), the model variant (Column 3). Each row reports the best algorithm on the set of 20 ATC-TCA instances described in Section 4.2.

We now recall the different acronyms used: MILP refers to the Mixed-Integer Linear Programming formulation of the ATC-TCA problem solved by using the IBM ILOG CPLEX MIP 12.0 solver; VNS, VNTS and TS refer to the metaheuristics Variable Neighbourhood Search, Variable Neighbourhood Tabu Search and Tabu Search; DJ and WOCP+FNWJ refer to the restricted neighbourhood Delayed Job and the combined restricted neighbourhoods Waiting Operation Critical Path and Free-Net Waiting Operations. We next give detailed information on the performance of the various best algorithms reported in Table 3.

Table 3: Best algorithms for various groups of instances

Time Horizon	Disturbance Type	Model Variant	Best Algo [38]	Best CE MH	Best RH MH
60 min	Normal	M1	RH MILP	VNTS DJ	VNS DJ
		M2	CE MILP	VNTS DJ	VNS DJ
		M3	CE MILP	VNTS DJ	VNS DJ
180 min	Normal	M1	RH MILP	VNTS DJ	VNS DJ
		M2	RH MILP	VNTS DJ	VNS DJ
		M3	RH MILP	VNTS DJ	VNS DJ
180 min	Disrupted	M1	RH TS	VNS WOCP+FNWJ	VNS DJ
		M2	RH TS	VNS WOCP+FNWJ	VNS DJ
		M3	RH TS	VNS WOCP+FNWJ	VNS DJ

Regarding CE MILP, RH MILP and RH TS, we use the same setting of the parameters and thus the same results presented in Samà et al. [38]. We recall that their time limit of computation is 240 (720) seconds for the 60-minute (180-minute) instances.

Regarding the metaheuristics developed in this paper, the parameters are set as described in Section 4.3. The best metaheuristics have been taken among VNS WOCP+FNWJ, VNS DJ, VNTS WOCP+FNWJ, VNTS DJ. We recall that the time limit of computation is 180 seconds for the CE MH. The RH MH have a time limit of 30 seconds (10 seconds) for each roll period of the 60-minute (180-minute) instances, since 6 (18) periods are required to solve the overall time horizon and the maximum computation time is also fixed to 180 seconds. The BB time limit for the CE MH (RH MH) is 10 seconds (4 seconds).

Table 4 provides the results obtained for the three groups of instances: the 60-minute delayed instances, the 180-minute delayed instances and the 180-minute disrupted instances. Column 1 presents a three-field code in order to identify each group of instances, where each code is structured as follows: [model variant]-[time horizon]-[NOrmal or DISrupted traffic]. Columns 2-3 report the average best objective function value (in seconds) obtained by all the algorithms tested in [38], and the average computation time (in seconds) at which that value was found. Columns 4-5 report the average best objective function value found by all the metaheuristics developed in this paper combined with the centralized and rolling horizon frameworks, and their average computation time. Columns 6-7 (8-9) (10-11) report the average best objective function value and the average computation time obtained by using only the best algorithm in [38] (the best CEntralized MetaHeuristic) (the best Rolling Horizon MetaHeuristic), as indicated respectively in Table 3. The best average values are reported in bold regarding the best solution and the best algorithm columns. The results

obtained for individual ATC-TCA instances can be found in [36].

Table 4: Average results for each type of instance

Instance Type	Best Sol [38]		Best Sol MH		Best Algo [38]		Best CE MH		Best RH MH	
	Value (sec)	Time (sec)	Value (sec)	Time (sec)	Value (sec)	Time (sec)	Value (sec)	Time (sec)	Value (sec)	Time (sec)
M1-60-NO	**4.0**	7.1	**4.0**	27.0	**4.0**	7.1	4.4	19.1	5.7	180
M2-60-NO	**7.45**	44.9	**7.45**	34.3	7.5	44.9	**7.45**	34.3	8.5	180
M3-60-NO	**7.45**	22.1	**7.45**	37.7	8.7	10.1	**7.45**	37.7	8.1	180
M1-180-NO	**24.5**	330.9	**24.5**	133.4	**31.2**	273.7	41.4	89.0	36.5	180
M2-180-NO	39.2	348.2	**37.6**	147.5	42.9	348.4	43.9	99.6	**42.7**	180
M3-180-NO	**25.1**	403.8	37.8	138.2	**25.7**	388.3	43.9	99.8	44.2	180
M1-180-DIS	749.3	506.8	**699.7**	107.4	844.6	720	**756.2**	24.6	858.9	180
M2-180-DIS	**688.8**	641.9	691.2	100.9	770.8	720	**747.1**	40.4	842.7	180
M3-180-DIS	795.5	627.5	**699.0**	92.3	872.2	720	**806.4**	30.1	907.0	180

For the 60-minute instances of Table 4, the best solution found by the new metaheuristics (Column 4) is always equal to the best solution found by the algorithms in [38] (Column 2), and all these solutions are proven optimal. When comparing the algorithms, the best CE MH is the best algorithm in terms of the average objective function value, even if it often requires a longer computation time that the best algorithm in [38]. The best RH MH is outperformed by the other best algorithms in terms of computation time, since the computation time of the rolling horizon framework is fixed to 180 seconds by construction.

For the 180-minute delayed instances of Table 4, the best CE MH and RH MH are significantly faster to compute their best solution compared to the best algorithm in [38] (i.e. RH MILP). Regarding the solution quality, the best RH MH is better than the best CE MH. However, the best algorithm in [38] gives the best average results, in a longer computation time, except for M2.

For the 180-minute disrupted instances of Table 4, the best CE MH outperforms the previously best known algorithm and the best RH MH in terms of both solution quality and time to compute the best solution. This strong improvement is due to the ability of VNS and VNTS to perform multiple simultaneous re-routing actions in the air segments of the TCA for landing aircraft. These re-routing actions are required in order to allow a better landing and take-off aircraft scheduling on the only available runway. The new best known solutions are found partly by the best RH MH and partly by the best CE MH, but the latter algorithm is quicker and has the best average performance in terms of the objective function value.

Table 5 presents the following aggregate comparisons: the best solutions computed via the new metaheuristics versus the best known solutions in [38], the best centralized metaheuristic (CE MH) versus the best algorithm in [38], the best rolling horizon metaheuristic (RH MH) versus the best algorithm in [38].

In Table 5, Column 1 gives the three field instance code [model variant]-[time horizon]-[NOrmal or DISrupted traffic]; Column 2 the type of comparison; Columns 3–5 the number of better solutions obtained by the new metaheuristics, the average reduction of the objective function value (in seconds), the average variation of the time (in seconds) to compute the best solution (we note that a negative variation means that

Table 5: Performance comparison between algorithms for each type of instance

Instance Type	Comparison Type	Num Better Solut	Avg Value Reduction (sec)	Avg Time Variation (sec)	Num Equal Solut	Avg Time Variation (sec)	Num Worse Solut	Avg Value Increase (sec)	Avg Time Variation (sec)
M1-60-NO	Best Solut: MH vs [38]	0	-	-	20	+20.0	0	-	-
	Best Algo: CE MH vs [38]	0	-	-	18	+9.5	2	4.5	+35.2
	Best Algo: RH MH vs [38]	0	-	-	19	+174.0	1	34.0	+151.9
M2-60-NO	Best Solut: MH vs [38]	0	-	-	20	-10.6	0	-	-
	Best Algo: CE MH vs [38]	1	1.0	-125.1	19	-4.6	0	-	-
	Best Algo: RH MH vs [38]	0	-	-	17	+144.6	3	6.7	+81.3
M3-60-NO	Best Solut: MH vs [38]	0	-	-	20	+15.6	0	-	-
	Best Algo: CE MH vs [38]	1	26.0	+45.9	19	+26.6	0	-	-
	Best Algo: RH MH vs [38]	1	26.0	+180.0	17	+172.0	2	6.5	+146.2
M1-180-NO	Best Solut: MH vs [38]	6	5.5	-230.8	11	-183.1	3	11.0	-183.7
	Best Algo: CE MH vs [38]	5	29.8	-169.1	2	-129.8	13	27.2	-199.1
	Best Algo: RH MH vs [38]	6	25.3	-103.3	5	-62.3	9	28.8	-104.7
M2-180-NO	Best Solut: MH vs [38]	4	21.3	-283.1	11	-194.3	5	11.0	-149.0
	Best Algo: CE MH vs [38]	5	24.6	-235.8	5	-282.1	10	14.2	-238.6
	Best Algo: RH MH vs [38]	5	32.2	-172.7	6	-153.3	9	17.3	-176.0
M3-180-NO	Best Solut: MH vs [38]	6	17.5	-324.2	1	-207.9	13	27.8	-243.0
	Best Algo: CE MH vs [38]	4	10.0	-306.4	3	-334.1	13	31.1	-272.5
	Best Algo: RH MH vs [38]	5	15.8	-238.8	1	-207.9	14	32.1	-197.4
M1-180-DIS	Best Solut: MH vs [38]	10	160.4	-345.7	3	-397.6	7	87.4	-476.9
	Best Algo: CE MH vs [38]	13	197.5	-684.7	0	-	7	114.3	-715.3
	Best Algo: RH MH vs [38]	5	136.0	-540.0	7	-540.0	8	120.8	-540.0
M2-180-DIS	Best Solut: MH vs [38]	9	109.7	-571.7	1	-445.1	10	103.6	-523.0
	Best Algo: CE MH vs [38]	9	187.4	-683.1	0	-	11	110.2	-676.8
	Best Algo: RH MH vs [38]	3	291.3	-540.0	6	-540.0	11	210.2	-540.0
M3-180-DIS	Best Solut: MH vs [38]	20	96.6	-535.0	0	-	0	-	-
	Best Algo: CE MH vs [38]	13	162.6	-695.7	0	-	7	113.9	-679.3
	Best Algo: RH MH vs [38]	5	223.2	-540.0	4	-540.0	11	164.6	-540.0

the new metaheuristics are faster, while a positive variation means that the new metaheuristics are slower); Columns 6–7 the number of equal solutions obtained by the new metaheuristics, the average variation of the best solution computation time (in seconds); Columns 8–10 the number of worse solutions obtained by the new metaheuristics, the average increase of the objective function value (in seconds), the average variation of the best solution computation time (in seconds).

For the 60-minute instances, the number of equal solutions is very large for all models and there is a small time variation in computing the best known solutions by the different approaches. This is related to the relative low complexity of the 60-minute delayed instances, and to the fact that the best known solutions in [38] were already optimal. Furthermore, the best algorithms in [38] are often very fast to compute the optimal solution, leaving a relatively small margin for further improving the computational performance. On average, the new metaheuristics take a larger computation time. However, the best CE MH presents a small increase of the computation time. Regarding the objective function value, the new metaheuristics compute three better solutions than the best algorithm in [38].

For the 180-minute delayed instances, for 16 out of 60 instances the new metaheuristics compute a new best known solution in a significantly smaller computation time compared to the one required to compute the previously best known solution. For other 23 instances, the best known solution is computed by new metaheuristics again in a smaller computation time compared to [38], while for the remaining instances 180 seconds of computation time are not sufficient to find the best known solution. In general, the main advantage of the new metaheuristics is related to the quickness to compute their best solution.

For the 180-minute disrupted instances, a new best known solution is found by the new metaheuristics in 39 out of 60 cases, the same best known solution is found in 4 cases but with a strongly reduced computation

time, and a worsening solutions is found with 180 seconds of computation in the remaining cases. For each row of the table, the average solution value increase is always smaller than the average solution value reduction. For all cases, a main advantage of the new metaheuristics (specially in the centralized framework) is again a strongly reduced time to compute the best solution, which is an important requirement to fit the real-time application of the proposed methodology even for the most difficult instances.

4.5 Discussion on the obtained results

Looking at the computational results, the main conclusions are next drawn. The new metaheuristics compute good quality solutions in much less time compared to state-of-the-art algorithms for the 180-minute traffic predictions. The results obtained by the new metaheuristics for the 60-minute traffic predictions are optimal, even if, for the latter instances, a slightly larger computation time is required compared with the best algorithms in [38]. A new best known upper bound value is provided for 55 out of the 120 instances for which no optimal solution is yet proven. For the disrupted instances, the new neighbourhoods and metaheuristic schemes, on average, outperform the best results obtained with the TS algorithm (i.e. the only algorithm proposed in [38] that is able to compute a feasible solution for all ATC-TCA instances during the first 180 seconds) within one minute of computation on a standard processor. The VNS and VNTS algorithms improve the results obtained by TS, since they explore multiple simultaneous aircraft re-routing actions and allow additional aircraft re-scheduling flexibility in the air segment and runway resources.

When comparing the centralized versus the rolling horizon frameworks, the metaheuristics are, on average, faster in the centralized framework. However, the solutions provided by the combination of the metaheuristics with the rolling horizon framework help to find new best known solutions for a significant number of instances. The strength of the rolling horizon approach is the problem decomposition in small time horizons, that has the effect of simplifying the aircraft scheduling problem and thus enables a more effective evaluation of the potential routing moves during the neighbourhood search strategy.

The metaheuristics present a different performance for the different types of instances. Regarding the time horizon and type of disturbance, the 180-minute disrupted instances are more difficult to solve compared to the other instances, since the 60-minute instances present a reduced number of variables and the 180-minute delayed instances have a reduced number of deadline constraints. Regarding the model variants, no significant performance difference is found when comparing M1 and M2, even if the two models have a different number of due date constraints. On the contrary, M3 is more difficult to solve than the other models, since there are additional deadline constraints compared to M1 and M2. We recall that the presence of deadline constraints may strongly reduce the set of feasible schedules.

For the instances with deadline constraints, the new metaheuristics are performing very well compared to the approaches in [38], since these complex traffic situations require to perform simultaneous aircraft re-routing actions frequently in order to find good quality solutions. Another important reason is the use of the deadline implications and the pre-processing procedure developed in [38], that help the metaheuristics to find good quality solutions and to compute better solutions than the commercial MILP solver.

5 Conclusions and future research

This paper proposes fast scheduling and routing metaheuristics for air traffic control at a busy TCA, considering aircraft sequencing, assignment of resources (routing) and timing of operations, with particular focus on the efficient control of strong traffic disturbances (such as multiple aircraft delays and a temporarily disrupted runway). To this end, several algorithmic innovations are considered, which relate to design of effective metaheuristics based on a decomposition of the problem into scheduling and routing decisions. A new set of neighbourhoods and new metaheuristic schemes are proposed on the basis of the hybridization between a variable neighbourhood search and a tabu search. These algorithms are furthermore combined with a rolling-horizon based decomposition framework.

The algorithms and frameworks are evaluated on an Italian practical case study, i.e. the TCA of Milan Malpensa, and on a restricted group of instances found to be the most challenging in [38]. Those instances are among the most complex instances in the literature of the ATC-TCA problem, including a large number of aircraft and various sources of disturbances. We also compared the performance of the various approaches for three model variants, with differences in the set of constraints.

The new metaheuristics are benchmarked against state-of-the-art ATC-TCA approaches which include optimization algorithms developed in the AGLIBRARY solver and an MILP formulation for simultaneous aircraft scheduling and routing solved by a commercial solver. The algorithms proposed in this paper, with new neighbourhood structures and search strategies, improve the effectiveness of the previously developed approaches. From the computational experiments, the main contributions are next summarized: a general significant reduction of the time required to compute good quality solutions; a better performance in terms of solution quality and computation time, especially for the most difficult instances including disruptions; and the computation of new best known solutions for the several instances for which the optimal solution is not yet proven.

The average computational behaviour of the new metaheuristics is also analyzed in depth, resulting in a slightly slower descent compared to the TS (the best performing algorithm in [38] during the first three minutes of computation) in the first 20 seconds. However, at around 1 minute of computation the new metaheuristics offer the possibility to strongly reduce the objective function value computed by TS.

Future efforts should be dedicated to a path towards the implementation of advanced ATC-TCA approaches into a dynamic air traffic management system. The optimization models, algorithms and frameworks presented in this work should be a core component of the intelligent transport system. Other issues are worthwhile being investigated, including the development of good quality lower bounds for the ATC-TCA problem, the extension of our methodology to better deal with aircraft speed variations in the arriving and departing procedures, the integration and coordination with large-scale air traffic control measures.

References

[1] Alonso-Ayuso, A., Escudero, L.F., Martín-Campo, F.J., Mladenović, N. (2015) A VNS metaheuristic for solving the aircraft conflict detection and resolution problem by performing turn changes. *Journal of Global Optimization* **63 (3)** 583–596.

[2] Atkin, J.A.D., Burke, E.K., Greenwood, J.S., Reeson, D. (2007) Hybrid metaheuristics to aid runway scheduling at London Heathrow airport. *Transportation Science* **41 (1)** 90–106.

[3] Ball, M., Barnhart, C., Nemhauser, G., Odoni, A. (2007) Air transportation: Irregular operations and control. *Handbooks in Operations Research and Management Science* **14(1)** 1–67.

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

[5] Barnhart, C., Fearing, D., Odoni, A., Vaze, V. (2012) Demand and capacity management in air transportation. *EURO Journal of Transportation and Logistics* **1(1–2)** 135–155.

[6] Beasley, J.E., Sonander, J., Havelock, P. (2001) Scheduling aircraft landings at London Heathrow using a population heuristic. *Journal of the Operational Research Society* **52 (1)** 483–493.

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

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

[9] Corman, F., D'Ariano, A., Pacciarelli, D., Pranzo, M. (2010) A tabu search algorithm for rerouting trains during rail operations. *Transportation Research – Part B* **44 (1)** 175–192.

[10] 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.

[11] D'Ariano, A., Pacciarelli, D., Pistelli, M., Pranzo, M. (2015). Real-time scheduling of aircraft arrivals and departures in a terminal maneuvering area. *Networks* **65 (3)** 212–227.

[12] D'Ariano, A., Pistelli, M., Pacciarelli, D. (2012) Aircraft retiming and rerouting in vicinity of airports. *IET Intelligent Transport Systems Journal* **6(4)** 433–443.

[13] Dear, R.G. (1976) *The dynamic scheduling of aircraft in the near terminal area.* Flight Transportation Laboratory (FTL) Report R76-9, Massachusetts Institute of Technology.

[14] Eurocontrol (2011) Airport collaborative decision-making. Accessed from *http://www.euro-cdm.org/*

[15] Faye, A. (2015) Solving the aircraft landing problem with time discretization approach. *European Journal of Operational Research* **242 (3)** 1028–1038.

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

[17] Glover, F. (1986) Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research* **13 (5)** 533–549.

[18] Hancerliogullari, G., Rabadi, G., Al-Salem, A.H., Kharbeche, M. (2013) Greedy algorithms and metaheuristics for a multiple runway combined arrival-departure aircraft sequencing problem. *Journal of Air Transport Management* **32** 39–48.

[19] Hansen, P., Mladenović, N., Moreno Pérez, J.A. (2008) Variable neighbourhood search: Methods and applications. *4OR – Quarterly Journal of Operations Research* **6 (4)** 319–360.

[20] Hu, X.B., Chen, W.H. (2005) Genetic algorithm based on receding horizon control for arrival sequencing and scheduling. *Engineering Applications of Artificial Intelligence* **18 (5)** 633–642.

[21] Hu, X.-B., Di Paolo, E. (2008) Binary-representation-based genetic algorithm for aircraft arrival sequencing and scheduling. *IEEE Transactions on Intelligent Transportation Systems* **9 (2)** 301–310.

[22] Hu, X.B., Di Paolo, E. (2009). An efficient genetic algorithm with uniform crossover for air traffic control. *Computers and Operations Research* **36 (1)** 245–259.

[23] Jiang, Y., Xu, Z., Xu, X., Liao, Z., Luo, Y. (2014) A schedule optimization model on multirunway based on ant colony algorithm. *Mathematical Problems in Engineering*, Volume 2014, 11 pages, http://dx.doi.org/10.1155/2014/368208

[24] Kohl, N., Larsen, A., Larsen, J., Ross, A., Tiourine, S. (2007) Airline disruption management – Perspectives, experiences and outlook. *Journal of Air Transport Management* **13 (3)** 149–162.

[25] Kuchar, J.K., Yang, L.C. (2000) A review of conflict detection and resolution modeling methods. *IEEE Transactions on Intelligent Transportation Systems* **4 (1)** 179–189.

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

[27] Moreno Pérez, J.A., Moreno-Vega, J.M., Rodríguez Martín, I. (2003). Variable neighborhood tabu search and its application to the median cycle problem. *European Journal of Operational Research* **151 (2)** 365–378.

[28] Murça, M.C.R., Müller, C. (2015) Control-based optimization approach for aircraft scheduling in a terminal area with alternative arrival routes. *Transportation Research – Part E* **73 (1)** 96–113.

[29] Nosedal, J., Piera, M.A., Solis, A.O., Ferrer, C. (2015) An optimization model to fit airspace demand considering a spatio-temporal analysis of airspace capacity. *Transportation Research – Part C* **61 (1)** 11–28.

[30] Pacciarelli, D., Pranzo, M. (2003) Production scheduling in a steelmaking-continuous casting plant. *Computers & Chemical Engineering* **28 (12)** 2823–2835.

[31] Pellegrini, P., Rodriguez, J. (2013) Single European sky and single European railway area: A system level analysis of air and rail transportation. *Transportation Research – Part A* **57 (1)** 64–86.

[32] Pinol, H., Beasley, J.E. (2006) Scatter search and bionomic algorithms for the aircraft landing problem. *European Journal of Operational Research* **171 (2)** 439–462.

[33] Psaraftis, H.N. (1978) *A dynamic programming approach to the aircraft sequencing problem.* Flight Transportation Laboratory (FTL) Report R78-4, Massachusetts Institute of Technology.

[34] Salehipour, A., Modarres, M., Moslemi Naeni, L. (2013) An efficient hybrid meta-heuristic for aircraft landing problem. *Computers and Operations Research* **40 (1)** 207–213.

[35] Salehipour, A., Moslemi Naeni, L., Kazemipoor, H. (2009) Scheduling aircraft landings by applying a variable neighborhood descent algorithm: Runway-dependent landing time case, *Journal of Applied Operational Research* **1 (1)** 39–49.

[36] Samà, M., D'Ariano, A., Corman, F., Pacciarelli, D. (2015) Metaheuristics for efficient aircraft scheduling and re-routing at busy terminal control areas. Technical Report RT-DIA-213-2015, pages 1–41, Roma Tre University, Department of Engineering, Section of Computer Science and Automation, Rome, Italy.

[37] Samà, M., D'Ariano, A., Pacciarelli, D. (2013) Rolling horizon approach for aircraft scheduling in the terminal control area of busy airports. *Transportation Research – Part E* **60 (1)** 140–155.

[38] Samà, M., D'Ariano, A., D'Ariano, P., Pacciarelli, D. (2014) Optimal aircraft scheduling and routing at a terminal control area during disturbances. *Transportation Research – Part C* **47 (1)** 61–85.

[39] Samà, M., D'Ariano, A., D'Ariano, P., Pacciarelli, D. (2016) Scheduling models for optimal aircraft traffic control at busy airports: Tardiness, priorities, equity and violations considerations, *Omega*, DOI: 10.1016/j.omega.2016.04.003

[40] SESAR (2015) The roadmap for delivering high performing aviation for Europe: European ATM Master Plan. Accessed from *http://ec.europa.eu/transport/modes/air/sesar/doc/eu-atm-master-plan-2015.pdf*

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

[42] Soomer, M.J., Koole, G.M. (2008) *Fairness in the aircraft landing problem.* Technical Report, Department of Mathematics, Vrije Universiteit Amsterdam, The Netherlands.

[43] Solveling, G., Solak, S., Clarke, J.-P., Johnson, E. (2011) Scheduling of runway operations for reduced environmental impact. *Transportation Research – Part D* **16 (2)** 110–120.

[44] Zhang, Y. (2008) *Real-time inter-modal strategies for airline schedule perturbation recovery and airport congestion mitigation under Collaborative Decision Making (CDM).* University of California Transportation Center, UCTC Dissertation No. 154, USA.

[45] 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 Transactions on Intelligent Transportation Systems* **11 (2)** 399–412.