The aircraft runway scheduling problem: A survey

Sana Ikli, Catherine Mancel, Marcel Mongeau, Xavier Olive, Emmanuel Rachelson

The aircraft runway scheduling problem: A survey

Sana Ikli[a,*], Catherine Mancel[a], Marcel Mongeau[a], Xavier Olive[b], Emmanuel Rachelson[c]

[a] *ENAC, Université de Toulouse, France*
[b] *ONERA DTIS, Université de Toulouse, France*
[c] *ISAE-SUPAERO, Université de Toulouse, France*

Abstract

The aircraft scheduling problem consists in sequencing aircraft on airport runways and in scheduling their times of operations taking into consideration several operational constraints. It is known to be an NP-hard problem, an ongoing challenge for both researchers and air traffic controllers.

The aim of this paper is to present a focused review on the most relevant techniques in the recent literature (since 2010) on the aircraft runway scheduling problem, including exact approaches such as mixed-integer programming and dynamic programming, metaheuristics, and novel approaches based on reinforcement learning. Since the benchmark instances used in the literature are easily solved by high-performance computers and current versions of solvers, we propose a new data set with challenging realistic problems constructed from real-world air traffic.

Keywords: Runway scheduling problem, Aircraft landing problem, Aircraft take-off problem, Mixed-integer optimization, Metaheuristics, Survey.

1. Introduction

Airport runways are recognized to be one major bottleneck of the Air Traffic Management (ATM) system (acronyms meaning are also outlined in Table C.10), and one of the key factors that determine airport capacity.

*Corresponding author
Email addresses: sana.ikli@recherche.enac.fr (Sana Ikli), catherine.mancel@enac.fr (Catherine Mancel), marcel.mongeau@enac.fr (Marcel Mongeau), xavier.olive@onera.fr (Xavier Olive), emmanuel.rachelson@isae-supaero.fr (Emmanuel Rachelson)

Building new infrastructure (runways, airports) to alleviate the congestion problem at the runway is not always a possible recourse because of the investment cost (*e.g.*, building a new runway in Dublin airport costs 320 million euros [1]). An alternative is to optimize the utilization of existing infrastructures through an improved planning.

Several researchers have been interested in optimizing the utilization of runways. This aircraft runway scheduling problem is generally split into two types of problems, involving either landings or take-offs, which come with specific physical constraints (aircraft can stop before take-off contrary to aircraft that are about to land), and are referred to in the literature as the *Aircraft Landing Problem* (ALP) and the *Aircraft Take-off Problem* (ATP).

Generally speaking, the ALP (respectively ATP) consists in first assigning an available runway to each aircraft ready to land (take off), then allocating a scheduled landing (take off) time to each. When we consider the sequencing and scheduling of both landings and take-offs, the problem is referred to as the *Aircraft Scheduling Problem* (ASP). The latter problem is more realistic, since air traffic controllers simultaneously deal with arriving and departing air traffic.

In current day-to-day operations, solving the ALP consists of three steps: first an initial schedule is created in a first-come first-served fashion. Then, it is modified during the so-called *approach* phase, and finally frozen as aircraft reach the *final approach* phase. The initial schedule includes aircraft within the radar range of the airport's landing planer, which corresponds to a time horizon of about 40 minutes prior to landing. An update process is performed each time a new aircraft enters the radar range so as to improve the previous landing schedule [2].

For the ATP, flow management authorities such as the Network Manager Operations Centre (NMOC) in Europe – sometimes also referred to as the Central Flow Management Unit (CFMU) in previous surveys – assign take-off slots to departing aircraft. The assigned time slot is a Calculated Take-Off Time Window (CTOTW) defined usually by a target time, denoted T, and a 15-minute time range [3], *i.e.*, the interval $[T-5, T+10]$.

In the mathematical formulation of the ALP/ATP, several operational constraints must be taken into consideration. The most common requirements include the safety separation between consecutive aircraft, allowed time windows defined by an earliest and a latest times of operation based on fuel considerations, and precedence constraints. Various objective functions can be considered depending on the decision maker, which can be an airport,

an airline company, a government, etc. The most common stakeholder point of view considered in the literature is that of the airport, which typically seeks at increasing runway capacity, or the viewpoint of the airline company, which aims at maximizing punctuality and minimizing fuel consumption. An exhaustive list of the objective functions considered in the literature can be found in [2].

In the remainder of this paper, the acronym RSP (Runway Scheduling Problem) is used to designate the ALP, the ATP or the ASP; we shall specify which problem is concerned when relevant.

The RSP is known to be NP-hard [4]. Therefore, its solution time by exact methods does not scale well with the problem size, here the number of aircraft. The RSP represents an ongoing challenge for air traffic controllers because they will have to deal with an increasing number of aircraft.

Since the publication of the first approach for solving the RSP in 1976 by Dear [5], several models and solution approaches have emerged in the literature. Bennell *et al.* [2] proposed in 2010 a comprehensive overview of the approaches from the literature. Recently, Veresnikov *et al.* published two surveys on the ALP [6, 7]. The first one focuses on some exact approaches to the ALP (mainly mixed-integer programming), and the second one overviews mainly genetic and memetic algorithms. The common aspects between the two above-mentioned surveys is that they are focused on theoretical aspects of the ALP. For more general airport operation problems, a short review is proposed in [8].

In this paper, we present a comprehensive review of state-of-the-art solution approaches in the literature to the three problems ALP, ATP, and ASP. We concentrate not only on exact methods and metaheuristics, but we also review two novel approaches based on Reinforcement Learning (RL), which appear to be promising approaches for runway scheduling. We also highlight the analogy between the RSP and well-known combinatorial optimization problems, such as the traveling salesman problem and the vehicle routing problem. Moreover, via a comparative study, we show how benchmark instances used in the literature are no longer challenging for current versions of solvers, because they can be solved optimally in reasonable computation times. Therefore, we provide new data sets of challenging problems which feature more realistic aspects of runway scheduling, and are constructed from real-world traffic.

Our goal is to provide both the operational research community and the ATM practitioners with an up-to-date synthesis on the RSP, with common

notions and vocabulary, as well as common test instances and a general view of the literature, providing thereby a common ground with new insights on the RSP. Based on this view of the literature, promising future research directions are derived, to address the limitations of current works on the matter, and to consider more realistic aspects of this problem.

The remainder of this paper is organized as follows. Section 2 overviews the basic aeronautical concepts related to the RSP. Section 3 first briefly presents the most cited model in the literature, that of Beasley *et al.* [9]. The analogies of the RSP with classical combinatorial optimization problems are highlighted. Sections 5 and 6 present an overview of exact and stochastic solution approaches (respectively) to the RSP, with a focus on recent contributions in the literature. Finally, in Section 7 conclusions and promising future research directions are suggested.

2. Aeronautical background

From take-off to landing and through the controlled airspace, aircraft are guided by air traffic controllers, whose duty is to ensure a safe *separation* (see below) between aircraft and an orderly traffic flow. Since safety rules impose that the runway can be utilized by only one aircraft at a time, departing and arriving aircraft near large airports are sequenced according to criteria such as equity (limited deviation from the first-come first-served order) and runway capacity [10].

The aim of this section is to define the basic concepts related to the runway scheduling, starting with the most important constraints that must be taken into consideration, until the basic techniques used by controllers to sequence aircraft. Finally, we will briefly present the decision-support tools available to assist controllers in sequencing and managing aircraft. Readers may refer to [11] for further details on the aeronautical concepts related to the RSP or to ATM in general.

2.1. Separation

The separation requirements are one of the factors that determine runway capacity, in terms of maximum number of aircraft that can use the runway system during a specified time interval. According to [10], the most commonly used separation standards are the following:

- The *radar separation* is a minimal pairwise distance separation that must be ensured between aircraft under radar control areas. The International Civil Aviation Organization (ICAO) specifies the minimum vertical separation for aircraft flying under *Instrument Flight Rules* (IFR) as 1000 ft (300 m) below flight level 290 (altitude 29000 ft), and 2000 ft (600 m) at or above this level [12]. If any two aircraft are separated by less than the minimum vertical separation, a longitudinal separation is required. The ICAO sets this longitudinal separation at 5 nautical miles (nm; 1 nm = 1852 meters), but it can be reduced to 3 nm in congested areas [12].

- The *Wake Vortex* (WV) *separation* ensures that no aircraft is affected by the *wake-vortex turbulence* generated by a leading aircraft, especially during take offs and landings. Depending on their *maximum take-off mass*, the ICAO classifies aircraft into three main categories: *Heavy* (H), *Medium* (M) and *Light* (L)[1]. The separation requirements between two consecutive aircraft depend on the wake turbulence category of the leading aircraft and that of the trailing aircraft. The WV separation is therefore sequence dependent. The WV separation between departures is commonly expressed in time units, while between landings, it is given in terms of distance, but can be converted to a time separation using representative approach speeds, as explained in Appendix A.

Today's ICAO WV separation rules are outdated and considered too conservative. For this reason, EUROCONTROL and the Federal Aviation Administration (FAA) conducted research to redefine wake turbulence categories and their associated separation. RECAT (*i.e.*, Re CATegorization) is the FAA's program that refines the ICAO's classical categories into six new categories. It has the objective of defining a pairwise separation that is continuously updated based on real-time information such as weather conditions and aircraft-derived data [13].

In practice, other factors may play a role in the application of separation between aircraft, *e.g.*, weather conditions, airport runway configuration, specific local departure and approach route structures, etc. For instance, in Milan–Malpensa airport, the two parallel runways 17R/35L and 17L/35R are

[1]The ICAO classification includes also the *Super Heavy* category, which only contains Airbus A380-800.

used both for take offs and landings. The final approach separation to runway 17L/35R is the ICAO's distance separation given in Table A.5. On the other hand, for runway 17R/35L, the minimum separation between two landings is always 6 nm, regardless of the type of the aircraft involved, because it takes longer time for aircraft to free this runway after landing [14].

2.2. Sequencing techniques

The *First-Come First-Served* (FCFS) rule is a sequencing technique that schedules aircraft based on the *Estimated Time of Arrival* (ETA) and the *Estimated Take-off Time* (ETT) associated to each aircraft. For arriving aircraft, a FCFS sequencer computes the Scheduled Landing Times (SLT) based on the order given by the ETA – obtained when an aircraft enters the airport's radar – and taking into consideration the constraints imposed by the runway separation requirements [15]. Similarly, for departing aircraft, the sequencer computes the Scheduled Take-off Time (STT) based on the ETT – which is given by the order of aircraft queueing at the airport holding positions – and the separation requirements. The FCFS heuristic is widely used in practice, since it is easy to implement, requires little controller workload, and guarantees equity between aircraft. The drawback of this technique is that, in congested airports, it rarely provides optimal sequences in terms of runway throughput or average delay, due to the WV separations.

The *Constrained Position Shifting* (CPS) is a concept first introduced in 1976 in the PhD thesis of Dear [5]. It allows aircraft to deviate from the nominal FCFS order up to a maximum number of position shifts. For instance, if an aircraft occupies the 5^{th} position in a landing sequence, and the maximum number of position shifts allowed is equal to 2, then this aircraft can only be re-scheduled in positions 3, 4, 5, 6 or 7. The CPS has the double advantage of being a reasonable constraint in order to limit inequity and of reducing drastically the solution time of exact scheduling techniques. Although restricting the set of possible sequences it considers, the CPS can considerably improve the FCFS sequence and increase runway throughput, as we can see in Figure 2.2, because it allows to avoid undesirable landing sequences, in which a "Heavy" aircraft is followed by a "Light" aircraft, that requires the largest WV separation of 6 nm, *i.e.*, about 196 seconds. Several models presented in the literature [4, 16, 17, 18, 19] rely on the CPS.

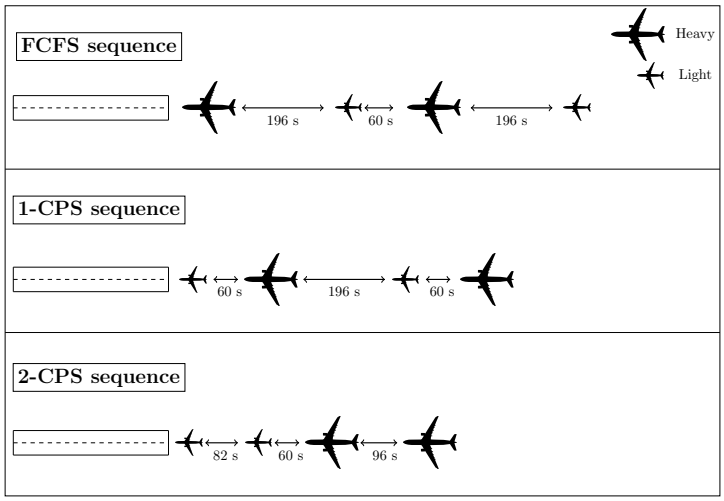
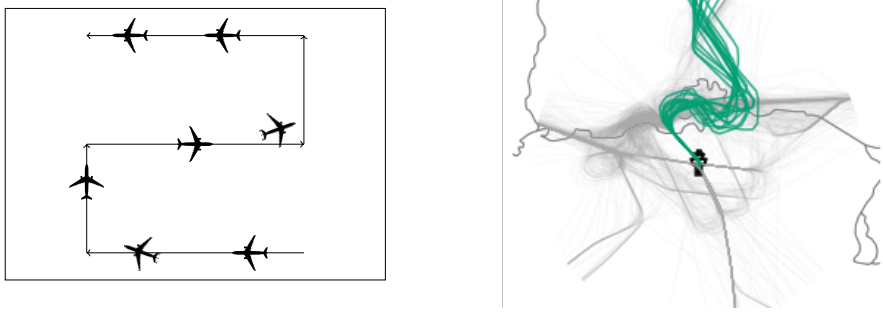


Figure 1: Comparison of three landing sequences using the required time separations of Table A.7: the FCFS sequence that requires a total of 452 seconds to land the four aircraft in the sequence, the FCFS sequence with a maximum number of position shifts equal to 1 (1-CPS sequence) that requires 316 seconds, and the FCFS order with maximum position shifts equal to 2 (2-CPS sequence) that requires 238 s

2.3. Holding techniques

Sequencing and scheduling models aims at providing scheduled landing (or take off respectively) times optimizing some given objective function while satisfying the above-mentioned separation constraints. These models typically suppose that the aircraft to be scheduled are somehow "waiting" in the air (on the ground respectively) and able to land (take off) at anytime. However, when unpredictable delays occur, it may be difficult to respect the initial schedule. Thus, some aircraft may need to *hold* before landing (or taking off) [20]. This is particularly critical to the ALP where, contrary to aircraft waiting for taking off, air traffic controller can not stop an aircraft. A feature of several arrival routes in the airspace surrounding airports around the world in which aircraft can hold is the so-called *trombone* arrival routes, illustrated in Figure 2. They are predefined cycling tracks in the arrival routes that some aircraft are asked to follow until they receive a clearance from air traffic controllers. According to [21], this structure allows a fluid and efficient traffic flow to the runway.

In [2], other techniques used by controllers to delay aircraft are presented



(a) Schematic representation (Based on [21]) (b) Zurich airport

Figure 2: A schematic representation of the trombone 2a, and a real-world illustration from real traffic in Zurich airport 2b.

such as, Vector For Space (VFS), which is a holding technique that stretches the path of an aircraft instead of letting it flying the direct path between two points, and Holding Patterns (HP) that are waiting loops on different flight levels designed to delay aircraft by constant delay (typically 4 minutes to accomplish one loop). Since the HP delays aircraft by a chosen multiple of a constant time (4 minutes), it is appropriate for inducing substantial delays (at the expense of extra fuel consumption).

For departures, aircraft can be held at specific points on the ground called *holding positions*.

2.4. Decision-support tools

There are a number of decision-support tools available to assist controllers in managing arriving and/or departing flows. One of these tools is the *Center-TRACON Automation System* (CTAS) developed by NASA and the FAA. It consists of three sub-tools, namely the *traffic management advisor* that provides runway assignment and scheduled landing times for arriving aircraft, the *descent advisor* that assists controllers in guiding aircraft to *metering fixes*[2], and the *final approach sequencing* tool that provides speed and heading recommendations. The CTAS has a component referred to as the *expedite departure path* that helps managing departing aircraft. Readers may refer for

[2]Metering fixes: specific points along an established air route over which aircraft will be metered prior to entering the terminal airspace surrounding airports.

instance to [22] for more details on integrated tools in CTAS.

The European analogue of CTAS are the *Arrival MANager* (AMAN) tools. They aim at assisting controllers in guiding arriving flows in terminal areas to specific points, such as the runways threshold or metering fixes. Basic AMAN versions provide a landing sequence, a timeline with the view at runway threshold, and a target landing time for each aircraft. More advanced versions can provide advanced control actions such as radar vectoring (altitude, speed, and heading control). Unlike the CTAS, the AMAN tools cannot provide *conflict* (*i.e.*, violated separation) detection and resolution strategies, which may result in an increased pressure on controllers managing dense air spaces [11]. There exists a version of AMAN called *Extended AMAN* (E-AMAN), whose considered horizon is extended up to 500 nm from the airport, instead of around 100 to 200 nm for current AMAN [23]. The motivation for such an extension is to begin sequencing much earlier so as to attempt at reducing congestion, noise, and fuel burn in the airspace near airports.

The *Departure MANager* (DMAN) is the European tool used for managing and merging departure flows into the en-route traffic. The DMAN tools did not receive as much attention as the AMAN, and it is considered to be less mature than the latter one.

3. Mathematical Models

In this section, we first describe the RSP, then we present the most cited model in the literature, which is the mathematical formulation of [9]. The analogy between the RSP and some classical combinatorial optimization problems is also highlighted in this section.

Consider a set of runways $\mathcal{K} = \{1, 2, ..., m\}$, and a set of aircraft denoted $\mathcal{A} = \{1, ..., n\}$, ready to land (respectively to take off). Each aircraft $i \in \mathcal{A}$ has a pre-defined time window denoted $[E_i, L_i]$, and a possible preferred time of *operation* (landing or take-off) denoted T_i.

In its most basic versions, the RSP consists in first assigning an available runway $k \in \mathcal{K}$ and a scheduled time of operation, denoted x_i for each aircraft $i \in \mathcal{A}$, subject to operational constraints, mostly the WV separation to ensure safety, and time-window restrictions to prevent excessive delays. Additional constraints may also be considered, such as the CPS presented in Section 2.2. Several objective functions may be considered according to the decision-maker priorities. Readers may refer to [2] for a complete list of

relevant objective functions according to whether the decision maker is an airport, an airline, or a government.

Several mathematical formulations are proposed in the literature to model the RSP, most of them are Mixed-Integer Programming (MIP) formulations. Recently, the Markov Decision Process (MDP) framework has been used to model the ATP [24], and the ALP [25]. A comprehensive review of these two approaches is presented in Section 6. The mathematical formulations proposed in the literature can be classified according to:

- The **availability of the input data**; the model is said to be *static* when all the input data are known in advance, or *dynamic* when some inputs are unknown for the considered time horizon.
- The **uncertainty on the parameters values**; the model is then said *deterministic* or *under uncertainty* accordingly.
- The **number of runways and their configuration**; one distinguishes then the single-runway case from the multiple-runway model. In the latter case, several configuration (*i.e.*, layout or design) are possible, such as the parallel (two or more) runways, whose centerlines are parallel. A comprehensive list of articles regrouped by the number of runways and their configuration is reported in [26].
- The **objective function**; one may maximize runway throughput (number of operations per hour), minimize maximum (over all aircraft) delay, minimize weighted deviations from target times, etc.
- The **constraints** taken into account; the fundamental constraints considered are related to the safety separation, but one may also add time windows, the CPS, or precedence constraints.

3.1. Mixed-Integer Programming formulations

Beasley *et al.* [9] present the most cited MIP model. The problem considered is the ALP with multiple runways, but the model can be easily adapted to incorporate take offs. The majority of MIP models proposed in the literature are based on this formulation.

Input data For each aircraft $i \in \mathcal{A}$, the input parameters are presented in Table 1.

Notation	Parameter
◦ T_i	◦ Target/preferred landing time.
◦ $[E_i, L_i]$	◦ Landing time window ($L_i > E_i$).
◦ c_i^-	◦ Penalty cost (≥ 0) per time unit for landing before T_i.
◦ c_i^+	◦ Penalty cost (≥ 0) per time unit for landing after T_i.
◦ S_{ij}	◦ The WV separation between aircraft i and j, when i and j land on the same runway (i lands before j).
◦ s_{ij}	◦ The required separation (see [26]) between aircraft i and j, when i and j land on different runways (i lands before j).

Table 1: Input parameters of Beasley's model [9].

Decision variables The proposed formulation involves both binary and continuous variables. The binary variables, for the runway assignment and for sequencing, are defined as follows:

- $a_{ik} = \begin{cases} 1 & \text{if aircraft } i \text{ is assigned to runway } k, \\ 0 & \text{otherwise,} \end{cases}$

  $i \in \mathcal{A}, k \in \mathcal{K}$,

- $z_{ij} = \begin{cases} 1 \text{ if aircraft } i \text{ and } j \text{ are assigned to a same runway,} \\ 0 \text{ otherwise,} \end{cases}$

  $i, j \in \mathcal{A} : i \neq j$,

- $\delta_{ij} = \begin{cases} 1 \text{ if aircraft } i \text{ lands before } j, \\ 0 \text{ otherwise,} \end{cases}$

  $i, j \in \mathcal{A} : i \neq j$.

The continuous variables, for assigning times at runway threshold, are defined as follows for each $i \in \mathcal{A}$:

- x_i : landing time,
- x_i^-, x_i^+ : deviations from target times (before and after T_i respectively),

In fact, x_i^- and x_i^+ are introduced in order to linearize the objective function; these variables are understood as: $x_i^- = \max(0, T_i - x_i)$ and $x_i^+ = \max(0, x_i - T_i)$.

Complete formulation

The complete formulation of [9] is:

$$\underset{a,z,\delta,x,x_i^-,x_i^+}{\textbf{minimize}} \quad \sum_{i\in\mathcal{A}} c_i^- x_i^- + c_i^+ x_i^+ \tag{1}$$

subject to

$$E_i \le x_i \le L_i \quad i \in \mathcal{A} \tag{2}$$
$$0 \le x_i^- \le T_i - E_i \quad i \in \mathcal{A} \tag{3}$$
$$0 \le x_i^+ \le L_i - T_i \quad i \in \mathcal{A} \tag{4}$$
$$x_i^- \ge T_i - x_i \quad i \in \mathcal{A} \tag{5}$$
$$x_i^+ \ge x_i - T_i \quad i \in \mathcal{A} \tag{6}$$
$$x_i = T_i - x_i^- + x_i^+ \quad i \in \mathcal{A} \tag{7}$$
$$\delta_{ij} + \delta_{ji} = 1 \quad i,j \in \mathcal{A} : i \ne j \tag{8}$$
$$\sum_{k\in\mathcal{K}} a_{ik} = 1 \quad i \in \mathcal{A} \tag{9}$$
$$z_{ij} = z_{ji} \quad i,j \in \mathcal{A} : i < j \tag{10}$$
$$z_{ij} \ge a_{ik} + a_{jk} - 1 \quad i,j \in \mathcal{A} : i < j,\ k \in \mathcal{K} \tag{11}$$
$$x_j \ge x_i + S_{ij} z_{ij} + s_{ij}(1 - z_{ij}) - M\delta ji \quad i,j \in \mathcal{A} : i \ne j \tag{12}$$
$$a_{ik}, z_{ij}, \delta_{ij} \in \{0,1\} \quad i,j \in \mathcal{A} : i \ne j, k \in \mathcal{K} \tag{13}$$

The chosen objective function (1) aims at minimizing the sum of weighted deviations from target times. Minimizing total delay is a particular case of this objective function, and can be achieved by only considering the latest landing. Constraints (2) represent the time-window restrictions on landing times. Constraints (3) and (4) restrict the maximum possible advance and delay respectively. Constraints (5)-(7) represent the relation between the continuous variables. Constraints (8) express the precedence relationship between aircraft. Constraints (9) ensure that each aircraft lands on exactly one runway. Constraints (10) are the symmetry constraints for aircraft landing on the same runway. Constraints (11) ensure that, if two aircraft i and j land on the same runway k, then z_{ij} is forced to be one; also if aircraft i and j do not land on the same runway, *i.e.*, $z_{ij} = 0$, then a_{ik} or a_{jk} is forced to be zero. Constraints (12) guarantee the WV separation (S_{ij}) between aircraft landing on the same runway, and the mandatory separation (s_{ij}) between aircraft landing on different runways. Finally, constraints (13) ensure the binary restrictions.

In the original formulation of [9], the set of all possible pairs, $i, j \in \mathcal{A}$, of aircraft is partitioned into three sets according to their relative time-window positions; the separation constraints (12) are then specifically expressed for each set (which we do not include here to simplify the presentation of the model).

The above formulation has a weak continuous relaxation [10]. In order to enhance it, several techniques are presented in the literature, such as:

- Optimal *big-M values*. Instead of using an arbitrary large value for the so-called big-M parameter involved in the classical way to model logical constraints, it is recommended to choose it as small as possible (to avoid the well-known convergence problems), and tailored to each particular pair $(i, j) \in \mathcal{A} \times \mathcal{A} : i \neq j$ of aircraft. This can be achieved by choosing $M(i, j) = L_i + S_{ij} - E_j$.

- *Variable-Fixing Strategies*. This is based on the Earliest Landing-time Windows (ELW) rule presented in [27] (Lemma 1). It consists in fixing the order of some aircraft that belong to the same *class* (the WV category for instance) following the non-decreasing lexicographic order of their time windows (*e.g.*, Prakash *et al.* [4]). If the CPS constraints are imposed, further variable values can be fixed based on these constraints.

- Additional *Valid Inequalities*, that may be redundant, can prove to be useful to strengthen the MIP relaxation (*cf.* [9]).

3.2. Analogies and complexity

The ALP/ATP or more generally the ASP, is similar to a number of well-known combinatorial optimization problems, such as the job-shop scheduling problem, the Traveling Salesman Problem TSP), and the Vehicle Routing Problem (VRP).

The relation between a job-shop scheduling and a basic ALP is described in [9]. The runways are interpreted as the machines, and the aircraft to be sequenced are the jobs. If job j follows job i, the sum of the processing time of job i and the sequence-dependent set-up time between i and j corresponds to the minimum WV separation between two landings i and j. The earliest (respectively latest) landing time usually considered in the ALP is interpreted as the release date (respectively due date). A typical objective function minimizes the landing time of the last aircraft in the sequence (*makespan*).

In this case, and considering a single runway, the ALP is equivalent to the TSP, which is known to be NP-hard [28].

In a more realistic context, [14] shows that scheduling aircraft landings and take-offs in terminal control areas, taking into consideration runway configuration as well as the structure of inbound and outbound routes, is analogous to a no-wait job-shop scheduling problem. In this context, each approach (or departure) route is decomposed into smaller flight segments of about 5 nautical miles length, and these flight segments, additional to runways, correspond to machines. Arriving (or departing) aircraft correspond to jobs to be scheduled, while as before, safety separations between aircraft pairs correspond to the sequence dependent set-up times. The authors suppose that every aircraft has earliest landing/take-off times, which again correspond to the release dates.

The analogy between the ALP and the VRP can be derived from [27] as follows. Runways correspond to vehicles to dispatch, aircraft correspond the customers to serve, and the separation time between pairs of aircraft corresponds to the distance between two customers, which is in this case asymmetric, since the separation times between aircraft for both landings and take offs are asymmetric. Target landing times correspond to times at which customers prefer to be served. Lower and upper bounds on these preferred times correspond to the classical time-window constraints. The objective is to serve (to land) each customer (aircraft) within its time window so that the total penalty cost of deviations from target times is minimized.

As a consequence of these reductions, the three problems ALP, ATP, and ASP have the same complexity : they are NP-hard problems. However, if one considers the same scheduling horizon, the ASP is expected to be more complicated to solve, since it involves more traffic to schedule than considering only the ALP (landings) of the ATP (take-offs). Nevertheless, some particular variants of the RSP presented below can be solved in polynomial times. These include:

1. The ALP with aircraft classes [27]. This variant assumes that aircraft can be regrouped in classes (*e.g.*, the WV categories), and that aircraft belonging to the same category are similar, *i.e.*, have same delay cost. The authors show that aircraft can then be sequenced according to the FCFS rule. They propose algorithms that are polynomial in the number of aircraft, but non-polynomial in the number of classes. For simple variants of the ALP involving a single runway and one class, the

proposed algorithm has a complexity $O(n^3)$. However, if one considers three classes as in the ICAO WV categories, the algorithm may not be tractable for a real-world application ($O(n^{23})$), see Theorem 4 in [27].

Regrouping aircraft in classes allows one to define only class-dependent delay costs, instead of aircraft-dependent cost, which may not be realistic, since delay cost depend on features that are proper to each aircraft, such as the number of passengers and the fuel consumption.

2. The ALP with the Constrained Position-Shifting restrictions [16]. Under this CPS restriction, the authors propose a dynamic programming, approach which scales linearly with the problem size. Indeed, for n aircraft and a maximum position shifts of p, the complexity of the proposed algorithm is $O(n(2p+1)^{(2p+2)})$, which is however exponential in p (of little consequence, since the value of p in practice is usually small: 2 or 3).

4. Summary of solution approaches

Several solution approaches are proposed in the literature to solve the RSP, ranging from exact approaches, such as dynamic programming and mixed-integer programming, to stochastic approaches including metaheuristics such as genetic algorithms, simulated annealing, tabu search, and ant colony optimization. Reinforcement learning techniques are also used in the literature to address the ALP and the ATP. Hybrids of metaheuristics and mathematical programming, termed matheuristics, are becoming more and more frequent in the recent literature.

We display in Figure 3 the number of articles from the recent literature, depending on the problem they address (ALP, ATP, or ASP), and regrouped in four types of methodologies: Dynamic programming, mixed-integer programming, metaheuristic/heuristic methods, and reinforcement learning. We can observe from this histogram that most of the works from the literature use heuristic approaches and metaheuristics. A possible explanation of this tendency is the (NP-hard) complexity of the problem, that makes researchers favor stochastic methods capable of providing good-quality solutions in low computation times, over exact methods that may require high computation times. Another possible explanation is the dynamic nature of the problem. Indeed, aircraft in practice appear dynamically in the scheduling horizon. Thus, an optimal solution of the RSP in a given scheduling horizon may not

be optimal when new aircraft are involved, and computing new optimal solutions with an exact method may require high computation times. We can also observe in this histogram that solution approaches are mainly focused on the aircraft landing problem; only few works address the aircraft take-off problem as an independent problem.

After this high-level summary of the solution approaches proposed in the literature, we propose an in-depth discussion of each method in the next two sections.

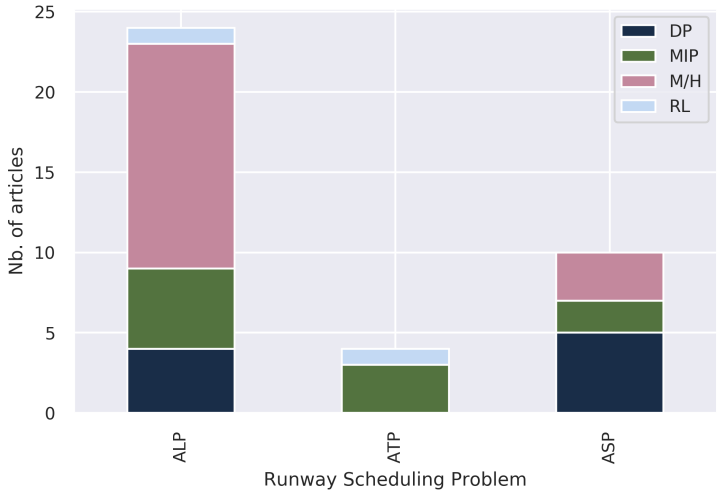


Figure 3: Summary of the number of articles for the RSP, regrouped in five types of methodologies: Dynamic Programming (DP), Mixed-Integer Programming (MIP), Metaheuristics/Heuristics (M/H), and Reinforcement Learning (RL)

5. Exact solution approaches

Since the publication of the first approach to solve the ALP [5], several research works were interested in different versions of the runway scheduling problem. In this section, we provide an overview of the most relevant exact solution approaches used in the literature, with a focus on recent contributions (from 2010 up to now). We also provide details on the test instances used to validate the approaches, and discuss results of numerical tests.

Several exact approaches are proposed in the literature to address the RSP; the majority of them are either Mixed-Integer Programming based

methods and dynamic programming approaches. In the current study, we focus on exact solution methods based on MIP or DP as these are the approaches that are widely used in the literature. Remark that there are a few studies that also address the RSP via constraint programming (see [29] for a survey on approaches using constraint programming for air traffic management problems, including the RSP). Here are two representative examples. A decision-support tool for air traffic controllers is proposed in [30] to plan the movements of departing aircraft (ATP). The problem is modeled as a constraint satisfaction problem that features several realistic aspects of the ATP, such as the airport layout, the take-off time slots, and the safety separation between pairs of aircraft. The model is solved using ILOG solver to schedule optimally medium-size airports. In [31], the authors highlight the analogy between the ALP and the K-king problem, where the airspace corresponds to a two-dimensional chessboard, the runway represents a special square, and the aircraft correspond to the kings.

Exact methods	Source	Problem	Nb of runways, configuration	Operational constraints	Objective	Model	Solution approach	Test instances (Nb of aircraft)
DP	Briskorn and Stolletz [27]	ALP	≥ 1, independent	Sep. and TW	min $\sum$ deviations	Directed graph + ELW, MIP + ELW	DP, CPLEX	Bianco *et al.* [32] (30 and 44)
	Lieder *et al.* [33]	ALP	≥ 1, independent	Sep. and TW	min $\sum$ delays	MIP + ELW, Directed graph + ELW + dominance criterion	CPLEX, DP	Bianco *et al.* [32] (30 and 44), generated (50 to 100)
	Lieder and Stolletz [26]	ASP	≥ 1, interdependent heterogeneous	Sep. and TW	min $\sum$ delays	MIP + ELW, Directed graph + ELW + dominance criterion	CPLEX, DP	Generated (30 to 62)
	Balakrishnan and Chandran [16]	ALP	1	CPS, Prec., Sep., and TW	min makespan	Directed graph	DP + CPS	Denver international airport (11 to 23), generated (10 to 50)
	Balakrishnan and Chandran [17]	ASP	1	CPS, Prec., Sep., and TW	min $\sum$ delays, min max delay	Directed graph	DP + CPS	Generated (10 to 50)
	Rathinam *et al.* [34]	ATP	1	Prec., Sep., and TW	min $\sum$ delays	Multi-objective optimization	Generalized DP	Generated based on DFW airport (6 to 40)
	Ravidas *et al.*[35]	ATP	2, independent	Prec., Sep., and TW	min $\sum$ delays	Multi-objective optimization	Generalized DP	Generated (8 to 25)
	Montoya *et al.* [36]	ATP	1	CPS, Prec., Sep., and TW	Multi-obj: min $\sum$ delays, max rwy throughput	Permutation problem	Multi-obj DP	DFW airport based (16 to 32)
	De Maere *et al.* [37]	ATP	1	Sep. and TW	Multi-obj: min (makespan, delay + CTOT compliance)	Permutation problem	DP + Pruning rules	London Heathrow airport (55), OR-group Bologna (60)
	Bennell *et al.* [38]	ALP	1	Sep. and TW	Multi-objective: min (makespan + ALT + Fuel + TWV)	Queue network	DP	London Heathrow airport (21, 42, and 84), Generated (depending on scenarios)
	Faye [39]	ALP (Fixed sequence)	1	Sep. and TW	min $\sum$ deviations	LP	DP, FICO-Xpress	OR-library (10 to 500)
MIP	Ghoniem and Farhadi [40]	ALP	≥ 1, independent	Sep. and TW	min $\sum$ delays	MIP + valid inequalities	CPLEX	OR-library (10 to 500), generated (15 to 25)
	Faye [41]	ALP	≥ 1, interdependent	Sep. and TW	min $\sum$ deviations	IP + time discretization	Matrix approx. + FICO Xpress	OR-library (10 to 44)
	Hancerliogullari *et al.* [42]	ALP	≥ 1, independent	Sep., TW, and traffic balance	min $\sum$ delays	MIP	Solver not specified	Generated (15 to 25)
	Kim *et al.* [43]	ALP	≥ 1, independent	Sep. and TW	min emissions	MIP	Solver not specified	Detroit airport (3 to 40)
	Furini *et al.* [44]	ASP	1	Sep. and TW	min $\sum$ delays	MIP	CPLEX + RH	OR-group Bologna (60)
	Malik et al. [45]	ASP	≥ 1, interdependent	CPS, Prec., Sep., and TW	min $\sum$ delays	MIP	GuRoBi	CLT airport based (10 to 35)
	Avella *et al.* [46]	ASP	1	Sep. and TW	min $\sum$ delays	BILP + VI	VFS + column generation + CPLEX	Stockholm airport (33 and 40), Hamburg airport (57, 58, and 72), OR-group Bologna (60)
	Prakash et al [4]	ASP	1	CPS, Sep., and TW	min makespan	MIP	GuRoBi + CPS + data-splitting	Generated (30 to 60)
	Pohl *et al.* [47]	ASP	≥ 1, independent	Sep., and TW	min delays	MIP	GuRoBi + VI + Pr + Starting solution	Munich airport (30, 40, 60, and 75), OR-library (100 to 500)

Table 2: A summary of recent exact solution approaches in the literature.

The ALP is thereby reduced to a constraint satisfaction problem. The authors propose several search strategies to solve the resulting model, and are able to schedule optimally small-sized instances (6 to 14 aircraft).

Table 2 summarizes the most relevant exact approaches surveyed in this section. In this table, the abbreviations Prec. and Sep. refer respectively to the Precedence and the Separation constraints. The acronyms ALT, TWV, and VFS mean respectively Average Landing-Time, Time-Window Violation, and Variable-Fixing strategies.

5.1. Dynamic programming

Psaraftis [48] develops one of the first DP approaches to address the ALP, involving at first only one runway, and then extending it to two runways. In his work, there is no assumption on the landing times, *i.e.*, all aircraft can land at time 0 and no restrictions are imposed on delays. Moreover, it is considered that the set of aircraft can be partitioned into a small number of *classes*: aircraft that belong to a same class are similar in the process of scheduling. Two objectives are investigated: minimizing the runway throughput, which is equivalent to minimizing the landing time of the last aircraft in the sequence, and minimizing the total passenger delay cost, which only depends on the aircraft class.

Briskorn and Stolletz [27] extend the approach to multiple independent[3] runways, and bounded landing times (time windows). Under the assumption that aircraft are partitioned into classes and ELW rule, they prove that, within each class, it is optimal to schedule aircraft according to the FCFS rule. The ALP is then modeled as finding the shortest path in a directed acyclic graph, and solved using DP. For the more general case of multiple runways and multiple aircraft classes, the state variables of the dynamic programming are the number of aircraft that have been scheduled in each class, and the runway occupation profile (time and aircraft class) of the last scheduled aircraft on each runway. Briskorn and Stolletz provide theoretical results about the polynomial complexity of the proposed approach, which is however exponential in the number, W, of aircraft classes: $O(RW^{R+1}n^{(R+1)W^2+R+1})$, where R is the number of runways, and n is the number of aircraft. The proposed approach is not implemented, but they adapt the MIP model of [9]

[3]Runways are said independent if operations (landings or take offs) on one runway do not affect operations on the other runways

to their objective function, which depends on each aircraft class. The obtained MIP formulation is then solved using CPLEX. They conclude from their results that computation times can be significantly reduced if a MIP model is enhanced with constraints based on the ELW rule.

The approach proposed in [33] uses the DP framework of [27], but it does not allow early landings (landings before target time). The authors also develop a criterion to avoid considering states that are dominated: if two states have the same number of aircraft (of each class) that have already been scheduled, but differ in terms of the runway occupancy profile, the *dominance criterion* then selects the state for which the runways are freed earlier. This dominance criterion together with their restriction that prohibits early landings significantly reduce the state space, which allow them to implement efficiently the DP algorithm of [27]. Results show the benefit of using this DP method instead of a standard MIP approach. Nevertheless, the proposed approach ensures only *successive separation* (separation between two successive aircraft), assuming that the triangular inequality necessarily holds, in order to guarantee separation between any pair of aircraft. Recall finally that the above authors assume that runways are independent. This assumption may not be true in a general setting: heterogeneous interdependent runways are common in airports. For this more general case, Lieder and Stolletz [26] extend the approach to ensure complete separation (separation between all pairs of aircraft), and to handle both landings and take offs.

Another DP framework is proposed by Balakrishnan and Chandran [16, 17, 18], assuming this time the CPS constraints. The idea in De Maere *et al.* [37] is to model the problem as a modified shortest path on a directed graph and then to solve it by DP. The graph consists of n stages (n is the number of aircraft), where each stage represents an aircraft position in the final sequence. A node in this graph features a subsequence of aircraft. Several objective functions are considered. In [16], the dynamic programming recursion aims at finding the landing schedule that maximizes the runway throughput, in a deterministic setting. The approach is later extended in [18] to take into consideration uncertainty in actual landing times, that may differ from the scheduled landing times, causing violation of the constraints. The objective is then to compute trade offs between good runway throughput and schedule robustness to the violation of safety constraints. The dynamic programming recursion is rewritten in [17] to consider other objectives, such as minimizing the average delay and minimizing the maximum delay.

Multi-objective dynamic programming is also proposed in the literature

to solve the RSP [34, 35, 36, 37].

In [34], Rathinam *et al.* consider the problem of scheduling aircraft departures (ATP) on a single runway. They take into consideration time-window constraints and precedence order between aircraft in a same runway queue. Separation requirements are ensured *only* between successive pairs of aircraft, since the authors assume that the *triangular inequality* holds. The objective function minimizes total delay. The authors reformulate this problem as a multi-objective optimization problem, that minimizes total delay and the departure time of the last aircraft in the sequence (makespan). The state variable of their multi-objective dynamic programming (called *generalized dynamic programming*) is a *partial* schedule, *i.e.*, a sub-sequence of aircraft with their corresponding take-off times. The dynamic programming recursion that incrementally computes the take-off times is proven to compute *correctly* all the non-dominated solutions for each state. Test instances are artificially generated, so that they mimic real-traffic scenarios on Dallas/Forth Worth airport. These instances involve 6 to 40 aircraft. Results show that this approach computes optimal departure schedules in very short computation times (less than one second). This encourages the extension of this work to the two (independent) runways case in [35]. In this latter work, the authors also artificially generate instances involving *only* 8 to 25 aircraft, which may be small given that they consider two runways. The computation times to find optimal departure schedules (for two runways) are reasonable for the instances involving 8 to 22 aircraft (less than one minute). However, for the remaining instances, especially for the ones involving 25 aircraft, the computation times may be long for a real-time application (more than 125 seconds).

The work of Montoya *et al.* [36] also addresses the ATP. The authors consider the same constraints as in [34] and [35], but add the CPS constraints. Regarding the separation constraints, three scenarios are taken into account: the *WV separation* between all pairs of aircraft (unlike [34] and [35] that consider *only successive* separation), *reduced WV separation* if the trailing aircraft is departing to a different metering fix[4], and a minimum *separation* due to runway *crossings* by arriving aircraft. The bi-objective function aims at minimizing total delay and maximizing runway throughput.

[4] Recall that a *metering fix* is a specific point along an established air route, over which aircraft will be metered in the terminal airspace surrounding airports.

The proposed multi-objective DP specifies the Pareto-optimal solutions, *i.e.*, non-dominated solutions with respect to the two above-mentioned objectives. A state in their DP is represented by a permutation of the subsets of aircraft. The overall algorithm can be decomposed in two main blocks. The *branching block* adds a new aircraft to the current partial solution, provided that it does not violate any constraint. The *Pruning block* removes dominated solutions. Tests are performed on some instances created by mimicking real-traffic scenarios on runway 17R in Dallas/Forth Worth airport, involving 16 to 32 planes. The multi-objective DP algorithm is then compared with a baseline algorithm, based on extreme solutions of the Pareto frontier obtained by the two algorithms, and using two performance metrics: the average total delay (in minutes) and the runway usage rate (number of aircraft per hour). Results show that their algorithm has performances similar to those of the baseline algorithm in terms of the runway usage. However, the multi-objective DP outperforms the basic algorithm in terms of total delays, reaching 55% decrease in delay for some instances.

The last work we review that uses multi-objective DP is De Maere *et al.* [37], which addresses an ATP that can be extended to incorporate arrivals, according to the authors. They consider time windows and separation constraints, without assuming that the triangular inequality holds. Unlike the above-mentioned works, De Maere *et al.* consider a multi-objective, non-convex piecewise linear function. This function is defined by two components: the first component represents the landing of the last aircraft in the sequence, while the second component is a weighted sum the delay and CTOT compliance. This objective function was defined in collaboration with runway controllers at London Heathrow airport. The authors propose six pruning rules to reduce the search space, without compromising the optimality. These pruning rules are incorporated in a dynamic program to find Pareto-optimal solutions. For the numerical study, the authors use 36 real-world instances from London Heathrow airport, that contain 55 aircraft each. Additional tests are also conducted on the 12 instances of the OR-group Bologna. The authors incorporate the CPS constraints in their algorithm, in order to compare it with to two DP approaches from the literature: Balakrishnan and Chandran [17] and Psaraftis [49]. Results show that their algorithm have similar average computation times, for small values of the CPS. However, for larger values of the CPS (especially: 8, 9, 10, and 55), their algorithm outperforms the two above-mentioned works. Moreover, the algorithm succeeds to solve all OR-group Bologna instances in 0.064 seconds. Remark that the

OR-group Bologna instances are not very challenging for today's high performance computers and current versions of solvers, as one can solve all of these 12 instances, with the MIP of Beasley [9] and CPLEX 12.8, in a total time of 2.8 seconds, as we shall show in Subsection 5.3.

Recently, Bennell *et al.* [38] proposed - in addition to two stochastic approaches - a dynamic programming approach for the ALP, involving a single runway dedicated to landings. They also consider a multi-objective function, but opt for a weighted sum of the objectives, given the (NP-hard) complexity of the problem. These objectives take into account the priorities of three stakeholders: the airport, by maximizing punctuality; the airlines, by minimizing fuel costs; and the air traffic control, by maximizing runway capacity. The constraints considered are the separation and the time windows. The state variable of the DP is similar in its definition to that of [27]. Their DP algorithm can be decomposed in three main steps. The *Next-Stage Generation* step, adds an aircraft to the list of the already landed aircraft, and assesses the cost of the new state if it is feasible. The second *Next-Stage Elimination* step eliminates redundant states and keeps the one that minimizes the cost function. The last *Select Solution* step selects, among the final states where all the aircraft have landed, the one with the lowest cost. This algorithm is tested not only for the static case, where all parameters are known in advance, but also for the dynamic case with a *sliding-horizon* approach. The numerical tests are performed on instances generated from real traffic at London Heathrow airport, involving 21, 42 and 84 aircraft. Additional tests are also carried out on some artificially-generated instances. The authors compare their algorithm with the performance of an air traffic controller on the basis of five performance indices. The results show that, for the static case, their algorithm have similar performances compared with an air traffic controller in terms of runway capacity. However, the dynamic programming (and their two other stochastic approaches) achieves better results on the other performance indices, especially on fuel cost. A possible explanation is that air traffic controllers mainly aim at maximizing runway capacity, without considering the two other objectives (fuel cost and punctuality).

Even more recently, Faye [39] proposed a DP algorithm to solve a subproblem of the ALP, which consists in finding the landing times for a fixed sequence of aircraft. The considered constraints are the time windows and the safety separation *only* between successive pairs of aircraft, because he also assumes that the triangular inequality holds. The objective function minimizes total deviations from target times, given by a convex piecewise

linear function. The proposed algorithm first tightens the time windows of all aircraft in the sequence. Then, it recursively constructs the minimal cost functions for increasing numbers of aircraft from the sequence. These cost functions are proven to be piecewise linear, with multiples breakpoints. Optimal landing times are then obtained from these functions, by finding the points that minimizes them. The dynamic programming part (which constructs the minimal cost functions) has a complexity $O(N^2)$, where N is the total number of aircraft in the sequence. The construction of optimal landing times are computed from these functions in $O(N)$. To examine the performance of his dynamic programming algorithm, Faye compares it with a linear program solved by FICO-Xpress. The two algorithms (DP and linear program + FICO-Xpress) are embedded in a simulated-annealing based algorithm, to solve the complete aircraft landing problem (sequencing + landing times). Tests are performed on the public instances of the OR-library , involving 10 to 500 aircraft. He concluded that, when embedded in a simulated annealing algorithm, his DP algorithm outperforms linear programming in terms of quality of the solutions obtained within the predefined time limits. Simulated annealing performs indeed more iterations (> 10 times) when combined with his DP than with linear programming within the same time limits, which allow a better exploration of the search space.

5.2. Mixed integer programming approaches

Beasley *et al.* [9] present the most cited MIP approach in the literature for the ALP. First, they introduce a model for scheduling aircraft landings on a single runway taking into consideration the WV separation, time-window constraints, and additional constraints that may be redundant, but useful to improve the linear programming relaxation. The objective is to minimize total deviations from target landing times. The model is also extended to multiple interdependent runways. It is solved using CPLEX, for increasing numbers of runways (up to four runways). The approach is tested on problems from the OR-library [50], involving up to 50 aircraft and 4 runways.

Briskorn and Stolletz [27] adapt the model presented in [9] to objective functions that depend on each aircraft class. They propose an approach that uses the ELW rule in the model and they then solve it with CPLEX. This yields better computation times than [9]. Ghoniem and Farhadi [40] adapt the formulation presented in [9] to incorporate take offs (ASP), and enhance it with valid inequalities. Their empirical tests show that valid inequalities and pre-processing may be not sufficient to solve optimally the MIP formulation

for even modest numbers of aircraft and runways (20 and 4, respectively). For this reason, they propose a set-partitioning model, and solve it with a column-generation approach, which allows them to solve larger instances in manageable computation times.

The same model is used in [41] in a time-discretization approach. Assuming integer-valued problem parameters, the author shows that the solution is also integer valued. The planning horizon is then discretized in time slots, so that no event occurs between two consecutive time slots, which leads to a finite number of possible scenarios (landing time and runway assignment) for each aircraft. Based on these scenarios, the author presents an alternative 0-1 linear problem. The proposed approach to solve exactly the resulting problem is called *dynamic constraints generation* algorithm. It can be decomposed in two main blocks: the first block solves a relaxed version of the above-mentioned 0-1 linear problem, obtained by relaxing the time-separation constraints. This relaxation relies on estimating each of the separation matrices (*longitudinal* separation for aircraft landing on the same runway, and *diagonal* separation for aircraft landing on different runways) by a rank-2 matrix, such that each entry of the rank-2 matrices is less than or equal to each entry of the corresponding separation matrix. In the second block, the algorithm detects pairs of aircraft that may violate the separation constraints due to the relaxation, then corrects them. It terminates with a solution of the relaxed problem that satisfies all separation constraints, which provides an optimal solution. By means of numerical tests on instances from the OR-library , it is shown that this approach yields better results than [9] in terms of computation times.

Hancerliogullari *et al.* [42] propose a MIP formulation for scheduling aircraft operations (ASP) on multiple runways inspired from job-shop scheduling on parallel machines. Their model differs from [9] in two aspects: first, the runways are supposed to be independent, *i.e.*, the separation requirements for aircraft landing on different runways are supposed to be automatically satisfied, while in [9], the model takes into consideration additional separation requirements (*diagonal* separation) for aircraft landing on different runways. The second aspect is related to balancing traffic on the available runways, that [42] incorporates in the model, by adding lower- and upper-bound constraints on the number of aircraft assigned to a runway. The authors report results in terms of computation times for solving their proposed MIP formulation (solver not specified) on artificially-generated congested instances, ranging from 15 to 25 aircraft, involving 2 to 5 runways. Their tests feature

large computational times to obtain optimal solutions for almost all test instances. This motivates the authors to propose metaheuristics and greedy algorithms – that we shall review in Section 6 – to obtain good-quality solutions in shorter computation times.

Kim *et al.* [43] propose a novel MIP approach to include – additionally to runways – terminal-area entry fixes. Their model computes the runway and TRACON-fixe [5] assignment of each aircraft, as well as a runway schedule for landings, that minimizes total emissions in the TRACON, including airport surface emissions. By means of numerical tests on data from Detroit international airport, they show that their approach does not only reduce emissions by 29.9% in the studied case, but also increases runway throughput, thanks to inbound traffic balance among runways and TRACON fixes.

Salehipour *et al.* [51] consider the same problem as [9]. They propose a MIP model similar to that of [9]. The two mathematical models are similar in the definition of the decision variables, but the model of [51] simplifies some redundant constraints. Results obtained with CPLEX show that instances from the OR-library involving up to 50 aircraft and 3 runways can be solved to optimality in reasonable computation times.

Furini *et al.* [44] present an alternative MIP formulation to schedule both landings and take offs (ASP) on a single runway, so as to minimize total weighted delay. In their model, continuous variables are used to indicate a runway operation (landing or take off), and binary variables are used to indicate the aircraft *position* in the runway sequence, which is the main difference between their formulation and the model of [9]. The model is solved using CPLEX within a rolling-horizon approach to find improved solutions (not necessarily optimal), compared with the FCFS rule. Tests are performed on instances from the OR-group Bologna, and feature small computation times to solve instances of 60 aircraft. Later [10], this model is compared with the formulation of [9], on some constructed instances, and imposing a time limit for CPLEX. Results show that the formulation of [9] outperforms that of [44], in terms of both computation times and quality of the solution obtained within the predefined time limit. However, the tabu-search approach that they develop outperforms the results obtained by CPLEX on the model of [9].

Malik *et al.* [45] construct a runway scheduler for Charlotte Douglas

[5] Terminal Radar Approach CONtrol (TRACON): controlled airspace close to airports

(CLT) airport, that schedules different operations (landings, take-offs, and crossings) on multiple interdependent runways. The objective is to minimize total delay. The particularity of this work is that it considers, additional to classical constraints (CPS, time-window, separation), constraints that are specific to the layout of CLT airport. An example of such specific constraints is to fix an FCFS order for aircraft departing to (or arriving at) a same metering fix, and specifying a special spacing between these aircraft. Another particularity of this work, compared with other MIP-based approaches from the literature, is that it uses the (reduced) separation matrix from the RECAT projects [13], introduced in Subsection 2.1. For the numerical tests, the authors generate instances based on real traffic on CLT airport, involving 10 to 35 aircraft in a 15-minute time period. Results show that this approach leads to significant improvements in total delay, compared with the FCFS order. However, the run time of the algorithm grows too large with increasing numbers of aircraft for the value 3 of the maximum position-shifting parameter.

Unlike the majority of MIP formulations from the literature, Avella *et al.* [46] opt for a time-indexed formulation for the ASP. They motivate this choice by the fact that time-index formulations represent a good trade-off between the compactness of the formulation and the quality of the Linear-Programming (LP) bounds. The authors take into consideration separation constraints and time-window restrictions, with the objective of minimizing total tardiness. The RSP is modeled as a Binary-Integer Linear Program (BILP), enhanced with valid inequalities. The proposed exact algorithm to solve their resulting BILP can be decomposed in three main steps. First, the *Presolve* phase uses variable-fixing strategies, and tightens the time windows. Then, the *LP-relaxation* step solves the LP relaxation of their BILP model, which provides lower bounds on the optimal solution. Finally, a feasible integer (upper bound) solution is then computed (using CPLEX) over *only* a subset of variables generated from the previous step, and the optimality of the algorithm is assessed by comparing these (upper and lower) bounds. Extensive computational tests are conducted using: (i) Stockholm Arlanda airport instances with 33 and 40 aircraft, (ii) Hamburg airport instances that involve 57, 58, and 72 aircraft, and (iii) the OR-group Bologna instances with 60 aircraft. To examine the effectiveness of their formulation, they compare it with the basic BILP model (without any valid inequalities or variable-fixing strategies), in terms of the quality of the solutions provided by CPLEX within a time limit of 600 seconds. Results show that the enhanced model

is superior to the basic model, in terms of both computation times and the number of instances solved within the predefined time limit. Moreover, 31 of the 39 (Arlanda and Hamburg airports) instances are solved in less than 30 seconds. Their algorithm also solves all the OR-group Bologna instances in less than 4 seconds.

Prakash *et al.* [4] present a MIP-based approach for scheduling both landings and take offs (ASP) on a single runway, in which they incorporate the CPS constraint explicitly. The MIP formulation is enhanced with variable-fixing strategies, based on the CPS and on the ELW rule. The objective is to minimize the completion time of the sequence (makespan). They propose a data-splitting method that makes use of the CPS to split the original sequence into all possible pairs of leading and trailing aircraft sub-sequences, and then solve independently each pair of sub-sequences using GuRoBi, while ensuring global optimality. For the numerical tests, they artificially generate congested instances (30 to 40 aircraft), mimicking peak traffic conditions, and show that this approach outperforms state-of-the-art DP methods in terms of computation times, especially for values of the maximum position-shift parameter larger than 3.

Pohl *et al.* [47] also opt for a MIP approach for the integrated problem of scheduling simultaneously landings, take-offs, and snow removals during winter times. They take into consideration the time-window constraints, the separation requirements between aircraft, and the separation requirements between snow removals and next aircraft operations, to guarantee the safety of operations on the runway. The objective function minimizes the total weighted delay. Solving directly their MIP model leads to prohibitory computation times (beyond one hour). For this reason, they perform some preprocessing and add valid inequalities to the MIP formulation to reduce the search space and tighten the LP relaxations. Moreover, they construct a feasible-heuristic solution that they feed to GuRoBi solver as a starting solution. Numerical tests are performed on real-traffic instances from Munich airport, involving 30, 45, 60, and 75 aircraft, and 2 or 3 runways. Results show that preprocessing, valid inequalities, and the starting solution for GuRoBi significantly reduce the computation times, and yield optimal solutions obtained within 60 seconds for most of the Munich-airport instances. Moreover, the proposed algorithm performs better (in terms of total delay) than the heuristic solution used by controllers and runway planers to schedule aircraft and snow removals on runways. Additional tests are also performed on the very large instances (100 to 500 aircraft) from the OR-library , and

show that their MIP approach succeeds to solve to optimality 9 out of the 14 instances within 60 seconds. However, for the three instances of the OR-library : `airland10`, `airland11`, and `airland13`, all with 3 runways, the computation times may be considered too long (beyond one hour) for a real-time application.

5.3. Numerical tests

In this part, we present results obtained after implementing three different MIP models from the literature, namely Beasley *et al.* [9], Salehipour *et al.* [51], and Furini *et al.* [10], to which we shall refer to in the sequel with the author acronyms: BKSA, SMM and FKPT, respectively. We test the three models on the benchmark instances that are the most used in the literature:

- The *OR-library* instances [50]. Available on line: `http://people.brunel.ac.uk/~mastjjb/jeb/info.html`.
- The *OR-group Bologna* instances [10]. Available on line: `http://or.dei.unibo.it/`.

All the numerical tests are performed on a personal computer under GNU/Linux operating system, processor Intel(R) Core(TM) i7-4700M with 8 GB of RAM, and implemented using Docplex, the Python API of the CPLEX solver, version 12.8.

In Table 3, the first column reports the benchmark instance origin, and the second column presents the name of the instances. The third and fourth columns display the total number of aircraft, $|\mathcal{A}|$, and runways, $|\mathcal{K}|$, respectively. The remaining columns present the computation time, in seconds, to find an optimal solution for each of the three above-mentioned formulations. We impose a time limit of 300 seconds (5 minutes) for each test. For the FKPT formulation, the symbol "-" simply indicates that tests involving multiple runways are not relevant, since this model is not suited to this case.

It can be seen from Table 3 that, for the OR-library instances involving up to 50 aircraft, and for the OR-group Bologna instances, the two models BKSA and SMM succeed to find optimal solutions in very short computation times, whereas the model FKPT requires long computation time, even for small instances of $|\mathcal{A}| = 15$ aircraft. We can deduce that, with a good MIP formulation, and current versions of solvers (*e.g.*, CPLEX 12.8), even large

Table 3: Numerical tests

	Instance	$\|\mathcal{A}\|$	$\|\mathcal{K}\|$	CPU time (s)		
				BKSA [9]	SMM [51]	FKPT [10]
OR-library	airland1	10	1	0.05	0.03	43.05
			2	0.03	0.04	-
	airland2	15	1	0.16	0.07	63.05
			2	0.07	0.09	-
	airland3	20	1	0.07	0.03	> 300.00
			2	0.12	0.14	-
	airland4	20	1	6.69	0.54	> 300.00
			2	154.90	2.14	-
	airland5	20	1	20.38	4.04	> 300.00
			2	67.36	3.20	-
	airland6	30	1	0.03	0.01	> 300.00
			2	29.16	0.17	-
	airland7	44	1	1.46	0.11	> 300.00
			2	1.08	0.04	-
	airland8	50	1	0.90	0.25	> 300.00
			2	3.66	5.65	-
	airland9	100	1	> 300.00	> 300.00	> 300.00
			2	> 300.00	63.00	-
OR-group Bologna	FPT01	60	1	0.23	0.24	> 300.00
	FPT02	60	1	0.27	0.27	> 300.00
	FPT03	60	1	0.24	0.28	> 300.00
	FPT04	60	1	0.45	0.44	> 300.00
	FPT05	60	1	0.20	0.29	> 300.00
	FPT06	60	1	0.20	0.15	> 300.00
	FPT07	60	1	0.33	0.35	> 300.00
	FPT08	60	1	0.33	0.39	> 300.00
	FPT09	60	1	0.28	0.41	> 300.00
	FPT10	60	1	0.28	0.39	> 300.00
	FPT11	60	1	0.13	0.12	> 300.00
	FPT12	60	1	0.06	0.07	> 300.00
Ikli	alp_7_30	30	1	> 300.00	> 300.00	> 300.00
			2	126.00	> 300.00	-
	alp_7_40	40	1	> 300.00	> 300.00	> 300.00
			2	> 286.00	> 300.00	-
	alp_7_50	50	1	> 300.00	> 300.00	> 300.00
			2	> 300.00	> 300.00	-
	alp_11_30	30	1	> 300.00	> 300.00	> 300.00
			2	8.00	214.00	-
	alp_11_40	40	1	> 300.00	> 300.00	> 300.00
			2	> 300.00	> 300.00	-
	alp_11_50	50	1	> 300.00	> 300.00	> 300.00
			2	> 300.00	> 300.00	-
	alp_15_30	30	1	> 300.00	> 300.00	> 300.00
			2	189.00	> 185.00	-
	alp_15_40	40	1	> 300.00	> 300.00	> 300.00
			2	117.00	> 300.00	-
	alp_15_50	50	1	> 300.00	> 300.00	> 300.00
			2	> 300.00	> 300.00	-
	alp_19_30	30	1	7.00	5.00	> 300.00
			2	2.00	2.00	-
	alp_19_40	40	1	> 300.00	> 300.00	> 300.00
			2	3.00	290.00	-
	alp_19_50	50	1	> 300.00	> 300.00	> 300.00
			2	> 300.00	> 300.00	-

instances from the literature involving up to 60 aircraft, can be solved to optimality within very short CPU times. The only difficult instances that remain from the literature are the very large ones, involving 100 to 500 aircraft.

For this reason, we construct four new data sets as a basis for generating 12 novel test instances for the ALP. These data sets, together with the test instances and implementations from this subsection, are publicly available from `http://data.recherche.enac.fr/ikli-alp/`.

The construction of these instances is explained in Appendix B which also provides some important information on each instance, as well as on the four data sets that serve as a basis to create new ALP instances. Results of the three formulations BKSA, SMM and FKPT on these new test instances are presented in Table 3. Observe from this table that only few instances are solved to optimality within short computational times, namely: `alp_11_30` with two runways, `alp_19_30` with one and two runways, and `alp_19_40` with two runways. None of the tree models BKSA, SMM and FKPT succeed to solve the remaining instances within reasonable computation times.

6. Stochastic solution approaches

The dynamic nature of the runway scheduling problem motivates the recourse to solution approaches with low computation times that allow the update of solutions whenever a new event occurs. Several researchers thereby favor heuristic approaches and metaheuristics.

In the literature prior to 2010, Genetic Algorithms (GA) were the mostly-used metaheuristic [52, 53, 54, 55, 56]). Recently, Tabu Search (TS) and Simulated Annealing (SA) gained much more attention, and became the most-used metaheuristics [10, 19, 42, 51, 57, 58, 59]. Other metaheuristics are also used in the literature, such as Ant Colony Optimization (ACO) [60, 61, 62], and Variable Neighborhood Descent (VND) [51, 63]. Table 4 summarizes the most relevant stochastic approaches surveyed in this section, including metaheuristics, heuristics, and novel approaches based on reinforcement learning.

6.1. Ant colony optimization

A number of ACO frameworks are proposed in the literature for the ALP. For the single runway case, Zhan *et al.* [60] consider the objective of minimizing total delay. They present, for the first time, an ant colony system incorporated in a Receding Horizon (RH) algorithm named RHC-ACS-ASS

(*Receding Horizon Control* with *ant colony system* for *aircraft sequencing and scheduling*), controlled by two parameters: the time interval of a scheduling window, and the total width of the RH. In their proposed ant colony system, the FCFS heuristic is used to compute the initial *pheromone*. Then, local and global updates are performed to increase *population diversity* (diversification), and increase the *desirability* of best found solutions (intensification). Their numerical studies artificially generate congested instances, involving 30 and 60 aircraft and consider as well data from [64], in order to compare their approach with the GA introduced in [54] that use the same instances. They show that, when incorporated in a rolling-horizon approach, the ACO performs better than the GA.

For the multiple runway case, and the more general objective of minimizing the total deviation cost, Bencheikh *et al.* [61] present an adapted ACO framework for runway and landing-time assignment. Unlike the model of [60], based on permutations, Bencheikh *et al.* propose a bi-level graph to model the ALP. The first level chooses an available runway, and the second sets the aircraft to land on this runway. In their proposed ACO algorithm, ants start from a dummy initial node, select a runway, then an aircraft to insert in this runway, based on the priority of the aircraft and the memory of the ant colony. The selection process is repeated until the list of aircraft available to land is empty. Global pheromone updates are then performed, and the overall algorithm ends when a stopping criterion is met. Numerical tests are performed on the public instances of the OR-library, involving 10 to 50 aircraft, and 1 to 5 runways. Results show that their ACO enhanced with the improvement heuristic has short computation times, and is capable of finding optimal solutions for 80% of the test instances.

Stochastic methods	Source	Problem	Nb of runways, configuration	Operational constraints	Objective	Model	Solution approach	Test instances (Nb of aircraft)
Meta-heuristics	Zhan *et al.* [60]	ALP	1	Sep. and TW	min $\sum$ delays	Permutation problem	ACO + RH	Generated (30 and 60), Bianco [32] (20 and 30)
	Bencheikh *et al.* [61]	ALP	≥ 1, interdependent	Sep. and TW	min $\sum$ deviations	Bi-level graph	ACO + local search	OR-library (10 to 50)
	Hu [65]	ALP	1	Sep. and TW	min makespan	Permutation problem	Matrix approximation + ACO	Hu *et al.* [54] (30), Generated (20 to 260)
	Hu and Paolo [52]	ALP	1	Sep. and TW	min $\sum$ delays	2D Artificial space + Ripple spreading	GA	Generated (20 to 60)
	Shohel *et al.* [66]	ASP	≥ 1, different configurations	Sep.	max rwys throughput	Constrained optimization problem	CPS + GA	Generated Based on Chicago O'Hare airport (50)
	Furini *et al.* [10]	ASP	1	Sep. and TW	min $\sum$ delays	Permutation problem	TS + RH	OR-group Bologna (60)
	Soykan and Rabadi [57]	ASP	≥ 1, independent	Sep., TW, and Traffic balance	min $\sum$ delays	MIP	TS + TTFGA	Ghoniem *et al.* [67](15 to 25)
	Salehipour *et al.* [63]	ALP	≥ 1, independent	Sep. and TW	min $\sum$ delays	MIP	VND + GA	Hansen [68] (12, 15 and 20)
	Salehipour *et al.* [51]	ALP	≥ 1, independent	Sep. and TW	min $\sum$ deviations	MIP	VND + SA , VNS + SA	OR-library (10 to 500)
	Sabar and Kendall [69]	ALP	≥ 1, interdependent	Sep. and TW	min $\sum$ deviations	MIP	ILS+VND	OR-library (10 to 500)
	Hancerliogullari*et al.* [42]	ALP	≥ 1, independent	Sep., TW, and traffic balance	min $\sum$ delays	MIP	SA + AATCSR, SA + ERT, SA + FPI	Generated (15 to 25)
	Rodriguez-Diaz *et al.* [19]	ASP	1	CPS, Sep., and TW	min $\sum$ delays	Permutation problem	SA + CPS	OR-library (10 to 500), Generated (50 to 200), Gatwick airport (not specified)
	Hammouri *et al.* [58]	ALP	≥ 1, interdependent	Sep. and TW	min $\sum$ deviations	Permutation problem	SA + Iterated local search	OR-library (10 to 500)
Heuristics	Vadlamani and Seyedmohsen [70]	ALP	1	Sep. and TW	min $\sum$ deviations	MIP	ALNS + CPLEX	OR-library (10 to 150)
	Xiao-Peng *et al.* [71]	ALP	1	Sep. and TW	min makespan	Permutation problem	DSSE algorithm	Beijing airport(10 to 150), Generated (20 to 50), Zhan *et al.* [60]
	Salehipour and Ahmadian [72]	ALP	1	Sep. and TW	min $\sum$ deviations	IP	Local search + CPLEX	OR-library (10 to 500)
	Salehipour [73]	ALP	≥ 1, independent	Sep. and TW	min $\sum$ deviations	MIP	Local search + GuRoBi	OR-library (10 to 500), GHiniem *et al.* [40] (15 to 50)
	Ikli *et al.* [74]	ALP	1	CPS, Sep., and TW	min $\sum$ delays	Search tree	OP-based heuristic	ALP instances [75] (18 to 40)
RL	Soares *et al.* [24]	ATP	1	Sep. and TW	min taxiing delay	MDP	*Q-learning*	JFK airport
	Brittain and Wei [25]	sequencing and separating aircraft	-	Sep.	min conflicts	RL formulation	*Hierarchical* deep RL	NASA sector-33 game [76] (2 to 5)

Table 4: A summary of recent stochastic solution approaches in the literature.

An ant-colony based algorithm is also proposed for the ALP, with the objective of minimizing the makespan in [65]. The considered constraints are the time windows and the WV separation, which is assumed to satisfy the *triangular inequality*. The proposed solution approach has two main steps. In the first step, the separation matrix is approximated by a rank-2 matrix, similar to [41] discussed in Subsection 5.2. This new matrix renders the problem sequence independent, and hence easier to solve. Then, an ant colony algorithm is used to compensate the loss in precision due to the separation matrix approximation. The overall algorithm stops when a time limit is reached. For the numerical study, the author uses an instance borrowed from [54]. He also generates other instances involving 20 to 260 aircraft, with an increment of 30. The comparison with the ACO algorithm of [77] on the instance of [54] shows that his algorithm succeeds to find the optimal solution within 15 seconds, whereas the algorithm of [77] finds *only* a good-quality solution, but in *less* than one second. For the large generated instances, the algorithm of [65] finds better solutions – in terms of the makespan of the sequence – than CPLEX within the predefined time limits that are questionable for a real-time application (120 and 3600 seconds).

6.2. Genetic algorithm

Hu and Paolo [52] consider the problem of scheduling aircraft landings on a single runway, in order to minimize total airborne delay. They introduce a novel GA framework, in which the representation of *chromosomes* differ from the classical representations based on aircraft permutations [54, 68]. The approach is called *Ripple-Spreading Genetic Algorithm* (RSGA), inspired from the ripple spreading on a liquid surface. It consists in projecting a candidate sequence onto an artificial space, where the representation of a solution is based on numerical values (points) rather than on a permutation, then connect all these points by the ripple-spreading process, to form a landing sequence. In this work, the chosen artificial space is a two-dimensional space: the x axis represents time and the y axis displays the aircraft WV categories, converted in the same time unit as the x axis, using WV time separation requirements. Traditional GAs are then used to optimize the ripple-spreading parameters. Extensive computational tests are performed on generated data involving up to 60 aircraft, considering under-congested instances, normal instances, and very congested instances. The authors of [52] claim similar performance in terms of total airborne delay compared with other GA-based

approaches from the literature for the normal and the under-congested instances, and smaller delays for the congested instances.

Shohel *et al.* [66] also opt for a GA-based approach to solve the ASP. The novelty in this work is that it considers, in addition to the ASP, the problem of finding the best-fit runway configuration[6], *i.e.*, a runway configuration that maximizes runway throughput. It is the first time such an integrated problem is tackled in the literature of the RSP, according to the authors. The considered constraints are the final-approach separation (WV separation) and the CPS constraints. The later constraints are integrated in their genetic algorithm. The chosen solution approach relies on a cooperative GA, where a *population* of candidate aircraft sequences and candidate runway configurations co-evolve to find the best-fit sequence and a runway configuration that maximizes the runway throughput. Numerical tests are conducted on generated instances, mimicking peak traffic hours in Chicago O'Hare airport, that contains 8 runways. The possible runway configurations are also based on this airport. The authors observe that, with their integrated model, they achieve a better runway utilization compared with the baseline capacity of the above-mentioned airport. Indeed, the baseline capacity is 168 operations (landings and take-offs) per hour, while their model achieves a capacity of 170 operations per hour.

6.3. Tabu search

Furini *et al.* [10] consider the problem of scheduling aircraft operations on a single runway (ASP). Two types of constraints are considered: WV separation and time windows, with the objective of minimizing the total weighted delay. They propose a rolling-horizon approach that consists in subdividing the initial instance into a set of sub-instances, called *chunks*, according to some *chunking rules*. Then, they sequentially solve each individual sub-instance using either their MIP approach (reviewed in Subsection 5.2), or a TS method. In their proposed TS, a candidate solution is defined by a permutation of the set of aircraft. Neighborhood solutions are obtained either by swapping two aircraft positions, or by shifting an aircraft to a new position, if the new candidate solution is not forbidden by the *tabu list*. The TS continues improving the candidate solution until a stopping criterion (maximal

[6]Recall that a runway configuration represents the number of runways and their layout, *e.g.*, single runway, multiple parallel runways, multiple crossing runways, etc.

number of iterations or time limit) is met. Numerical tests on the OR-group Bologna instances show that their TS outperforms the MIP approach to solve the small chunks, within the predefined time limit of 15 seconds.

Soykan and Rabadi [57] propose a TS approach for the more general problem of scheduling aircraft operations on multiple independent runways. Their approach is decomposed in two main steps. The first step is similar to the FCFS sequence; it computes an initial solution using a greedy approach, called *Target Time First Greedy Algorithm* (TTFGA), which consists in making aircraft land / take off according to their ascending order of target times, and on the runway on which the (weighted) delay is minimal. The second step is a TS that improves the initial solution. Candidate solutions are generated either by swapping two aircraft on the same runway, or on different runways, or by deleting/inserting aircraft off/in the sequences. When a best-to-date value of the objective function is found, an *aspiration* mechanism is employed to ignore the tabu restrictions. When no improvement occurs after a given number of iterations, then the algorithm stops. Tests are performed on instances from [67] involving 15, 20, and 25 aircraft, and up to 5 runways. Theirs results feature very short computation times (less than 1 second), and an average gap to the optimal solution of 10.15%. However, the gap may be large for some instances, in particular for instance 48, which involves 25 aircraft and 4 runways, where the gap is 91.75%.

6.4. Variable Neighborhood Descent

VND is a metaheuristic that belongs to the family of Variable Neighborhood Search (VNS) algorithms, introduced by Mladenovic and Hansen [78]. It is used by Salehipour *et al.* [63] to tackle the problem of scheduling aircraft landing on multiple independent runways, in order to minimize total schedule delay. The safety separation between pairs of aircraft landing on the same runway are assumed constant (2 units of time), contrarily to the classical WV separation that depends on the leading and the trailing aircraft types. The separations between pairs of aircraft landing on different runways are also constant (1 unit of time). The proposed VND computes an initial guess based on a genetic algorithm introduced in [68]. Four neighborhood structures are proposed: swapping aircraft on a same runway, on different runways, or swapping runway sequences; the fourth neighborhood removes an aircraft from one runway to insert it on another runway. Their numerical tests rely on generated instances from [68], involving up to 20 aircraft

and 5 runways. Their approach improves the first guess, but requires high computation times, compared with state-of-the-art heuristics.

Salehipour *et al.* [51] use the VND in a hybrid algorithm to improve candidate solutions within an SA. The algorithm, called SA+VND, relies on 3 out of the 4 neighborhood structures defined above: swapping two aircraft landing on a same runway or on different runways, and the remove/insert technique. The VND chooses one neighborhood structure, and attempts at improving the current incumbent solution. When no more improvement is possible, then the neighborhood structure is changed. When the search reaches the last neighborhood, the process is repeated until a stopping criterion is met. Computation results on the OR-library instances show that their hybrid metaheuristic is able to find the optimal solution for instances involving up to 50 aircraft, and good-quality solutions – in terms of optimality gap – for the very large instances involving up to 500 aircraft.

VND was also used in Sabar and Kendall [69], within an iterative local search algorithm, to tackle the multiple-runway ALP. the considered constraints are the time windows and the WV separation. The chosen objective function minimizes the total deviations from target landing times. The local-search phase in their algorithm is based on a VND that relies on four neighborhood structures, similar to those introduced in [63]. The VND stops if the local optimum found by the last neighborhood cannot be improved any further. Computational tests are conducted on the OR-library instances, involving 10 to 500 aircraft, and 1 to 5 runways. The authors compare different set-ups of their algorithm with several heuristics from the literature: the hybrid metaheuristics of [51], the Scatter search and the Bionomic algorithm of [79]. Although their algorithm could not find the best-known solutions for some instances, it was able to find new best solutions for 16 out of the 49 instances. Moreover, from a computational time perspective, the approach of [69] has the smallest run times compared with the above-mentioned heuristics, for all the OR-library instances.

6.5. Simulated annealing

Hancerliogullari *et al.* [42] present several heuristics to address their model described in Subsection 5.2, including an SA framework, whose initial guess is constructed using three types of greedy algorithms. The *Earliest Ready Time* (ERT) greedy algorithm consists in landing aircraft according to their arrival order, on the runway where it can start its operation (landing or take off) the earliest (similar to Soykan and Rabadi [57] greedy algorithm). The *Adapted*

Apparent Tardiness Cost with Separation and Ready Times (AATCSR), and the *Fast Priority Index* (FPI) greedy algorithms inspired from the job-shop scheduling literature, that land aircraft according to a priority index. Generating candidate solutions for the SA is performed by randomly choosing two aircraft and swapping their positions, provided that their time-window constraints are not violated. Several tests performed on generated instances of 15, 20, and 25 aircraft show that their SA framework requires short computation times, and significantly improves the initial solutions provided by their three greedy algorithms AATCSR, ERT and FPI.

Rodriguez-Diaz *et al.* [19] consider the problem of scheduling aircraft operations on a single runway (ASP), taking into consideration the WV separation and the CPS constraints. The objective is to propose near-optimal solutions that significantly improve the FCFS order, without deviating too much from it. Similarly to [51], they opt for an SA framework. candidate solutions are generated by swapping aircraft so as to fulfill the CPS and the WV separation constraints. Numerical tests performed on generated instances for which the exact solutions are known *a priori* show that their approach is able to find an optimal solution for 828 out of the 2000 instances (41.4%). Further tests on the OR-library instances are performed to compare their SA approach with the heuristics of [51] and [79]. Results show that their SA outperforms the hybrid metaheuristic VNS+SA of [51], in terms of computation times for all the 12 instances in OR-library, whereas the results of [51] and those of [79] are better in terms of percentage improvement (but the later allow earliest landings, *i.e.*, landings before target times). Additional tests performed on real data from Gatwick airport, show that the SA approach of [19] can reach a 30% of improvement on real data, for large values of the maximum number of position shifts.

Recently, Hammouri *et al.* [58] proposed a hybrid method that uses simulated annealing and iterated local search to tackle the ALP, involving one or multiple runways, and taking into consideration time-window restrictions and safety separation constraints. The proposed approach contains two main loops. The inner loop uses simulated annealing to improve current solutions through a cooling schedule, using the same neighborhood structures as in [63]. The outer loop relies on an iterated local search approach to perturb the solution or restart the search, so as to explore other solution space regions, and escape local minima. The perturbation process is performed using the same above-mentioned neighborhood structures, but without guidance from the objective function. Extensive computational tests are conducted on in-

stances from the OR-library to compare their proposed approach with state-of-the-art methaheuristics, such as the hybrid method of [72] and the scatter search of [79]. Results are reported in terms of required computational time and the gap to the best-known solutions, and show that their algorithm is able to find the best-known solutions for all the small instances (up to 50 aircraft) in very short computation times. For some of the remaining large instances, this algorithm does not find the best-known solutions. However, it reaches new best results on four large instances, and has the best mean computational time, when compared with the other methods.

6.6. Matheuristics

Matheuristics are hybrid optimization methods obtained by the interpolation of mathematical programming and metaheuristics [80]. Some works from the literature have used these methods. In particular, Vadlamani and Seyedmohsen [70], Salehipour and Ahmadian [72], and the very recent work of Salehipour [73].

Vadlamani and Seyedmohsen [70] consider the ALP involving a single runway, with the objective of minimizing total weighted deviations from target times. They propose a heuristic approach, based on Adapted Large Neighborhood Search (ALNS), which can be decomposed in two steps. First a *scheduling problem* finds a near-optimal landing sequence. Then, a *feasibility problem* finds landing times satisfying the constraints via a linear program solved with CPLEX. The approach is tested on the OR-library instances, and compared with the SA and the SA+VNS metaheuristic of [51]. Their approach finds optimal solutions for instances involving 10 to 150 aircraft, outperforming SA+VNS that could not find optimal solutions for instances with 50 and 100 aircraft. However, the computation times for the instance of 150 aircraft are long (234 seconds) for a real-time application context.

Salehipour and Ahmadian [72] consider the same problem as in [70], and opt for a heuristic approach that can be likewise decomposed into two steps. A first step generates an initial sequence based on the target times in a FCFS fashion. It is then improved using a local search algorithm that allows a percentage of aircraft to change their initial position, until a stopping criterion is met. With the sequence fixed, the scheduling of landing times is finally performed using CPLEX. Tests are performed on the OR-library instances, and the approach is compared with two of the best existing heuristic in the literature: the scatter search heuristic of Pinol and Beasley [79], and the hybrid VNS+SA metaheuristic of Salehipour *et al.* [51]. Results show that

their heuristic has the best gap for 10 out of the 13 instances from the OR-library. This approach was recently extended to the multiple-runway case, by Salehipour [73]. This recent work assumes that runways are independent, *i.e.*, no separation is required between aircraft landing on different runways. Thus, the author constructs a feasible solution (in a FCFS-like oder) for each runway, and then improves it using the above-discussed heuristic. Extensive computational tests are conducted on the OR-library instances, with 1 to 5 runways, as well as on the 75 benchmark instances from [40] that contains 15 to 50 aircraft, and 2 to 5 runways. This approach is compared with GuRobi and two other heuristics [69, 81] from the literature, in terms of percentage of best solutions obtained within predefined time limits. Results show that this heuristic outperforms the two other heuristic and GuRoBi on the Ghoniem *et al.* [40] instances. On the other hand, for the OR-library instances, the heuristic of [69] has the best score in terms of percentage of best solutions.

6.7. Other heuristics

Unlike most of the works in the literature that deal with the static RSP, Xiao-Peng *et al.* [71] consider the dynamic scheduling of aircraft landings, taking into consideration time-window and separation constraints. The chosen objective function is the makespan of the sequence. The authors model the ALP as a (constrained) permutation problem, and propose a population-based heuristic algorithm, named *Dynamic Sequence Searching and Evaluation (DSSE)*, combined with a receding-horizon approach to solve it. The scheduling horizon is divided into smaller intervals, and the ALP is gradually solved on each interval, using the DSSE algorithm. Several test instances are used in the numerical study: (i) an instance from Beijing capital international airport with 50 aircraft, (ii) 12 generated instances involving 20 to 50 aircraft, and (iii) some other instances from Zhan *et al.* [60] with 20 and 30 aircraft. The DSSE algorithm is then compared with several heuristics from the literature, including that of Vadlamani and Seyedmohsen [70] reviewed above. Results show that the algorithm of [71] is very competitive in terms of computation times. Moreover, it reaches the best objective value on average, on the 12 generated instances.

Ikli *et al.* [74] also consider the ALP involving a single runway, with the objective of minimizing the total weighted delay. They propose a novel heuristic-search approach based on Optimistic Planning [82, 83]. It models the ALP as an environment composed of *states*, *actions*, *transitions* and *costs*

inspired from Markov decision process. The heuristic search starts at the initial state where all aircraft are available to land, and tries to identify which sequence of actions are the best to take (*i.e.*, which sequence of aircraft to land) within a limited time budget. Tests are performed on realistic instances involving 18 to 40 aircraft, available at [75], imposing for each instance a time limit of 3, 5 and 15 seconds. Results show that their approach can reach, for some instances, a percentage of objective-function value improvement of 37% with respect to the FCFS heuristic.

6.8. Reinforcement learning

Reinforcement learning [84] is used to tackle a wide variety of problems; this includes the problem of sequencing and scheduling aircraft operations.

Soares *et al.* [24] consider the problem of scheduling aircraft take-offs on a single runway, with the objective of respecting the assigned take-off time windows. The problem is modeled as an MDP, and solved using a popular RL algorithm, called *Q-learning* [85]. In their proposed MDP, *agents* correspond to aircraft, *states* correspond to the position of the aircraft on the ground, depending on its phase (parked, taxiing, taken-off). *Actions* consist in delaying aircraft, and the *reward* is defined so as to minimize delay during taxiing, in order to respect aircraft assigned time windows. Tests are performed on real data from John F. Kennedy international airport, that feature 698 departure flights, which corresponds to two days of operations in this airport. They generate 42 learning scenarios from these data. Results show that their algorithm has similar performance in terms of percentage of time windows respected compared with human ground controllers. However, when facing disturbance (stochastic case), their algorithm performs better.

Brittain and Wei [25] propose another framework to model the problem of sequencing and separating aircraft, so that the model fits the NASA sector-33 environment [76], which is an air traffic control application that contains 35 problem instances involving up to five aircraft. They involve providing speed and route control to aircraft so as to sequence and separate them. Their proposed model (similar to an MDP) is composed of *agents*, *states*, *actions*, and *rewards*. This model features two types of agents: *parent agents* and *child agents*. The *state* of a parent agent contains a screen-shot of the game screen. The *state* of a child agent contains information about the metering fix goal, aircraft speed and acceleration, a route identifier, additionally to information about the N-closest agents in order to permit communication

between agents. *Actions* for a parent/child agent consist in changing or maintaining the route/speed of aircraft. The *reward* is designed so as to penalize agents in conflict (separated by less that 3 nm). A *hierarchical* deep reinforcement learning algorithm is then used to solve the model. This algorithm combines *Q-learning* [85] algorithm and neural networks [86]. It is called *hierarchical* because actions are taken at two levels: the *parent-state* level chooses the route, then *child-state* level selects the speed for the aircraft. Tests performed on the above NASA environment (involving only 2 to 5 flights) show the viability of their approach to sequence properly and separate aircraft from and to given metering fixes.

7. Conclusion and future directions

Making the best use of Air Traffic Management infrastructures, especially the runway, which is considered to be the major bottleneck, is the motivation behind runway scheduling problems ALP, ATP, and the integrated problem ASP.

Several models and solution approaches are proposed in the literature, ranging from exact approaches, mainly dynamic programming and mixed-integer programming, to stochastic approaches including metaheuristics such as genetic algorithms, simulated annealing, tabu search, and ant colony optimization. Reinforcement learning techniques are also used in the literature to tackle the ALP and the ATP. Hybrids of metaheuristics and mathematical programming, termed matheuristics, are becoming more and more frequent in the recent literature. Notable examples of such methods are the hybrids of mathematical programming and local search of [70, 72].

Very efficient DP algorithms are proposed in the literature for the single runway case, considering different objectives (makespan, delays, etc.), and even multi-objective functions. However, for the multiple-runway case, DP struggles to find optimal solutions in reasonable computation times, except if one assumes additional restrictions that reduce the search space (*e.g.*, precedence order, CPS constraints, time windows, and the triangular-inequality assumption). Mixed-integer programming approaches are very flexible, and allow one to incorporate different operational constraints, and different types of the separation requirements that may arise when considering multiple runways or multiple operations on the runway (*e.g.*, landings, take-offs, crossings, snow removals). Moreover, they yield optimal solutions in short computation times for the OR-group Bologna instances, and for the benchmark problems

from the literature (OR-library instances) that involve up to 50 aircraft and 4 runways. However, for the very large instances ($\geq$ 100 aircraft), MIP suggers from the NP-hard complexity of the RSP.

If constrained position-shifting restrictions are taken into consideration, the data-splitting method of [4], and stochastic solution approaches such as the simulated annealing of [19] seems to be efficient approaches to solve the RSP for small values of the maximum number of position shifts. The first one yields optimal solutions in short computation times for some generated instances, outperforming the state-of-the-art dynamic-programming method of [17]. The second approach provides good-quality solutions for all the OR-library instances – including the very large ones – within a few seconds (less than 5 seconds).

If the CPS restrictions are not considered in the RSP however, then exact methods do not solve large instances in reasonable computation times. The scatter search of [79], the hybrid metaheuristic VND+SA of [51], the hybrid iterated local search / simulated annealing of [58], and the hybrid iterated local search / VND of [69] appear to be good candidate approaches to solve the RSP with one or multiple runways, and with the more general objective of minimizing deviations from target times. They provide good-quality solutions in terms of the gap to the best-known solutions, within reasonable computation times on the OR-library test problems.

After reviewing the state-of-the art methods proposed in the literature, we derive some concluding remarks, and suggest promising future directions.

- *Refining aircraft categories*: Researchers in the literature usually use the classical three ICAO (International Civil Aviation Organization) wake-turbulence categories introduced in Subsection 2.1, and their corresponding separation requirements. However, this categorization is rough and often leads to over-separation. A better understanding of the wake-turbulence phenomenon and the improvement of support tools assisting controllers have lead to refining the classical *three* categories into *six* new categories, as in RECAT projects [13]. Hence, a more accurate approach would consider the above-mentioned six wake-turbulence categories. In addition to being more accurate, the use of RECAT categories results in a non-negligible delay saving, as shown in the work of [87].

- *Integrating scheduling and air traffic control*: Most approaches consider only the scheduling problem to propose a landing (or take-off) time slot

for each aircraft, resulting in a better utilization of the runway system with the proposed time slots. However, the air traffic control problem of guiding aircraft so as to respect the scheduling solution is rarely considered. A more realistic approach could integrate the air traffic control problem, for instance by adding explicit altitude, speed, and heading control decisions to the optimization variables. These control actions (altitude, speed and heading) will allow air traffic controllers to judge whether a schedule is feasible in practice or not.

- *Enlarging the air traffic environment*: Works in the literature usually focus on optimizing solely the runway throughput, ignoring other aspects of the airspace close to the airport that are specific to each airport, which also contribute to the congestion. Indeed, during periods of congestion, bottlenecks may occur elsewhere in the terminal airspace close to airports, resulting in significant delays, not only at runways, but also at other points in this area [88]. More realistic approaches therefore consider for instance the specific (approach and departure) route structure of the airport, as addressed in [89, 90, 91, 92]. These examples consider not only the runway, but also approach and departure route segments and holding circles in the optimization.

- *Enlarging further to surface operations*: The ALP, ATP, and the management of *surface operations* (*i.e.*, taxiing) are often addressed separately in the literature, and handled independently in the airports. However, these three problems are linked and should ideally be treated at once. Indeed, if one considers each problem independently, the solutions found may not be feasible in practice, since the three problems are linked. Another alternative consists in considering these (sub-)problems sequentially. This may require less computation times compared to the complete integrated problem, but may lead to sub-optimal solutions. Examples from the literature that consider the integrated problem are [59, 93, 94]. However, the proposed approaches in these works solve the integrated sub-problems sequentially.

- *Matheuristics*: A few works from the literature use matheuristics [80], which are hybrid methods of mathematical programming and metaheuristics. Notable examples are [70, 72], which hybridize mathematical programming with a local search method. These approaches have shown their effectiveness in solving the very large instances from the

OR-library. Investigating further these hybrid methods for other problem instances, and for more realistic runway scheduling problems is therefore a promising future direction.

- *Addressing uncertainty*: The sources of uncertainty in air transportation are numerous, *e.g.*, weather change, human factors involved in air traffic management, and delay propagation, which result in a considerable uncertainty in the input data of the runway scheduling problem, which are commonly assumed to be known in a deterministic way. Only few authors address the uncertainty issue. Recently, [95] proposed a robust optimization framework for the ASP. Even more recently, [96] introduced a two-stage stochastic programming approach for the ALP. However, there is still a need for approaches that consider different sources of uncertainty, such as the time aircraft appear in the radar range, and the availability of runways.

Appendix A. Final-approach separation-matrix conversion

The wake-vortex separation matrix between landings – called final-approach separation matrix – is given in terms of distance, and expressed in nautical miles (nm) as shown in Table A.5. Since scheduling models are usually based on time, the separation matrix must be converted into an equivalent time-based matrix.

		Following aircraft		
		H	M	L
Leading aircraft	H	4	5	6
	M	2.5	2.5	4
	L	2.5	2.5	2.5

Table A.5: Final-approach separation matrix (nm) according to the ICAO's basic WV categories (Khassiba *et al.* [96])

The conversion process is complex, since it involves modeling the airspeed profile of each aircraft type, the wind speed, and the equation of motion on the final approach path. We present here a simplified model to convert the above distance matrix into a time matrix, based on the approach of de Neufville *et al.* [97] for computing the capacity of a single runway.

Appendix A.1. Definitions

Consider a single runway used solely for landings, and a meter fix (the Final Approach Fix (FAF) for instance) where arriving aircraft are merged to the final-approach segment, and must maintain the minimum required separation from each other, as shown in Figure A.4. Consider two aircraft i and j, where i is the leading aircraft, and j is the following aircraft.

We define the following quantities (we use the same notation as in [97]):

- r: distance (in nm) between the meter fix (FAF) and the runway threshold.
- S_{ij}: minimum WV separation (in nm) between aircraft i and j.
- v_i and v_j: an approximation of the final approach speed (in knots (kts), 1 kts = 1 nm/h or 1.852 km/h) of aircraft i and j respectively, supposed fixed, and depend on the aircraft WV category.

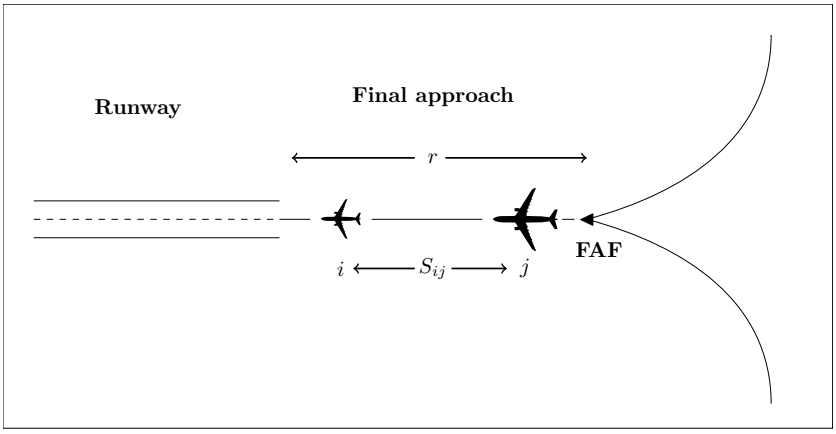


Figure A.4: A runway and the final-approach path

- o_i and o_j: runway occupancy time (in seconds) of aircraft i and j respectively, from the touch down to the instant when the aircraft leaves the runway.
- T_{ij}: minimum WV separation (in seconds) between two aircraft i and j.

Appendix A.2. Conversion formula

To determine the time separation T_{ij}, two cases can be distinguished depending on the speed of the leading and the trailing aircraft i and j respectively:

- The ***opening case***: when $v_i > v_j$. In this case, it is sufficient to verify the separation only at the meter fix to ensure that the two aircraft remain separated up to the runway threshold. Suppose that aircraft i starts its final approach, aircraft j must be at least at distance S_{ij} from aircraft i; since the final-approach path length is r, aircraft i and j will arrive at the runway threshold after $\frac{r}{v_i}$ and $\frac{r+S_{ij}}{v_j}$ respectively. Thus, the time separation between the two aircraft must satisfy:

$$T_{ij} \geq \frac{r + S_{ij}}{v_j} - \frac{r}{v_i} \tag{A.1}$$

  In addition, a runway safety constraint imposes that aircraft j cannot proceed to land before the leading aircraft j clears the runway, *i.e.*,

$$T_{ij} \geq o_i \tag{A.2}$$

The time separation in the opening case is therefore:

$$T_{ij} = \max\{\frac{r + S_{ij}}{v_j} - \frac{r}{v_i}, o_i\} \tag{A.3}$$

- The ***closing case***: when $v_i \leq v_j$. The separation in this case must be satisfied on the runway threshold. When aircraft i arrives at the runway threshold, aircraft j must be at least at distance S_{ij} from i, and hence will land not earlier than $\frac{S_{ij}}{v_i}$ later. Taking into consideration the same runway safety constraint, the required time separation is then:

$$T_{ij} = \max\{\frac{S_{ij}}{v_j} -, o_i\} \tag{A.4}$$

Appendix A.3. Example of computation

Table A.6 gives numerical values of the quantities defined in Appendix A.1, according to the WV categories. For the length of the final-approach path, *i.e.*, from the FAF to the runway threshold, we take $r = 5$ nm. The distance separation matrix is the ICAO WV separation matrix, given in Table A.5.

	WV categories		
	H	M	L
v_i (kts)	150	130	110
o_i (s)	70	60	55

Table A.6: Numerical values of the approach speed and runway occupancy time according to the WV categories [97]

Numerical application example: H-M

Consider two aircraft types H and M from the three WV categories. Suppose that the leading aircraft, i, is in the category H, while the following aircraft, j, is in the category M. From Table A.6, we have $v_i > v_j$, which corresponds to the *opening case*. The time separation is therefore, according to A.3 (using the category indices "H" and "M" instead of the aircraft indices "i" and "j"):

$$
\begin{aligned}
T_{HM} &= \max\{\frac{r + S_{HM}}{v_M} - \frac{r}{v_H}, o_H\} \\
&= \max\{\frac{5+5}{\frac{130}{3600}} - \frac{5}{\frac{150}{3600}},\ 70\} \\
&\approx 157 \text{ seconds}
\end{aligned}
$$

If we apply the same process to all possible pairs of successive aircraft types, from Table A.5, we get the WV separation matrix given in Table A.7.

		Following aircraft		
		H	M	L
Leading aircraft	H	96	157	240
	M	60	69	156
	L	60	69	82

Table A.7: The WV separation matrix in seconds

Appendix B. The new ALP instances

This appendix is organized as follows. We first explain the construction process of the 12 test instances used in Subsection 5.3. Then, we report some important features and statistics on each instance, such as the number of aircraft from each WV category, which reflects the mix of aircraft in the instance, and the number of aircraft per hour, which captures its congestion level.

Appendix B.1. Construction details

The raw data comes from two traffic days on Paris-Orly airport, obtained from the OpenSky Network [98]: one day in July 2018 (322 scheduled aircraft), and one day in April 2019 (294 scheduled aircraft). In order to obtain a larger and more congested set, we merge these two traffic days. Then, we remove duplicate aircraft to avoid redundancy. Next, we divide this data set into the following 4-hour intervals: 07:00 – 11:00 , 11:00 – 15:00, 15:00 – 19:00, and 19:00 – 23:00. This subdivision leads to the four data sets named `data_7_11.csv`, `data_11_15.csv`, `data_15_19.csv`, and `data_19_23.csv` in [99]. Finally, we artificially add light aircraft to each of the four data sets, so as to obtain a mix of 40% heavy, 40% medium, and 20% light aircraft, mimicking realistic peak traffic hours, as suggested in Prakash *et al.* [4].

These four data sets serve as a basis to generate test-bed instances for the aircraft landing problem. Indeed, the 12 test instances used as a benchmark in Subsection 5.3 are generated from these data sets, by simply considering the first $|\mathcal{A}|$ lines of data ($|\mathcal{A}| \in \{30, 40, 50\}$) of each of the four data sets. These 12 instances may therefore serve as a benchmark for future studies, and further test instances may also be generated from the four basis data sets.

The basis data sets, test instances, and implementations from Subsection 5.3 are publicly available at `http://data.recherche.enac.fr/ikli-alp/`. The information provided for each aircraft includes: the model of the aircraft (`mdl`), sometimes called aircraft type in the literature, the WV category (`category`), the target landing time (`sta`), and the actual landing time (`ata`) in HH:MM:SS format and in seconds. We also provide realistic delay costs for each aircraft, based on the work of Cook *et al.* [100], and of Cook and Tanner [101]. These two reports conduct detailed delay cost analysis, taking into account several factors: fuel consumption, number of passengers, maintenance, cabin crew, etc. Another particularity of

these reports is that they provide a piecewise linear cost function, which is more realistic than the linear cost given in the benchmark instances from the literature. However, these reports present delay costs only for some aircraft models (15 aircraft models in the latest report [101]). In order to extrapolate the costs for other aircraft types, we construct a linear regression fit-function on the aircraft of [101]. Details about this regression (variables, parameters, implementation, and r^2 score) are also available from `http://data.recherche.enac.fr/ikli-alp/`, in the folder `ikli_codes`.

Figure B.5 displays the delay cost variation for all the 26 aircraft models/types available in our data sets. The abscissas 300, 900, 1800, and 3600 are the (delay) breakpoints of the piecewise linear cost function, in seconds (corresponding to: 5, 15, 30, and 60 minutes, respectively). To the best of our knowledge, this is the first time such a detailed and realistic piecewise linear cost function is provided in a publicly available data set for the ALP.

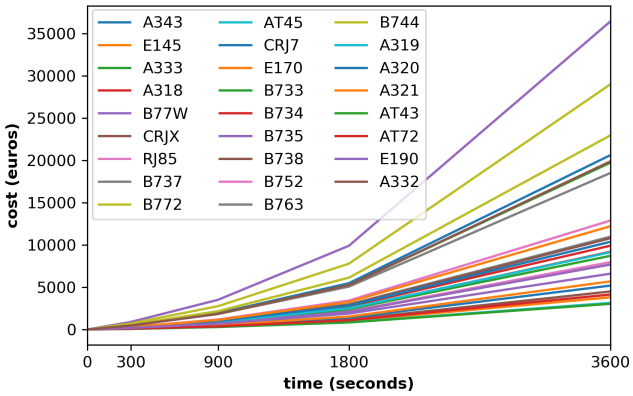


Figure B.5: Variation in the delay cost, in euros, for all aircraft types in our data sets

Appendix B.2. Data sets and instances description

Table B.8 summarizes some characteristics of the data sets. Throughout this table, the first and second columns present the name and the time interval of each data set. The third and fourth columns report the total number of aircraft "$|\mathcal{A}|$", and the average number of aircraft per hour "Average ac/hour", in each data set. The last column "Mix of ac" shows the mix of aircraft on each set, in terms of the three WV categories: H, M, and L, introduced in Section 2.

Data-set name	Time interval	$\|\mathcal{A}\|$	Average ac/hour	Mix of ac		
				H	M	L
data_7_11.csv	07:00 - 11:00	185	46.25	74	74	37
data_11_15.csv	11:00 - 15:00	207	51.75	83	83	41
data_15_19.csv	15:00 - 19:00	203	50.75	80	82	41
data_19_23.csv	19:00 - 23:00	167	41.75	67	67	33

Table B.8: Description of Ikli data sets.

Characteristics of the 12 benchmark instances are displayed in table B.9. We also report the number of aircraft per hour "Number ac/hour" for these instances: it is the number of aircraft in the instance ($|\mathcal{A}|$) divided by the time length of the instance (time interval), in hours. This number is an important characteristic, because it captures how congested the instance is.

From Table B.9, it can be seen that the most congested instances among the 12 proposed instances are `alp_15_30.csv` and `alp_15_40.csv`. They both have a number of aircraft per hour equal to 60, which corresponds to realistic scenarios of peak demand. Indeed, considering a scheduling horizon of one hour and peak traffic conditions, aircraft may appear in the scheduling horizon every 50 or 60 seconds [4]. On the other hand, the least congested instances are `alp_19_30.csv` and `alp_19_40.csv`, which both have a similar number of aircraft per hour (~ 36). This may explain why they are solved to optimality by CPLEX, in short computation times, using the MIP formulation of [9] (Table 3).

Instance origin	Instance name	Time interval	$\|\mathcal{A}\|$	Number ac/hour	Mix of ac		
					H	M	L
	alp_7_30.csv	07:00 - 07:40	30	45.0	13	11	6
data_7_11.csv	alp_7_40.csv	07:00 - 07:59	40	40.68	17	14	9
	alp_7_50.csv	07:00 - 08:10	50	42.86	22	17	11
	alp_11_30.csv	11:00 - 11:45	30	40.0	7	16	7
data_11_15.csv	alp_11_40.csv	11:00 - 11:51	40	47.06	11	18	11
	alp_11_50.csv	11:00 - 11:58	50	51.72	17	21	12
	alp_15_30.csv	15:00 - 15:30	30	60.0	15	11	4
data_15_19.csv	alp_15_40.csv	15:00 - 15:40	40	60.0	19	14	7
	alp_15_50.csv	15:00 - 15:51	50	58.82	22	19	9
	alp_19_30.csv	19:00 - 19:50	30	36.0	8	17	5
data_19_23.csv	alp_19_40.csv	19:00 - 20:05	40	36.92	9	21	10
	alp_19_50.csv	19:00 - 20:15	50	40.0	15	25	10

Table B.9: Description of Ikli-instances.

Appendix C. Acronyms

Acronym	Meaning
ACO	Ant Colony Optimization
ALP	Aircraft Landing Problem
AMAN	Arrival MANager
ASP	Aircraft Scheduling Problem
ATC	Air Traffic Control
ATM	Air Traffic Management
ATP	Aircraft Take-off Problem
CPS	Constrained Position Shifting
DMAN	Departure MANager
DP	Dynamic Programming
ELW	Earliest Landing-time Windows
FCFS	First-Come First-Served
GA	Genetic Algorithms
ICAO	International Civil Aviation Organization
MDP	Markov Decision Process
MIP	Mixed Integer Programming
RSP	Runway Scheduling Problem
SA	Simulated Annealing
SMAN	Surface MANager
TS	Tabu Search
VNS	Variable Neighborhood Search
WV	Wake Vortex

Table C.10: Table of acronyms

References

[1] M. Caswell, Business Traveler, https://www.businesstraveller.com/news/2016/04/07/dublin-airport-to-build-second-runway-by-2020, 2016. On line; accessed April 21, 2021.

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

[3] J.-B. Gotteland, N. Durand, J.-M. Alliot, Handling CFMU slots in busy airports, in: Proceedings of the Fifth Air Traffic Management R&D Seminar, Budapest, Hungary, 2003.

[4] R. Prakash, R. Piplani, J. Desai, An optimal data-splitting algorithm for aircraft scheduling on a single runway to maximize throughput, Transportation Research Part C: Emerging Technologies 95 (2018) 570–581.

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

[6] G. S. Veresnikov, N. A. Egorov, E. Kulida, V. G. Lebedev, Methods for solving of the aircraft landing problem. I. Exact solution methods, Automation and Remote Control 80 (2019) 1317–1334.

[7] G. S. Veresnikov, N. A. Egorov, E. Kulida, V. G. Lebedev, Methods for solving of the aircraft landing problem. II. Approximate solution methods, Automation and Remote Control 80 (2019) 1502–1518.

[8] J. Atkin, H. Hoogeveen, R. Stolletz, Airport operations management, Editorial of OR Spectrum 41 (2019) 613–614.

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

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

[11] M. Liang, Aircraft route network optimization in terminal maneuvering area, Ph.D. thesis, Université Paul Sabatier, Toulouse, France, 2018.

[12] The international civil aviation organisation, ICAO doc 4444., https://ops.group/blog/wp-content/uploads/2017/03/ICAO-Doc4444-Pans-Atm-16thEdition-2016-OPSGROUP.pdf, 2016. On line; accessed April 21, 2021.

[13] J. Cheng, A. Hoff, J. Tittsworth, W. A. Gallo, The development of wake turbulence re-categorization in the United States, in: 8th AIAA Atmospheric and Space Environments Conference, Washington, D.C., USA, 2016.

[14] L. Bianco, P. Dell'Olmo, S. Giordani, Scheduling models for air traffic control in terminal areas, Journal of Scheduling 9 (2006) 223–253.

[15] F. Neuman, H. Erzberger, Analysis of delay reducing and fuel saving sequencing and spacing algorithms for arrival traffic, Technical Report, TM-103880, Ames Research Center, NASA, USA, 1991.

[16] H. Balakrishnan, B. Chandran, Scheduling aircraft landings under constrained position shifting, in: AIAA Guidance, Navigation, and Control Conference and Exhibit, Keystone, CO, USA, 2006.

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

[18] B. Chandran, H. Balakrishnan, A dynamic programming algorithm for robust runway scheduling, in: 2007 American Control Conference, NYC, USA, 2007, pp. 1161–1166.

[19] A. Rodríguez-Díaz, B. Adenso-Díaz, P. L. González-Torre, Minimizing deviation from scheduled times in a single mixed-operation runway, Computers & Operations Research 78 (2017) 193–202.

[20] K. Artiouchine, P. Baptiste, C. Dürr, Runway sequencing with holding patterns, European Journal of Operational Research 189 (2008) 1254–1266.

[21] The Civil Aviation Authority, Airspace design guidance: Noise mitigation considerations when designing PBN departure and arrival procedures, Technical Report, Aviation House West Sussex, UK, 2016.

[22] H. Erzberger, T. Davis J, S. Green, Design of center-TRACON automation system, in: Proc. AGARD Guidance and Control Symposium on Machine Intelligence in Air Traffic Management, Berlin, Germany, 1993.

[23] A. Villardi de Montlaur, L. Delgado Munoz, Delay assignment optimization strategies at pre-tactical and tactical levels, in: Fifth SESAR Innovation Days, Bologna, Italy, 2015.

[24] I. Brito Soares, Y.-M. De Hauwere, K. Januarius, T. Brys, T. Salvant, A. Nowe, Departure management with a reinforcement learning approach: Respecting CFMU slots, in: IEEE 18th International Conference on Intelligent Transportation Systems, Las Palmas de Gran Canaria, Spain, 2015.

[25] M. Brittain, P. Wei, Autonomous aircraft sequencing and separation with hierarchical deep reinforcement learning, in: Proceedings of the International Conference for Research in Air Transportation, Barcelona, Spain, 2018.

[26] A. Lieder, R. Stolletz, Scheduling aircraft take offs and landings on interdependent and heterogeneous runways, Transportation Research Part E: Logistics and Transportation Review 88 (2016) 167–188.

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

[28] J. K. Lenstra, A. R. Kan, Complexity of vehicle routing and scheduling problems, Networks 11 (1981) 221–227.

[29] C. Allignol, N. Barnier, P. Flener, J. Pearson, Constraint programming for air traffic management: A survey, The Knowledge Engineering Review 27 (2012) 361–392.

[30] P. Van Leeuwen, N. Van Hanxleden Houwert, Scheduling aircraft using constraint relaxation, in: Proceedings of the 22nd Workshop of the UK Planning and Scheduling Special Interest Group, Glasgow, UK, 2003.

[31] K. Artiouchine, P. Baptiste, J. Mattioli, The K king problem, an abstract model for computing aircraft landing trajectories: On modeling a dynamic hybrid system with constraints, INFORMS Journal on Computing 20 (2008) 222–233.

[32] L. Bianco, P. Dell'Olmo, S. Giordani, Minimizing total completion time subject to release dates and sequence-dependent processing times, Annals of Operations Research 86 (1999) 393–415.

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

[34] S. Rathinam, Z. Wood, B. Sridhar, Y. Jung, A generalized dynamic programming approach for a departure scheduling problem, in: AIAA Guidance, Navigation, and Control Conference, Chicago ILL, USA, 2009.

[35] A. Ravidas, S. Rathinam, Z. Wood, An optimal algorithm for a two runway scheduling problem, Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering 227 (2013) 1122–1129.

[36] J. Montoya, S. Rathinam, Z. Wood, Multiobjective departure runway scheduling using dynamic programming, IEEE Transactions on Intelligent Transportation Systems 15 (2014) 399–413.

[37] G. De Maere, J. A. D. Atkin, E. K. Burke, Pruning rules for optimal runway sequencing, Transportation Science 52 (2018) 898–916.

[38] J. A. Bennell, M. Mesgarpour, C. N. Potts, Dynamic scheduling of aircraft landing, European Journal of Operational Research 258 (2017) 315–327.

[39] A. Faye, A quadratic time algorithm for computing the optimal landing times of a fixed sequence of planes, European Journal of Operational Research 270 (2018) 1148–1157.

[40] A. Ghoniem, F. Farhadi, A column generation approach for aircraft sequencing problems: A computational study, Journal of the Operational Research Society 66 (2015) 1717–1729.

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

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

[43] B. Kim, L. Li, J.-P. Clarke, Runway assignments that minimize terminal airspace and airport surface emissions, Journal of Guidance, Control, and Dynamics 37 (2010) 789–798.

[44] F. Furini, C. A. Persiani, P. Toth, Aircraft sequencing problems via a rolling horizon algorithm, in: International Symposium on Combinatorial Optimization, Springer, 2012, pp. 273–284.

[45] W. Malik, H. Lee, Y. C. Jung, Runway scheduling for Charlotte Douglas international airport, in: 16th AIAA Aviation Technology, Integration, and Operations Conference, Washington D.C. USA, 2016.

[46] P. Avella, M. Boccia, C. Mannino, I. Vasilyevc, Time-indexed formulations for the runway scheduling problem, Transportation Science 51 (2017) 1196–1209.

[47] M. Pohl, R. Kolisch, M. Schiffer, Runway scheduling during winter operations, Omega In Press, Reference: 102325 (2020).

[48] H. N. Psaraftis, A dynamic programming approach to the aircraft sequencing problem, Technical Report, R78-4, Flight Transportation Laboratory, MIT, Cambridge, MA, USA, 1978.

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

[50] J. E. Beasley, OR-library: Distributing test problems by electronic mail, Journal of the Operational Research Society 41 (1990) 1069–1072.

[51] A. Salehipour, M. Modarres, L. Moslemi Naeni, An efficient hybrid metaheuristic for aircraft landing problem, Computers & Operations Research 40 (2013) 207–213.

[52] X.-B. Hu, E. A. D. Paolo, A ripple-spreading genetic algorithm for the aircraft sequencing problem, Evolutionary Computation 19 (2010) 77–106.

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

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

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

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

[57] B. Soykan, G. Rabadi, A tabu search algorithm for the multiple runway aircraft scheduling problem, in: Heuristics, Metaheuristics and Approximate Methods in Planning and Scheduling, Springer, 2016, pp. 165–186.

[58] A. I. Hammouri, M. S. Braik, M. Azmi Al-Betar, M. A. Awadallah, ISA: A hybridization between iterated local search and simulated annealing for multiple-runway aircraft landing problem, Neural Computing and Applications (2019) 1–21.

[59] J. Ma, D. Delahaye, M. Sbihi, P. Scala, M. A. Mujica Mota, Integrated optimization of terminal maneuvering area and airport at the macroscopic level, Transportation Research Part C: Emerging Technologies 98 (2019) 338–357.

[60] Z.-H. Zhan, J. Zhang, Y. Li, O. Liu, S. Kwok, W. Ip, O. Kaynak, 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 (2010) 399–412.

[61] G. Bencheikh, J. Boukachour, A. Alaoui, Improved ant colony algorithm to solve the aircraft landing problem, International Journal of Computer Theory and Engineering 3 (2011) 224–233.

[62] Y. Jiang, Z. Xu, X. Xu, Z. Liao, Y. Luo, A schedule optimization model on multi runway based on ant colony algorithm, Mathematical Problems in Engineering , Volume 2014, Article ID 368208 (2014) 11 pages.

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

[64] L. Bianco, P. Dell'Olmo, S. Giordani, Scheduling models and algorithms for TMA traffic management, in: Modelling and Simulation in Air Traffic Management, Springer, 1997, pp. 139–167.

[65] B. Hu, An efficient ant colony algorithm based on wake-vortex modeling method for aircraft scheduling problem, Journal of Computational and Applied Mathematics 317 (2017) 157–170.

[66] A. M. Shohel, S. Alam, M. Barlow, A cooperative co-evolutionary optimisation model for best-fit aircraft sequence and feasible runway configuration in a multi-runway airport, Aerospace 5 (2018) 26 pages.

[67] A. Ghoniem, F. Farhadi, M. Reihaneh, An accelerated branch-and-price algorithm for multiple-runway aircraft sequencing problems, European Journal of Operational Research 246 (2015) 34–43.

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

[69] N. R. Sabar, G. Kendall, An iterated local search with multiple perturbation operators and time varying perturbation strength for the aircraft landing problem, Omega 56 (2015) 88–98.

[70] S. Vadlamani, S. Hosseini, A novel heuristic approach for solving aircraft landing problem with single runway, Journal of Air Transport Management 40 (2014) 144–148.

[71] J. Xiao-Peng, C. Xian-Bin, D. Wen-Bo, T. Ke, An evolutionary approach for dynamic single-runway arrival sequencing and scheduling problem, Soft Computing 21 (2017) 7021–7037.

[72] A. Salehipour, M. Ahmadian, A heuristic algorithm for the aircraft landing problem, in: The 22nd International Congress on Modelling and Simulation (MODSIM), Tasmania, Australia, 2017.

[73] A. Salehipour, An algorithm for single-and multiple-runway aircraft landing problem, Mathematics and Computers in Simulation 175 (2020) 179–191.

[74] S. Ikli, C. Mancel, M. Mongeau, X. Olive, E. Rachelson, An optimistic planning approach for the aircraft landing problem, in: Proceedings of ENRI International Workshop on ATM/CNS, Tokyo, Japan, 2019.

[75] S. Ikli, The aircraft landing problem instances, `https://personnel.isae-supaero.fr/emmanuel-rachelson/alp-instances.html`, 2019. On line; accessed April 21, 2021.

[76] J. Colen, NASA sector 33 application, `https://www.nasa.gov/centers/ames/Sector33/iOS/index.html`, 2013. On line; accessed April 21, 2021.

[77] W. Ma, X. Bu, M. Liu, H. Huang, An efficient approximation algorithm for aircraft arrival sequencing and scheduling problem, Journal of Computational and Applied Mathematics , Volume 2014, Article ID 236756, 8 pages (2014).

[78] N. Mladenović, P. Hansen, Variable neighborhood search, Computers & Operations Research 24 (1997) 1097–1100.

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

[80] M. Boschetti, V. Maniezzo, M. Roffilli, A. Boluf Röhler, Matheuristics: Optimization, simulation and control, in: Proceedings of 2009 Hybrid Metaheuristics, Lecture Notes in Computer Science, Springer, pp. 171–177.

[81] A. Awasthi, O. Kramer, J. Lassig, Aircraft landing problem: An efficient algorithm for a given landing sequence, in: IEEE 16th International Conference on Computational Science and Engineering, pp. 20–27.

[82] J.-F. Hren, R. Munos, Optimistic planning of deterministic systems, European Workshop on Reinforcement Learning, Springer (2008) 151–164.

[83] R. Munos, From bandits to Monte-Carlo tree search: The optimistic principle applied to optimization and planning, Foundations and Trends in Machine Learning 7 (2014) 1–129.

[84] R. S. Sutton, A. G. Barto, Reinforcement learning: An introduction, MIT press, 1998.

[85] C. J. Watkins, P. Dayan, Q-learning, Machine Learning 8 (1992) 279–292.

[86] Y. LeCun, Y. Bengio, H. Geoffrey, Deep learning, Nature 521 (2015) 436–444.

[87] N. Coleman, D. Knorr, A. Ramadani, Statistical model to estimate the benefit of wake turbulence re-categorization, in: Thirteenth USA/Europe Air Traffic Management Research and Development Seminar (ATM2019), Vienna Austria, 2019.

[88] E. Gilbo P., Optimizing airport capacity utilization in air traffic flow management subject to constraints at arrival and departure fixes, IEEE Transactions on Control Systems Technology 5 (1997) 490–503.

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

[90] M. Samà, A. D'Ariano, D. Pacciarelli, Rolling horizon approach for aircraft scheduling in the terminal control area of busy airports, Procedia - Social and Behavioral Sciences 80 (2013) 531–552.

[91] M. Samà, A. D'Ariano, P. D'Ariano, D. Pacciarelli, Optimal aircraft scheduling and routing at a terminal control area during disturbances,

Transportation Research Part C: Emerging Technologies 47 (2014) 61–85.

[92] M. Samá, A. D'Ariano, K. Palagachev, M. Gerdts, Integration methods for aircraft scheduling and trajectory optimization at a busy terminal manoeuvring area, OR Spectrum 41 (2019) 641–681.

[93] U. Benlic, A. E.I. Brownlee, E. K. Burke, Heuristic search for the coupled runway sequencing and taxiway routing problem, Transportation Research Part C: Emerging Technologies 71 (2016) 333–355.

[94] J. Guépet, O. Briant, J.-P. Gayon, R. Acuna-Agost, Integration of aircraft ground movements and runway operations, Transportation Research Part E: Logistics and Transportation Review 104 (2017) 131–149.

[95] K. K. Ng, C. Lee, F. T. Chan, Y. Qin, Robust aircraft sequencing and scheduling problem with arrival/departure delay using the min-max regret approach, Transportation Research Part E: Logistics and Transportation Review 106 (2017) 115–136.

[96] A. Khassiba, F. Bastin, B. Gendron, S. Cafieri, M. Mongeau, Extended aircraft arrival management under uncertainty: A computational study, Journal of Air Transportation 27 (2019) 131–143.

[97] R. D. de Neufville, A. R. Odoni, Airport systems: Planning, design, and management, McGraw-Hill, 2003.

[98] The OpenSky Network, Open air traffic data for research, `https://opensky-network.org`, 2013. On line; accessed April 21, 2021.

[99] S. Ikli, Library of codes, instances and data sets for the aircraft landing problem (ALP), `http://data.recherche.enac.fr/ikli-alp/`, 2020. On line; accessed April 21, 2021.

[100] A. J. Cook, G. Tanner, S. Anderson, Evaluating the true cost to airlines of one minute of airborne or ground delay, Technical Report, Transport Studies Group, EUROCONTROL and University of Westminster, London, UK, 2004.

[101] A. J. Cook, G. Tanner, European airline delay cost reference values, Technical Report, Transport Studies Group, University of Westminster, London, UK, 2011.