

# Integration of aircraft ground movements and runway operations

---

## Abstract

The Ground Routing Problem focuses on finding the optimal routing of aircraft from parking stands to runways. The Runway Sequencing Problem consists in ordering the sequence of takes-offs and landings on runways. We study the integration of these two problems with the aim of simultaneously increasing runway efficiency and reducing taxi times. We propose a heuristic sequential approach based on a novel mathematical formulation. We test our methods using real data of a major European airport. Our approach significantly reduces the total completion and taxi times within reasonable computation times, making it viable to be used in daily operations.

*Keywords:* Ground routing, runway sequencing, Mixed integer programming

---

## 1. Introduction

This paper focuses on the management of the departure process, from push back to take-off, through an integration of the Ground Routing Problem (GRP) and the Runway Sequencing Problem (RSP). In this first section, we present briefly the two problems and their interactions. Then we present the research questions addressed in this work and the outline of the rest of the paper.

*RSP.* The RSP consists in sequencing runway operations while ensuring safety, i.e. deciding in which order (and when) each aircraft takes off, lands on or crosses a runway, while respecting minimum separation requirements. Depending on the airport layout, runway assignment can be also part of the problem. This problem is issued by Air Traffic Controllers (ATC) on an operational window of typically 10 to 40 minutes. Longer time windows can hardly be considered because of perturbations occurring the day of operations. Hence a sliding time window scheme is used in the literature and in practice. Consequently, the problem has to be solved very often and computation times are critical.

Minimum separation requirement is the main limiting factor of runway capacity. Because of wake vortex, an air mass is perturbed when it is crossed by an aircraft and a minimum separation time must be respected between two aircraft to ensure the safety of the second one. The heavier the leading aircraft, the bigger its wake vortex is and, thus, the separation time is longer. The lighter the trailing aircraft, the more it is subject to turbulence and the longer is the separation time. Aircraft are classified in wake vortex categories by the International Civil Aviation Organization (ICAO) and the minimum separation between two aircraft depends on their respective classes (see e.g. Table 1). Additional separations may

Time [s]		Trailing aircraft		
		H	M	L
Leading aircraft	H	90	120	120
	M	60	60	90
	L	60	60	60

Table 1: Minimum take-off separation times (H = heavy, M = medium and L = light)

be necessary to prevent conflicts in airspace segment of routes, also known as Standard Instrument Departure routes (SID). SID separation constraints depend upon the departure routes and speeds of the aircraft.

When an air sector of an aircraft flight plan is congested or when the destination airport is facing adverse conditions, the Network Manager Operations Center (NMOC) assigns a Calculated Take-Off Time (CTOT). It generally results in delaying the take-off to prevent the situation from NMOC worse in the perturbed sector. The take-off is allowed within the interval  $[CTOT - 5 \text{ min}; CTOT + 10 \text{ min}]$ , called a NMOC slot. Otherwise, the aircraft has to wait for another slot from the NMOC. For a more detailed description of constraints that have to be taken into account in take-off scheduling, we refer the reader to [Atkin et al. \(2009b\)](#).

The quality of a runway sequence can be evaluated with different criteria. The main criterion is the efficiency and the good use of the runway capacity. In the literature, it is often modeled by the runway throughput (makespan), the total (weighted) completion time or the total (weighted) deviation to targeted take-off or landing times. Equity is another important criterion, it is often measured by the deviation to the First Come First Serve order (FCFS, the fairest order) or the maximum delay.

*GRP.* The GRP consists in scheduling the movements of aircraft between airport facilities without conflicts and in the most effective way. An arriving aircraft has to be routed from its landing runway to its stand or hangar. A departing aircraft has to be routed from its current parking position to its departure runway. The ground movements occur on a network of roads called taxiways which link airport facilities. In practice this problem is issued by Air Traffic Controllers (ATCs) on an operational window of typically 10 to 40 minutes.

The main constraints of the problem are related to the safety of aircraft: as in airspace, aircraft have to be separated from each other to avoid collisions. Several other routing constraints must also be taken into account such as taxi speeds and acceleration for passengers comfort, turning angle and aircraft / taxiway segment compatibility due to weight or width. There are also many different conflicts in the stand area. Among them, the push back conflicts and the head-on conflicts between departures and arrivals are the most common ones (see [Figure 1](#)).

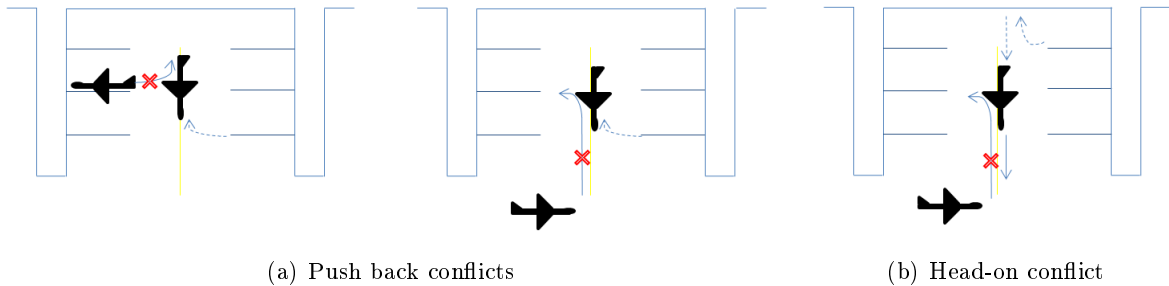


Figure 1: Common conflicts in the stand area

The quality of a routing schedule can be evaluated with different performance indicators. Among them the average taxi time and the average completion time are frequently used. The taxi time measures the time an aircraft spends on the ground with engines on, between push back (i.e. leaving the parking position) and take-off for a departure and between landing and park-in for an arrival. It includes any waiting time (e.g. runway queuing time) and not just the time spent moving, as engines cannot be turned off once started up. The completion time is the take-off time for departing flights and the park-in time for arriving flights.

For departures, the completion time and the taxi-time depend on the take-off sequence. Hence, the GRP and the RSP are intimately linked.

*Towards a better integration.* Two different managements of the departure process can be observed nowadays. In the first one, aircraft are pushed back as soon as possible and taxi to the runway where they can potentially be reordered to increase the runway capacity. The second practice proceeds sequentially in three steps. First, an earliest time at the runway is estimated for every aircraft through an estimation of taxi times. Then, the take-off sequence is optimized. Finally, ground movements (or only push back) are scheduled to match the predicted take-off sequence. This approach is recent and promoted by the Airport Collaborative Decision Making (A-CDM) project (Eurocontrol, 2009). It targets a better synchronization of ground movements and runway operations. It particularly allows to reduce runway queuing times through a better scheduling of push backs: aircraft can be held at their stand with engines off instead of waiting at the runway with engines on.

The aim of a better integration of the GRP and the RSP is to further improve this synchronization. The motivations are twofold: increasing runway efficiency and reducing taxi times. A-CDM approach relies on estimations of taxi times, which are particularly difficult to forecast. Nevertheless, the accuracy of these estimations is crucial. If taxi times are underestimated, aircraft are held too long, consequently creating idle time between take-offs and wasting runway capacity. On the contrary, if taxi times are overestimated, aircraft are not held long enough. Thus, an excessive queue will appear at the departure runway and fuel will be wasted. Furthermore, inaccurate taxi times can make the aircraft reach the runway in a different order from the desired one. Aircraft can be reordered at the runway but Atkin et al. (2009a) show that this reordering is constrained by the holding point layout and that not all sequences are feasible. A better synchronization leads to less reordering and potentially enables more efficient sequences.

*Research questions.* We mainly address two research questions in this paper. Is a better integration of runway sequencing and ground routing valuable? How can the integrated problem be solved efficiently? Indeed, computation times are critical since the GRP and the RSP should be solved on a rolling horizon time window of 10 to 40 minutes, with an update ideally at each event (up to twice per minute in peak hours).

The rest of the paper is organized as follows. A literature review is presented in Section 2 with a summary of our contributions. The problem is described in Section 3 with a formulation from the literature. A sequential approach is proposed in Section 4. Its principle is the same that the sequential approach of A-CDM, but a new runway sequencing approach is presented. The different methods are tested on Copenhagen Airport (CPH) layout in Section 7. We conclude and highlight some directions for future works in Section 8.

## 2. Literature review

For detailed literature reviews on the GRP and the RSP, we refer the reader respectively to Atkin et al. (2010); Guépet et al. (2016) and Bennell et al. (2011); Lieder et al. (2015). The RSP is generally recognized as a more critical problem than the GRP. We focus the rest of our literature review on papers that consider a partial or full integration of both problems.

*Full integration.* Deau et al. (2008, 2009) consider a sequential approach similar to the A-CDM one. A take-off sequencing problem is solved, which provides Target Take-Off Times (TTOTs) to a routing model. The take-off sequencing algorithm is a Branch & Bound algorithm (B&B) minimizing departure delay and deviation from NMOC slots.

Keith and Richards (2008) propose a Mixed Integer linear Program (MIP) directly integrating the RSP and the GRP in a single model. A weighted combination of the makespan, the average taxi time and the average taxi distance is minimized. The model is slightly adapted by Clare and Richards (2011) to improve computational efficiency. Nevertheless, computation times are too long: their model needs more than a minute to handle instances with eight aircraft on a small network with a single runway holding point.

Lee and Balakrishnan (2012) propose a simplified version of the model of Clare and Richards (2011) by restricting the routing possibility to a fix path. Computation times are still too long and they propose a sequential approach. The take-off sequencing is performed with the algorithm of Balakrishnan and Chandran (2010). The so-obtained TTOTs are provided to a routing model minimizing a linear combination of deviation to TTOTs and the total taxi time. The take-off sequence can still be changed in their routing model and several minutes are often necessary to solve a time window.

Bosson et al. (2014a) propose a three phase decomposition for sequencing take-offs and landings and scheduling airspace movements under uncertainty. Bosson et al. (2014b) additionally involve ground movements. They consider stochastic release times and due dates (estimated time of arrival and departure) through sampling (Sample Average Approximation) embedded in a 3-phases decomposition. The approach is computationally demanding since an important number of scenarios has to be considered. An instance of less than 15 aircraft is solved in 4 minutes. Besides, routing in the stand area is not considered.

*Partial integration.* Malik et al. (2010) also assume constant taxi times. They compute a take-off sequence with the reference MIP of the RSP (see Beasley et al. (2000)). Then, they deduce spot release times by subtracting the constant taxi times, which reduces runway queuing time. Jung et al. (2010, 2011) use the same approach but replace the runway optimization by an adaptation of the Dynamic Program (DP) of Rathinam et al. (2009).

Atkin et al. (2012) use a similar sequential approach but do not assume constant taxi times and consider stand contention during the design of the take-off sequence: sequences not allowing feasible push back times are pruned. Push back times are then reoptimized in a second phase. The take-off sequence is optimized with a complex heuristic which core algorithm is a B&B embedded in a rolling horizon scheme.

Atkin et al. (2004, 2007, 2009a) consider routing constraints in the holding point while optimizing the take-off sequence on a single runway. They propose a tabu search algorithm in which feasibility in the holding point is heuristically checked when a sequence is evaluated. Infeasible solutions are discarded.

Rathinam et al. (2009) also consider holding point constraints in a take-off sequencing problem on a single runway. Aircraft are pre-assigned to runway entry queues and a first come first serve order has to be respected inside each queue. Using this structure, the problem is efficiently solved by a Dynamic Program (DP) minimizing the total aircraft delay.

Kim et al. (2010) extend the MIP of the RSP (see Beasley et al. (2000)). Their model aims at minimizing total emissions in the Terminal Maneuvering Area (TMA, the airspace around the airport) through runway assignment and scheduling of take-off and landing. Constant taxi times (depending on gate / runway) are assumed, benefit in fuel consumption on the ground would thus be lessened if a routing optimization approach was considered (e.g. through stand holding).

*Summary of our contributions.* In this paper, we consider the integration of the RSP and the GRP on a single departure runway, with the objective of minimizing the total completion and taxi times. Landing times are assumed to be fixed inputs but our model can be easily adapted to include other runway operations.

The literature review reveals that there does not exist an efficient approach to solve this problem. The existing methods are not computationally efficient (Clare and Richards (2011), Lee and Balakrishnan (2012), Bosson et al. (2014b)). Sequential approaches have been proposed but the integration is rudimentary since ground routing is not taken into account when optimizing the take-off sequence (Deau et al. (2008, 2009)). On the contrary, Atkin et al. (2012) consider interactions through stand contention but do not use it to minimize fuel consumption during the runway sequencing, potentially missing more fuel efficient solutions. Moreover, the solutions of Atkin et al. (2012) are not compared to other approaches.

Our first contribution is to propose an efficient heuristic sequential algorithm based on an innovative formulation of the RSP including stand area conflicts. Several directions are also explored to improve the solving. A second contribution is to show that a better integration of RSP and GRP is highly valuable and significantly reduces the total completion and taxi times. A last by-product contribution is to improve the RSP discrete-time formulation of Beasley et al. (2000) through a reformulation of separation constraints. Our results are illustrated with numerical experiments based on Copenhagen Airport (CPH) layout.

### 3. The integrated runway sequencing and ground routing problem

The integrated runway sequencing and ground routing problem (I-RSP/GRP) consists in simultaneously scheduling the ground movements and the runway sequence. As many random events happen the day of operations and ready times are not accurately known long in advance, the I-RSP/GRP problem is issued on a relatively short time horizon of typically 10 to 40 minutes. This optimization horizon slides over time and a problem has to be solved each time an aircraft enters the system, i.e. each time a new landing is forecasted and each time a new departure is ready to push back.

In this section, we focus on solving the I-RSP/GRP on a time window for which ready times and landing times are accurately known. Finding the optimal solution for each time window does not guarantee optimality over the whole time horizon, as it will be explained in Section 7 when comparing different heuristic approaches.

*Problem description.* A departure (resp. arrival) has to be routed from its stand (resp. landing runway) to its take-off runway (resp. stand). Take-offs must also be scheduled. Landing times are inputs of the problem, as well as the stand allocation and the runway assignment. The objective is to minimize the completion times and taxi times, while respecting operational requirements of the GRP and the RSP. For a departure, the taxi time gathers all the time spent with engines on, from push back to take-off, whereas the completion time gathers all the time spent in the system, from the ready time to take-off. For an arrival, the taxi time and the completion time are identical and gather all the time between the landing and the park-in.

*Taxiway network and flight characteristics.*

- The taxiway network is modeled as a graph  $G = (V, E)$  with  $V$  the set of nodes and  $E$  the set of edges.
- The set of arriving and departing flights is  $\mathcal{F} = \mathcal{F}_{dep} \cup \mathcal{F}_{arr}$ .

- A flight  $i$  must be routed from its origin  $o_i$  to its destination  $d_i$  through a pre-determined path  $P_i = (o_i, u_2, \dots, u_{|P_i|-1}, d_i)$ . For departure  $i$ ,  $o_i$  is its stand and  $d_i$  is its take-off runway. For an arrival,  $o_i$  is its landing runway and  $d_i$  is its stand.
- $V_i \subset V$  and  $E_i \subset E$  are the set of nodes and edges that flight  $i$  uses (defined by path  $P_i$ ).
- A flight  $i$  can spend a minimum (maximum) time  $T_{iuv}^{min}$  ( $T_{iuv}^{max}$ ) on edges  $uv \in E_i$ , which can be directly computed from the minimum and maximum average speeds allowed on each edge of its path.
- Two flights  $i$  and  $j$  must have a minimum separation time  $S_{iju}$  at each node  $u \in V_i \cap V_j$  (if  $i$  uses node  $u$  first) to prevent collision. Note that if flights  $i$  and  $j$  use the same runway, the separation time includes wake vortex separation times at the runway.
- $\mathcal{G}$  is the set of stand blockages, i.e. the set of all flights pairs  $(i, j) \in \mathcal{F}_{dep} \times \mathcal{F}_{arr}$  assigned to the same stand, and whose slots at the stand overlap.
- A flight  $i$  is ready to leave its origin at time  $T_{o_i}$  and must reach its destination before  $L_{d_i}$ , which is necessary to prevent excessively unfair completion times.

In [Guépet et al. \(2016\)](#), we have shown that alternate path cannot significantly improve the performance indicators in CPH airport, while computation times increase exponentially with the number of possible paths. Hence, assuming a pre-determined path for each aircraft seems a reasonable assumption for the airport under consideration.

We will also use the following notations to denote the unimpeded taxi-time  $\sum_{\forall uv \in E_i} T_{iuv}^{min}$  for flight  $i$ , i.e. the minimum time necessary to taxi from stand to runway or vice-versa.

- $EXOT_i$  : unimpeded taxi-out time for a departure  $i \in \mathcal{F}_{dep}$ .
- $EXIT_i$  : unimpeded taxi-out time for an arrival  $i \in \mathcal{F}_{arr}$ .

*Decision variables and objective function.*

- $t_{iu} \in \mathcal{T}_{iu} = [e_{iu}, l_{iu}]$  the time when flight  $i$  reaches node  $u \in V_i$ . The bounds  $e_{iu}$  and  $l_{iu}$  are not inputs of our model but can be deduced from the taxiway network and flight characteristics (see [AppendixA](#) for computation details). They can also be useful to include CTOT constraints.
- $z_{iju} = 1$  if flight  $i$  arrives before flight  $j$  at node  $u \in V_i \cap V_j$ , 0 otherwise.

The objective function is the weighted sum of taxi times and completion times with  $c_i^{taxi}$  and  $c_i^{ct}$  the taxi and completion weights for flight  $i$ . In this paper, we chose to define  $c_i^{taxi} = 1$  and  $c_i^{ct} = 2$  for all flights  $i$ . A more detailed discussion of the impact of these coefficients on industry indicators can be found in [Guépet et al. \(2016\)](#).

*MIP formulation.* The I-RSP/GRP can be formulated as follows.

$$\begin{aligned}
(MIP) \quad & \min \sum_{i \in \mathcal{F}} c_i^{taxi} (t_{id_i} - t_{io_i}) + \sum_{i \in \mathcal{F}} c_i^{ct} (t_{id_i} - T_{o_i}) & (1) \\
& t_{io_i} \leq t_{jd_j} & \forall (i, j) \in \mathcal{G} & (2) \\
& T_{iuv}^{min} \leq t_{iv} - t_{iu} \leq T_{iuv}^{max} & \forall i \in \mathcal{F}, \forall uv \in E_i & (3) \\
& z_{iju} + z_{jiu} = 1 & \forall i, j \in \mathcal{F}, \forall u \in V_i \cap V_j & (4) \\
& z_{iju} = z_{ijv} & \forall i, j \in \mathcal{F}, \forall uv \in E_i \cap E_j & (5) \\
& t_{ju} \geq t_{iu} + S_{iju} - M_{iju}(1 - z_{iju}) & \forall i, j \in \mathcal{F}, \forall u \in V_i \cap V_j & (6) \\
& t_{iu} \in \mathcal{T}_{iu} & \forall i \in \mathcal{F}, \forall u \in V_i & (7) \\
& z_{iju} \in \{0, 1\} & \forall i \neq j \in \mathcal{F}, \forall u \in V_i \cap V_j & (8)
\end{aligned}$$

Constraints (2) prevent stand blockages. Constraints (3) ensure the respect of speed limitations. Constraints (4) ensure that either  $i$  uses node  $u$  before  $j$  or the opposite. It ensures the definition of variables  $z_{iju}$ . Constraints (5) prevent overtake and head-on conflicts on a single edge, which are illustrated in figures 2(b) and 2(c). Constraints (6) ensure the separation of aircraft at every node of the taxiway network as illustrated in Figure 2(a), where  $M_{iju}$  can be defined as  $l_{iu} + S_{iju} - e_{ju}$ .

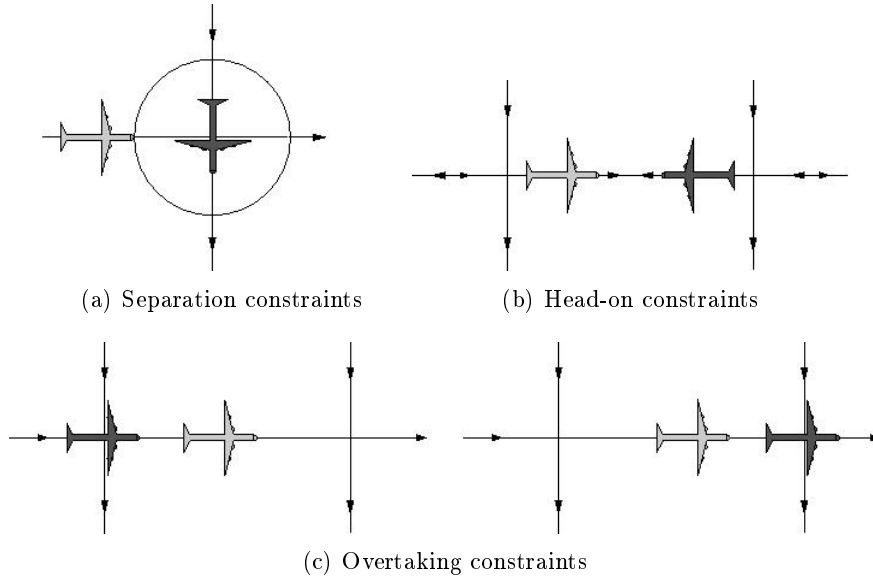


Figure 2: Safety constraints

Note that for two flights  $i$  and  $j$ , if  $l_{iu} \leq e_{ju}$  at node  $u \in V_i \cap V_j$ , then  $z_{iju}$  has to be equal to 1. These constraints can be added to strengthen the model and one of the two separation constraints (6) can be removed. Furthermore, if  $l_{iu} + S_{iju} \leq e_{ju}$ , then the separation is naturally forced. The associated variables  $z_{iju}$ ,  $z_{jiu}$  and constraints involving them can be removed. Constraints (7) ensure the respect of the slot at each node. Note that it forces arrival  $i$  to start taxiing as soon as the landing is completed since  $e_{io_i} = l_{io_i} = T_{o_i}$ .

*Fairness.* We do not consider fairness issues in this paper. A simple way to include a fairness criterion is to include in the MIP a third part in the objective function in order to minimize the maximum completion time (including or not the taxi time).

Our MIP is a generalization of the one presented for the GRP in Guépet et al. (2016) (by relaxing the runway sequence) but also of the one presented for the RSP in Beasley et al. (2000). This last MIP already offers poor performances because of a weak linear relaxation due to the big  $M$  constraints (6).  $M_{iju}$  cannot be further reduced without loss of generality and we did not succeed in strengthening this model. Consequently, we propose a heuristic approach in the next section.

#### 4. A heuristic sequential approach

Following Lee and Balakrishnan (2012), we consider a sequential approach that proceeds in three steps.

**Step 1.** Estimate the earliest runway times for departure flights and also the earliest stand times for arrival flights.

**Step 2.** Sequence take-offs using the earliest times provided by Step 1.

**Step 3.** Route aircraft in the taxiway network based on the take-off sequence obtained in Step 2.

This sequential scheme is convenient for airports since it decomposes the whole problem in three modules that can be changed independently from each other. Existing systems can be reused. For instance, most of big airports already have a runway optimization advisory system. This scheme is also compatible with the division of roles between the ground controller (ATC) managing the taxiway network and the runway controller ATC managing the runway. Finally, this sequential scheme is in line the A-CDM project.

We now provide some additional details for each step.

*Step 1.* As in Lee and Balakrishnan (2012), we estimate the earliest runway time for a departure flight by adding the unimpeded taxi-out time to the scheduled pushback time. In addition, we estimate the earliest stand time for an arrival flight by adding the unimpeded taxi-in time to the estimated landing time.

*Step 2.* We consider three different approaches using more or less information:

- (a) Sequence take-offs according to a First Come First Served (FCFS) policy, i.e. by increasing order of earliest runway times (provided by Step 1).
- (b) Sequence take-offs with the objective to minimize completion times, while respecting runway separation constraints. The problem is formulated with an IP formulation, denoted by  $(IP^r)$ , of the RSP adapted from Beasley et al. (2000). See Section 5 for more details.
- (c) Sequence take-offs with a new IP formulation, denoted by  $(IP^{rs})$ , of the RSP which takes into account runway separation constraints and stand conflicts. The objective function includes completion times and taxi times. See Section 6 for more details.

*Step 3.* The aircraft routing in the taxiway network is performed with the single path MIP formulation of the GRP presented in Guépet et al. (2016). This formulation is similar to (1)-(8) except that the take-off sequence computed during Step 2 is forced with additional constraints ( $z_{iju} = 1$  at runway  $u$  if flight  $i$  takes off before  $j$ ). It provides an optimized detailed schedule of ground movements, i.e. a time at each node for every aircraft, such that it respects the take-off sequence decided during Step 2. The



real interest of Step 3 is that it provides an accurate estimation of push back and take-off times, which can be different from the take-off times computed in Step 2 when building the sequence.

We now briefly discuss the computational complexity of each step. Step 2(b) can be solved very quickly for a large number of aircraft, either with an IP formulation (see next section) or by dynamic programming (see e.g. [Balakrishnan and Chandran \(2010\)](#)). Step 2(c) will be discussed in detail in sections 6 and 7. Step 3 can be solved in a few seconds for the instances under consideration ([Guépet et al., 2016](#)).

## 5. A stronger formulation of the RSP (step 2.b)

In this section, we focus on Step 2.b of the sequential approach. We consider the problem of sequencing departures on a single runway. with the objective to minimize the total weighted completion time, while respecting runway separation times. It was originally formulated by [Beasley et al. \(2000\)](#) in continuous and discrete time. In what follows, we remind these formulations and also present a reformulation of separation constraints using clique constraints introduced by [Fahle et al. \(2003\)](#). Then we present a stronger formulation of separation constraints based on wake vortex categories.

As we consider only departure flights at the runway node, we introduce some simpler notations :

- $S_{ij}^r$  : minimum runway separation time between flights  $i$  and  $j$  (equal to  $S_{ijd_i}$  when  $d_i$  is the runway node)
- $\mathcal{T}_i^r$  : set of possible take-off times for a departure  $i$  (equal to  $\mathcal{T}_{id_i}$  when  $d_i$  is the runway node)

### 5.1. Formulations from the literature

*Continuous time.* The problem of minimizing completion times while respecting runway separation times can be formulated with the following MIP adapted from [Beasley et al. \(2000\)](#).

$$\min \sum_{i \in \mathcal{F}_{dep}} c_i^{ct} (t_i - T_{o_i}) \quad (9)$$

$$(MIP^r) \quad s.t. \quad t_i \geq t_j + S_{ij}^r - M(1 - z_{ij}) \quad \forall i, j \in \mathcal{F}_{dep} \quad (10)$$

$$z_{ij} + z_{ji} = 1 \quad \forall i, j \in \mathcal{F}_{dep} \quad (11)$$

$$t_i \in \mathcal{T}_i^r \quad \forall i \in \mathcal{F}_{dep} \quad (12)$$

$$z_{ij} \in \{0, 1\} \quad \forall i, j \in \mathcal{F}_{dep}. \quad (13)$$

One may remark that separation constraints (10) are modeled with so-called *big M* constraints, which lead to a weak linear relaxation and poor computational performances.

*Discrete time.* A time discretization avoids these *big M* constraints. We discretize the take-off slots  $\mathcal{T}_i^r$  with a step of 30 seconds. In practice, earliest times at the runway are estimated with a precision of 1 minute and separation times of Table 1 with a precision of 30 seconds. Hence, a discretization step of 30 seconds ensures that an optimal solution of the discrete formulation is optimal for the continuous time formulation.

Let  $x_{it}^r = 1$  if flight  $i \in \mathcal{F}_{dep}$  takes off at time  $t \in \mathcal{T}_i^r$  and 0 otherwise. The discrete time formulation, adapted from [Beasley et al. \(2000\)](#), follows:

$$\min \sum_{i \in \mathcal{F}_{dep}} \sum_{t \in \mathcal{T}_i^r} \tilde{c}_{it}^r x_{it}^r \quad (14)$$

$$s.t. \quad \sum_{t \in \mathcal{T}_i^r} x_{it}^r = 1 \quad \forall i \in \mathcal{F}_{dep} \quad (15)$$

$$x_{it}^r + x_{jt'}^r \leq 1 \quad \forall i, j \in \mathcal{F}_{dep}, \forall t \in \mathcal{T}_i^r, \forall t' \in \mathcal{T}_j^r, t \leq t' < t + S_{ij}^r \quad (16)$$

$$x_{it}^r \in \{0, 1\} \quad \forall i \in \mathcal{F}_{dep}, \forall t \in \mathcal{T}_i^r \quad (17)$$

where  $\tilde{c}_{it}^r = c_i^{ct}(t - T_{o_i})$  models the total completion time in objective function (14). Constraints (15) ensure that flights take-off once and only once in their slot. Constraints (16) ensure minimum separation times as illustrated below.


Fahle et al. (2003) have reformulated the separation constraints (16) with clique constraints as follows :

$$x_{it}^r + \sum_{\substack{t' \in \mathcal{T}_j^r \\ t - S_{ji}^r < t' < t + S_{ij}^r}} x_{jt'}^r \leq 1 \quad \forall i, j \in \mathcal{F}_{dep}, \forall t \in \mathcal{T}_i^r \quad (18)$$

## 5.2. Reformulation of separation constraints using wake vortex categories

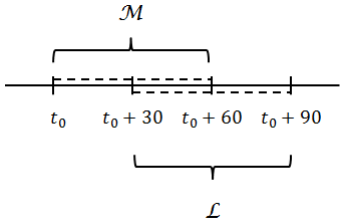
We propose a stronger formulation based on wake vortex categories. Our reformulation is presented for the minimum take-off separation times of Table 1, but a similar approach could be used for other separation standards or for SID separation constraints. We respectively denote  $\mathcal{H}$ ,  $\mathcal{M}$ ,  $\mathcal{L}$  the set of flights operated by heavy, medium and light aircraft.

*Clique based on the minimum separation time.* Two aircraft must be separated by at least 60 seconds. Consequently, the following inequalities are valid for all couples of aircraft and cover the minimum separation times for couples M/H, M/M, L/H, L/M, L/L.



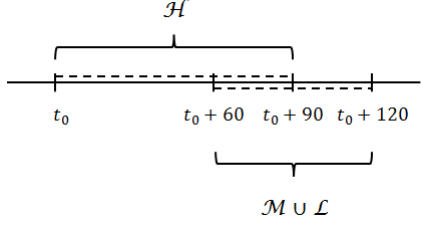
$$\sum_{i \in \mathcal{F}_{dep}} \sum_{\substack{t \in \mathcal{T}_i^r \\ t_0 \leq t < t_0 + 60}} x_{it}^r \leq 1 \quad \forall t_0 \in \bigcup_{i \in \mathcal{F}_{dep}} \mathcal{T}_i^r \quad (19)$$

*Clique based on the medium aircraft category.* The minimum separation times for M/M, L/L, and M/L are respectively 60, 60 and 90 seconds. Consequently, the following inequalities are valid and, with inequalities (19), they additionally cover couples M/L.



$$\sum_{i \in \mathcal{M}} \sum_{\substack{t \in \mathcal{T}_i^r \\ t_0 \leq t < t_0 + 60}} x_{it}^r + \sum_{i \in \mathcal{L}} \sum_{\substack{t \in \mathcal{T}_i^r \\ t_0 + 30 \leq t < t_0 + 90}} x_{it}^r \leq 1 \quad \forall t_0 \in \bigcup_{i \in \mathcal{F}_{dep}} \mathcal{T}_i^r \quad (20)$$

*Clique based on the heavy aircraft category.* The minimum separation for H/H, H/M and H/L are respectively 90, 120 and 120 seconds. Consequently, the following inequalities are valid and, with inequalities (19), they cover minimum separation for couples H/H, H/M, H/L.



$$\sum_{i \in \mathcal{H}} \sum_{\substack{t \in \mathcal{T}_i^r \\ t_0 \leq t < t_0 + 90}} x_{it}^r + \sum_{i \in \mathcal{M} \cup \mathcal{L}} \sum_{\substack{t \in \mathcal{T}_i^r \\ t_0 + 60 \leq t < t_0 + 120}} x_{it}^r \leq 1 \quad \forall t_0 \in \bigcup_{i \in \mathcal{F}_{dep}} \mathcal{T}_i^r \quad (21)$$

In the end, inequalities (19)-(21) form a reformulation of separation constraints (16). In the rest of the paper,  $(IP^r)$  will denote the IP including these reformulations.

$$(IP^r) \quad \min \quad \sum_{i \in \mathcal{F}_{dep}} \sum_{t \in \mathcal{T}_i^r} \tilde{c}_{it}^r x_{it}^r$$

*s.t.* (15), (19) – (21), (17)

Several other inequalities could be derived from Table 1, but only those presented appeared to be significantly effective.

### 5.3. Comparison of formulations

In his PhD thesis, Guepet (2015) provides a detailed numerical study comparing the previous formulations. Based on randomly generated instances representing different flight mixes in peak hours, he shows that discrete time formulations outperforms the continuous time formulation. Surprisingly, the reformulation of Fahle et al. (2003) appears to be less efficient than the discrete time model of Beasley et al. (2000), probably because of the presolve of Cplex solver. On the contrary, the reformulation based on wake vortex categories is shown to be the most efficient technique: computation times are significantly reduced and are short enough for a direct industrial application.

Note that very efficient dynamic programming approaches can be used to solve the RSP (see e.g. Balakrishnan and Chandran (2010)). However these approaches cannot be generalized to integrate stand area conflicts, which is the purpose of the next section.

## 6. Integrating stand area conflicts (step 2.c)

In this section, we extend  $(IP^r)$  to integrate stand area conflicts and taxi time indicator. At Copenhagen airport, Guépet et al. (2016) have shown that the main constraints of the problem are the conflicts in the stand area and the separation times at the runway. On the other hand, conflicts between stands and runways can be neglected. Based on these observations, we assume especially in what follows that:

- Aircraft can be routed at maximum speed between runways and stands, within time  $EXOT_i$  ( $EXIT_i$ ) for a departure (arrival)  $i$ .
- To prevent conflicts between two flights  $i$  and  $j$  in the stand area (see Figure 1), we define  $a_{ij} \leq b_{ij} \in \mathbb{R}$  such that if  $i$  pushes back or parks in at time  $t$ , then  $j$  cannot push back or parks in during the interval  $[t + a_{ij}, t + b_{ij}]$ . See AppendixB for details on how to compute  $a_{ij}$  and  $b_{ij}$ .

In what follows, we first present an extension of  $(IP^r)$  that takes into account stand area conflicts (denoted by  $(IP^{rs})$ ). Then we present some techniques to speed up computation.

### 6.1. Discrete time formulation

We introduce (or remind) some notations:

- $x_{it}^r = 1$  if flight  $i$  takes-off or lands at time  $t$  and 0 otherwise.
- $x_{it}^s = 1$  if flight  $i$  pushes back or parks in at time  $t$  and 0 otherwise.
- $\mathcal{T}_i^r$  : set of possible take off (landing) times for a departure (arrival)  $i$ . The landing set for an arrival is a singleton as landing times are an input of our model.
- $\mathcal{T}_i^s$  : set of possible push back (park-in) times for a departure (arrival)  $i$ .

We can now formulate an IP than takes into account stand area conflicts and taxi time.

$$\min \sum_{i \in \mathcal{F}} \sum_{t \in \mathcal{T}_i^r} c_{it}^r x_{it}^r + \sum_{i \in \mathcal{F}} \sum_{t \in \mathcal{T}_i^s} c_{it}^s x_{it}^s \quad (22)$$

$$(IP^{rs}) \quad s.t. \quad (15), (19) - (21)$$

$$\sum_{t \in \mathcal{T}_i^r} x_{it}^r = 1 \quad \forall i \in \mathcal{F}_{arr} \quad (23)$$

$$\sum_{t \in \mathcal{T}_i^s} x_{it}^s = 1 \quad \forall i \in \mathcal{F} \quad (24)$$

$$\sum_{t \in \mathcal{T}_i^r} tx_{it}^r - \sum_{t \in \mathcal{T}_i^s} tx_{it}^s \geq EXOT_i \quad \forall i \in \mathcal{F}_{dep} \quad (25)$$

$$\sum_{t \in \mathcal{T}_i^s} tx_{it}^s - \sum_{t \in \mathcal{T}_i^r} tx_{it}^r \geq EXIT_i \quad \forall i \in \mathcal{F}_{arr} \quad (26)$$

$$\sum_{t \in \mathcal{T}_i^s} tx_{it}^s \leq \sum_{t \in \mathcal{T}_j^s} tx_{jt}^s \quad \forall (i, j) \in \mathcal{G} \quad (27)$$

$$x_{it}^s + x_{jt'}^s \leq 1 \quad \forall i, j \in \mathcal{F}, \forall t \in \mathcal{T}_i^s, \forall t' \in \mathcal{T}_j^s, t + a_{ij} \leq t' \leq t + b_{ij} \quad (28)$$

$$x_{it}^r \in \{0, 1\} \quad \forall i \in \mathcal{F}, \forall t \in \mathcal{T}_i^r \quad (29)$$

$$x_{it}^s \in \{0, 1\} \quad \forall i \in \mathcal{F}, \forall t \in \mathcal{T}_i^s \quad (30)$$

To model objective function (1) of (MIP), we set the objective coefficients to

$$\begin{aligned} c_{it}^r &= tc_i^{ct} + tc_i^{taxi} & \forall i \in \mathcal{F}_{dep}, \forall t \in \mathcal{T}_i^r & & c_{it}^r &= -T_{o_i} c_i^{ct} - tc_i^{taxi} & \forall i \in \mathcal{F}_{arr}, \forall t \in \mathcal{T}_i^r \\ c_{it}^s &= -T_{o_i} c_i^{ct} - tc_i^{taxi} & \forall i \in \mathcal{F}_{dep}, \forall t \in \mathcal{T}_i^s & & c_{it}^s &= tc_i^{ct} + tc_i^{taxi} & \forall i \in \mathcal{F}_{arr}, \forall t \in \mathcal{T}_i^s \end{aligned}$$

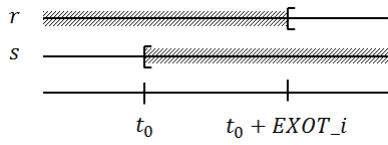
Constraints (15), (23) and (24) ensure that one and only one runway time and stand time is assigned to every flight. Constraints (25) and (26) ensure that minimum taxi-out and taxi-in times are respected. Constraints (27) prevent stand blockages. Constraints (28) guarantee minimum stand separation times. Other constraints ensure the definition of the variables.

According to our assumptions,  $\mathcal{T}_i^r$  is a singleton if  $i$  is an arrival. Hence variable  $x_{it}^r$  and related constraints can be removed for arrivals. Nevertheless, it is possible to generalize our formulation to optimize landings, by considering larger landing sets  $\mathcal{T}_i^r$  and by adding runway separation constraints similar to (16) or (19)-(21).

### 6.2. Reformulation of taxi time and stand blockages constraints

We now propose reformulations of constraints involving time in the coefficients, i.e (25)-(27). These reformulations will be shown to improve computation times in Section 7.

If departure  $i$  pushes back after  $t_0$  then it cannot take off before  $t_0 + EXOT_i$  and reciprocally. Hence taxi-out time constraints (25) can be reformulated as

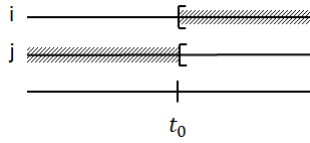


$$\sum_{\substack{t \in \mathcal{T}_i^s \\ t \geq t_0}} x_{it}^s + \sum_{\substack{t \in \mathcal{T}_i^r \\ t < t_0 + EXOT_i}} x_{it}^r \leq 1 \quad \forall i \in \mathcal{F}_{dep}, \forall t_0 \in \mathcal{T}_i^s. \quad (31)$$

The same reformulation works for taxi-in time constraints (26):

$$\sum_{\substack{t \in \mathcal{T}_i^r \\ t \geq t_0}} x_{it}^r + \sum_{\substack{t \in \mathcal{T}_i^s \\ t < t_0 + EXIT_i}} x_{it}^s \leq 1 \quad \forall i \in \mathcal{F}_{arr}, \forall t_0 \in \mathcal{T}_i^r. \quad (32)$$

Similarly, if departure  $i$  pushes back after  $t_0$ , then arrival  $j$  cannot park-in before  $t_0$  and reciprocally. Thus stand blockages constraints (27) can be reformulated as:



$$\sum_{\substack{t \in \mathcal{T}_i^s \\ t \geq t_0}} x_{it}^s + \sum_{\substack{t \in \mathcal{T}_j^s \\ t < t_0}} x_{jt}^s \leq 1 \quad \forall (i, j) \in \mathcal{G}, \forall t_0 \in \mathcal{T}_i^s \cap \mathcal{T}_j^s. \quad (33)$$

### 6.3. Filtering stand separation constraints

We now propose a heuristic filtering of stand separation constraints (28). Consider two medium aircraft parked in the same area of the airport. Their taxi-out times ( $EXOT_i$ ) are similar. As the runway separation time is 60 seconds (see Table 1), they are likely to be separated by 60 seconds in all the taxiway network. Based on this remark, we remove the stand separation constraints that are shorter than the runway separation constraints.

This filtering is heuristic as taxi-out times may vary because of conflicts with other aircraft. However, it is a good heuristic as the main conflicts occur at the runway and at the stands, and not in the intermediate taxiways. This will be shown in the numerical study.

### 6.4. Using the solution of previous time window

We can exploit the fact that the problem is addressed continuously through a sliding time window. It seems reasonable that push backs scheduled in the near future during the previous time window will still be scheduled in the near future in the current time window. This idea can be used to heuristically reduce sets  $\mathcal{T}_i^r$  and  $\mathcal{T}_i^s$  as follows.

Let  $i \in \mathcal{F}_{dep}$  a departure that was in the previous time window. Let  $t_{io_i}^*$  and  $t_{id_i}^*$  the push back and take-off times that were computed in Step 3. We reduce slots  $\mathcal{T}_{io_i}$  and  $\mathcal{T}_{id_i}$  as follows, which consequently reduces sets  $\mathcal{T}_i^s$  and  $\mathcal{T}_i^r$

$$\begin{aligned} e_{io_i} &\leftarrow \max\{e_{io_i}, t_{io_i}^* - \Delta_i\} & e_{id_i} &\leftarrow \max\{e_{id_i}, t_{id_i}^* - \Delta_i\} \\ l_{io_i} &\leftarrow \min\{l_{io_i}, t_{io_i}^* + \Delta_i\} & l_{id_i} &\leftarrow \min\{l_{id_i}, t_{id_i}^* + \Delta_i\} \end{aligned}$$

where, assuming that current time window starts at time 0 (within a translation),

$$\Delta_i = \begin{cases} 1 \text{ minute} & \text{if } 0 \leq t_{io_i}^* < 5 \text{ minutes} \\ 3 \text{ minutes} & \text{if } 5 \leq t_{io_i}^* < 10 \text{ minutes} \\ 5 \text{ minutes} & \text{if } 10 \leq t_{io_i}^* < 15 \text{ minutes} \\ 10 \text{ minutes} & \text{if } 15 \leq t_{io_i}^* < 20 \text{ minutes} \\ +\infty & \text{otherwise} \end{cases}$$

The same idea is used for arrivals.

The reduction of slots is illustrated in Figure 3, consists in almost fixing the flights with the earliest scheduled push back / park-in while giving more flexibility to further flights. Note that take-off time windows are also reduced according to the proximity of the scheduled push back time. It allows to reduce the number of variables and constraints but the quality of the solution can be deteriorated.

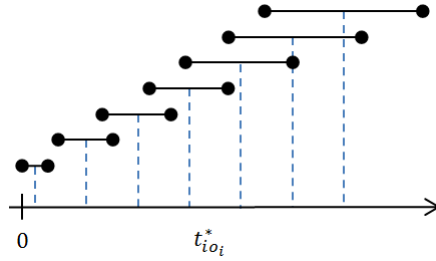


Figure 3: Using previous solution to reduce intervals  $[e_{iu}, l_{iu}]$

## 7. Numerical experiments

In this section, we compare the different approaches discussed previously on instances based on Copenhagen airport (CPH) layout.

### 7.1. Sliding time window scheme

Real operations are simulated with a sliding time window scheme. The length of the time window is set to 20 minutes. Each optimization over a single time window will be referred to as an iteration. An iteration is performed each time a new aircraft enters the system, i.e. each time the ready time at origin  $e_{io_i}$  of an aircraft enters the optimization time window. We repeat the process over a time horizon of one hour.

We consider four sliding time window algorithms :

- **MIP**: For each iteration, solve (*MIP*)
- **FCFS**: For each iteration, run the sequential algorithm using *FCFS* policy in Step 2
- **IP<sup>r</sup>**: For each iteration, run the sequential algorithm using (*IP<sup>r</sup>*) in Step 2
- **IP<sup>rs</sup>**: For each iteration, run the sequential algorithm using (*IP<sup>rs</sup>*) in Step 2

Reformulation of runway separation constraints (19)-(21) will always be used for **IP<sup>r</sup>** and **IP<sup>rs</sup>**. Other improvements of **IP<sup>rs</sup>** will be discussed in Section 7.4.

Solutions provided by the different algorithms are compared on the whole time horizon (one hour long). Let  $\mathcal{F}^{all}$  be the set of flights in the whole time horizon. If we denote by  $t_{io_i}^*$  and  $t_{id_i}^*$  the origin and destination times decided by an algorithm for flight  $i$ , then the cost of a solution is, consistently with objective function (1),

$$\sum_{i \in \mathcal{F}^{all}} c_i^{taxi} (t_{id_i}^* - t_{io_i}^*) + \sum_{i \in \mathcal{F}^{all}} c_i^{ct} (t_{id_i}^* - T_{o_i}).$$

All the approaches are heuristic, even **MIP** which provides the optimal solution on a time window but does not guarantee optimality on the whole time horizon. The gap reported in the next sections is the one with the best known solution on the whole time horizon (not systematically **MIP**).

## 7.2. Instances and test environment

*Copenhagen airport layout.* Figure 4 presents the graph of CPH taxiway network in the 22 runway configuration (the most frequent runway configuration), i.e. with take-offs on runway 22R (R=Right) and landings on runway 22L (L=Left). An edge represents an elementary taxiway segment. A node needs to be defined for each taxiway intersection. There is also a node for each stand. The graph is composed of 93 nodes and 235 edges. The standard path between each stand and each runway was provided by the airport, as well as the standard push back scheme and its duration, for each stand. We assume a maximal speed of 15 m/s for the taxiways around the runway (in blue in Figure 4), of 5 m/s for the taxiways around the stands (in red in Figure 4) and of 10 m/s for the other taxiways. A minimum (average) speed of 2 m/s is assumed on every edges. Note that a full stop of 4 minutes and a move at 10 m/s during 1 minute would respect the speed limit constraint, with an average speed of 2m/s. The minimum separation time ( $S_{iju}$ ) between two aircraft is assumed to be 40 seconds for every nodes (except the runways, see Table 1), which guarantees a minimum separation of 80 meters between aircraft. There are 95 stands (denoted by letters in Figure 4). The set  $\mathcal{G}$  is empty for instances involving only departures. When including also departures, its cardinality remains small since only a few pairs of conflicting departure/arrival flights are assigned to the same stand.

More details on the airport layout and characteristics can be found in [Guépet et al. \(2016\)](#).

*Traffic.* Instances of 1 hour are generated to represent intense and very intense departure peaks. More precisely, we consider 8 data sets being combinations of the following parameters:

- Number of departures : 40 or 60
- Number of arrivals : 0 or 20
- Flight mix (Light/Medium/Heavy in %) = 5-90-5 or 10-70-20. The first flight mix is representative of CPH traffic and some other European airports. The second flight mix is more representative of large US airports ([Lee and Balakrishnan, 2012](#)).

For each of this data set, we randomly generate 10 instances according to the following protocol :

- Ready times are uniformly distributed with a precision of 1 minute, then slots at each node  $[e_{iu}, l_{iu}]$  are deduced with the equations presented in [Appendix A](#). To ensure fairness, the maximum completion time at each node is also limited to 30 minutes ( $l_{iu} \leftarrow \min\{l_{iu}, e_{iu} + 30 \text{ min}\}$ ).
- Stands are randomly allocated for departures first and such that two departures do not use the same stand (which is rarely the case in the same hour). Then stands are randomly allocated to arrivals such that, on each stand, there are at least 35 minutes between an arrival and a consecutive departure to respect minimum turnaround times.

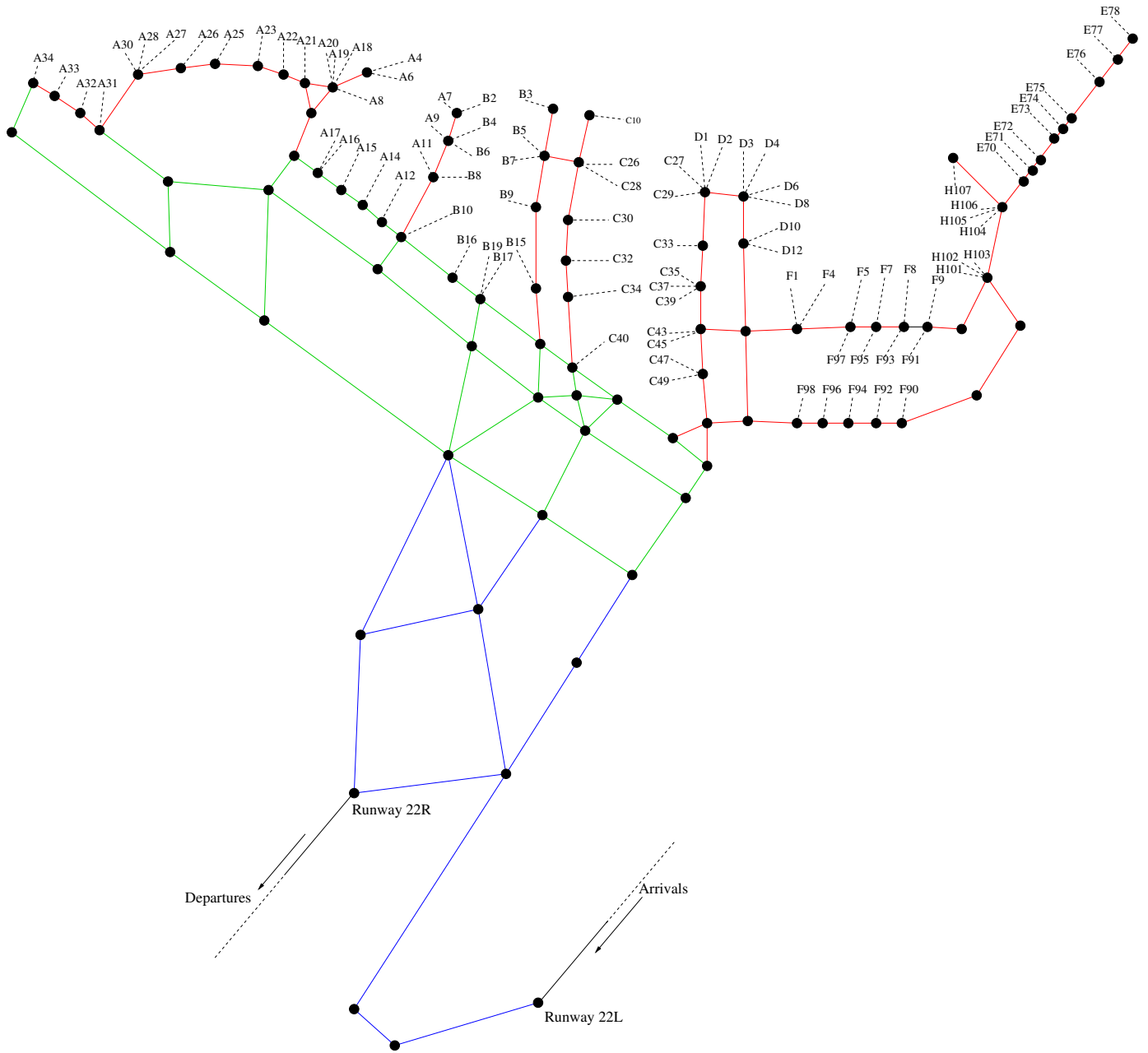


Figure 4: The taxiway network when runways are operated in 22 mode

Objective coefficients  $c_i^{ct}$  and  $c_i^{taxi}$  are respectively set to 2 and 1 to emphasize efficiency against taxi times, while not making taxi times negligible (see Guépet et al. (2016) for more details on the relationship between indicators).

*Computer configuration.* All MIPs were solved with Cplex 12.4 through Java Concert API on a personal computer (Intel Core i5-2400 3.10 Ghz, 4Go RAM) under Ubuntu 12.04 LTS. Default parameter tuning was used with a time limit of 300 seconds for all instances.



### 7.3. Comparison of algorithms

Table 2 presents the average and maximum gaps to the best known solutions over all instances. We remind that the best known solutions are not necessarily provided by **MIP**. When the time limit of 300 seconds is reached, **IP<sup>rs</sup>** sometimes provides better solutions than **MIP**. Moreover, because of the sliding time window, an exact formulation does not guarantee to find the best solution over all the one-hour instance.

		Flight mix 5-90-5				Flight mix 10-70-20			
		<b>FCFS</b>	<b>IP<sup>r</sup></b>	<b>IP<sup>rs</sup></b>	<b>MIP</b>	<b>FCFS</b>	<b>IP<sup>r</sup></b>	<b>IP<sup>rs</sup></b>	<b>MIP</b>
40 dep.	Avg	6.46 %	4.97 %	0.51 %	<b>0 %</b>	7.02 %	<b>3.63 %</b>	0.16 %	<b>0 %</b>
0 arr.	Max	12.9 %	9.31 %	0.90 %	<b>0 %</b>	13.9 %	<b>9.37 %</b>	0.63 %	<b>0 %</b>
40 dep.	Avg	6.38 %	5.19 %	0.60 %	<b>0.06 %</b>	6.71 %	3.20 %	0.92 %	<b>0 %</b>
20 arr.	Max	12.2 %	13.9 %	1.45 %	<b>0.55 %</b>	17.2 %	6.09 %	3.15 %	<b>0 %</b>
60 dep.	Avg	11.9 %	6.45 %	0.46 %	<b>0.10 %</b>	20.5 %	9.84 %	<b>0.18 %</b>	0.89 %
0 arr.	Max	18.3 %	11.8 %	1.17 %	<b>0.91 %</b>	29.2 %	21.1 %	<b>1.11 %</b>	2.17 %
60 dep.	Avg	14.7 %	8.63 %	0.68 %	<b>0.23 %</b>	19.2 %	9.54 %	1.11 %	<b>0.97 %</b>
20 arr.	Max	26.0 %	14.5 %	3.33 %	<b>0.96 %</b>	27.0 %	16.6 %	2.35 %	<b>2.15 %</b>

Max= maximum gap over all instances

Avg= average gap over all instances

Table 2: Gap to the best known solution

Table 3 presents the average and maximum computation times over all iterations. It also presents the percentage of iterations that were solved before the time limit.

		Flight mix 5-90-5				Flight mix 10-70-20			
		<b>FCFS</b>	<b>IP<sup>r</sup></b>	<b>IP<sup>rs</sup></b>	<b>MIP</b>	<b>FCFS</b>	<b>IP<sup>r</sup></b>	<b>IP<sup>rs</sup></b>	<b>MIP</b>
40 dep.	Max	<0.1s	0.2s	24s	300s	<0.1s	1.0s	4.0s	300s
	Avg	<0.1s	<0.1s	0.8s	6.8s	<0.1s	<0.1s	0.4s	20s
0 arr.	% solved	100 %	100 %	100 %	99.3 %	100 %	100 %	100 %	95.3 %
40 dep.	Max	<0.1s	0.3s	46s	300s	<0.1s	0.4s	8.5s	300s
	Avg	<0.1s	<0.1s	1.7s	44s	<0.1s	<0.1s	0.9s	23s
20 arr.	% solved	100 %	100 %	100 %	87.5 %	100 %	100 %	100 %	96.0 %
60 dep.	Max	<0.1s	1.8s	300s	300s	<0.1s	5.5s	300s	300s
	Avg	<0.1s	0.1s	12s	262s	<0.1s	0.8s	41s	267s
0 arr.	% solved	100 %	100 %	99.0 %	15.9 %	100 %	100 %	94.1 %	12.2 %
60 dep.	Max	<0.1s	3.2s	107s	300s	<0.1s	5.3s	300s	300s
	Avg	<0.1s	0.2s	8.6s	270s	<0.1s	0.9s	80s	292s
20 arr.	% solved	100 %	100 %	100 %	11.3 %	100 %	100 %	91.8 %	3.6 %

Max= maximum computation time over all iterations

Avg= average computation time over all iterations

% solved= percentage of iterations solved before the time limit of 300 s

Table 3: Computation times

Two main results come out from these tables. Firstly, Table 2 shows that **MIP** and **IP<sup>rs</sup>** perform significantly better than **IP<sup>r</sup>** and **FCFS**. We conclude that a better integration of ground routing and runway sequencing is valuable. Nevertheless, Table 3 reveals that the computation times of **MIP** and **IP<sup>rs</sup>** are long and do not match the industrial requirements for all instances. **MIP** particularly suffers from long computation times: the time limit is reached very often on instances with 60 departures.

Secondly,  $\mathbf{IP}^{rs}$  provides good solutions and even finds better solutions than  $\mathbf{MIP}$  for some instances with 60 departures. It highlights that most of routing conflicts are successfully captured. Furthermore  $\mathbf{IP}^{rs}$  offers much shorter computation times than  $\mathbf{MIP}$ . Average computation times are short enough for instances with 40 departures, but they are still too long for instances with 60 departures, particularly for flight mix 10-70-20. In Section 7.4, we explore how our reformulations and filtering improve these computation times.

Figure 5 additionally presents the results in terms of average completion time and taxi time. Average completion times are significantly improved by  $\mathbf{MIP}$  and  $\mathbf{IP}^{rs}$ : approximately 30 seconds are earned on instances with 40 departures and 1 minute on instances with 60 departures. Improvements in average taxi time are less important but still significant.

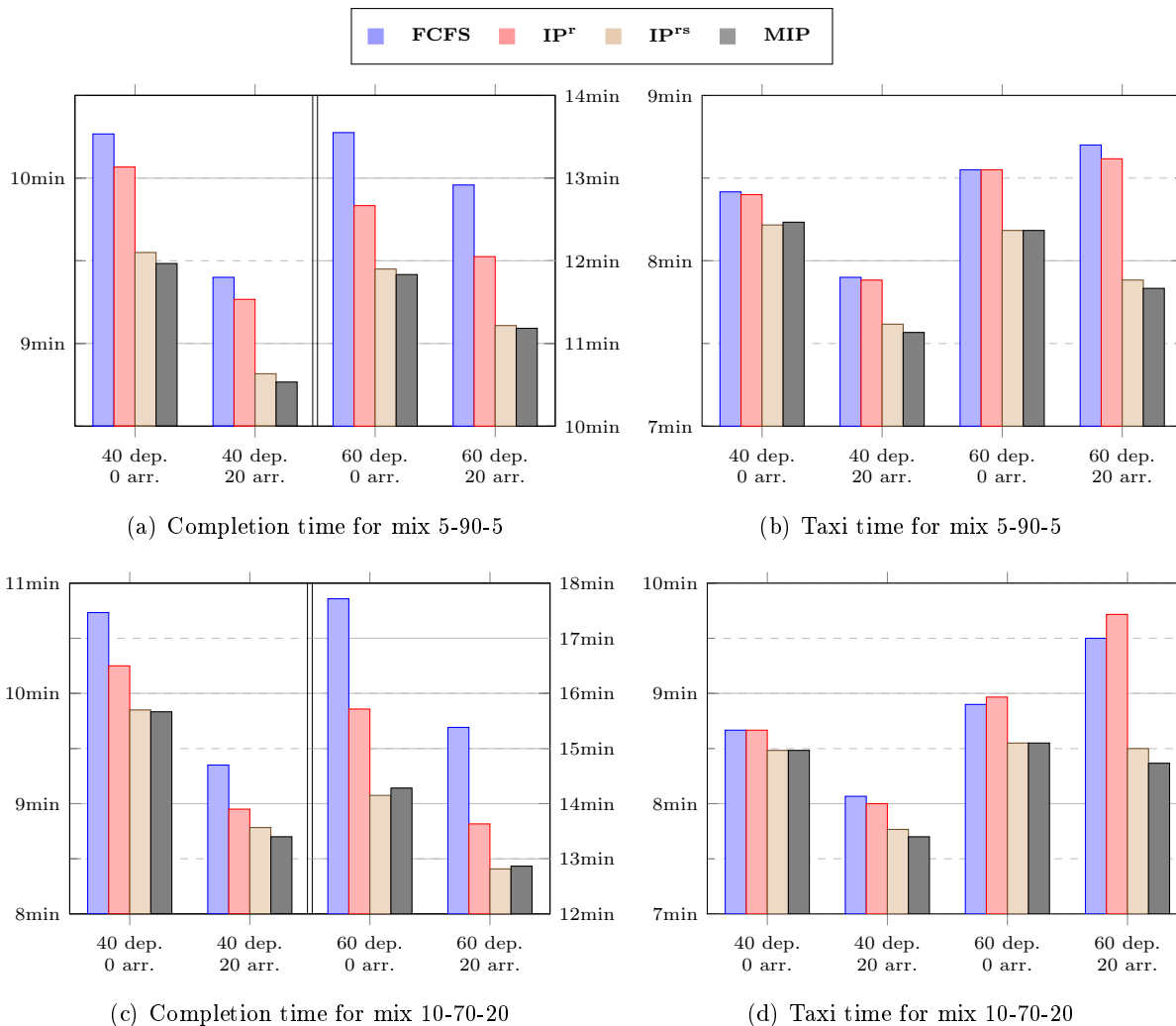


Figure 5: Average taxi time and completion time

In conclusion, significant benefits arise from a better integration of the GRP and the RSP. Nevertheless, the proposed methods are not suitable for a direct application because of excessive computation times. The next section shows how the improvements proposed for  $\mathbf{IP}^{rs}$  (see sections 6.2, 6.3 and 6.4) address this problem.

#### 7.4. Improving computation times of $IP^{rs}$

In this section, we will study the effect of implementing the improvements of ( $IP^{rs}$ ) presented in Section 6. These improvements will be referred to as follows.

- *Taxi time & stand blockage* (Section 6.2): Taxi time and stand blockages constraints (25-27) are reformulated by constraints (31-33).
- *Filter short* (Section 6.3): Stand separation constraints are filtered.
- *Using previous time window solution* (Section 6.4): The solution of the previous time window can be used to heuristically reduce the number of variables and constraints.

We will also study the following combinations of the above variations : *Taxi time & stand blockage + Filter short* and *Taxi time & stand blockage + Filter short + Previous time window*.

*Effect of reformulations and filtering.* Figure 6 presents the effect on computation times of the reformulations and filtering. The figures above the bars are the number of iterations for which the time limit was reached.

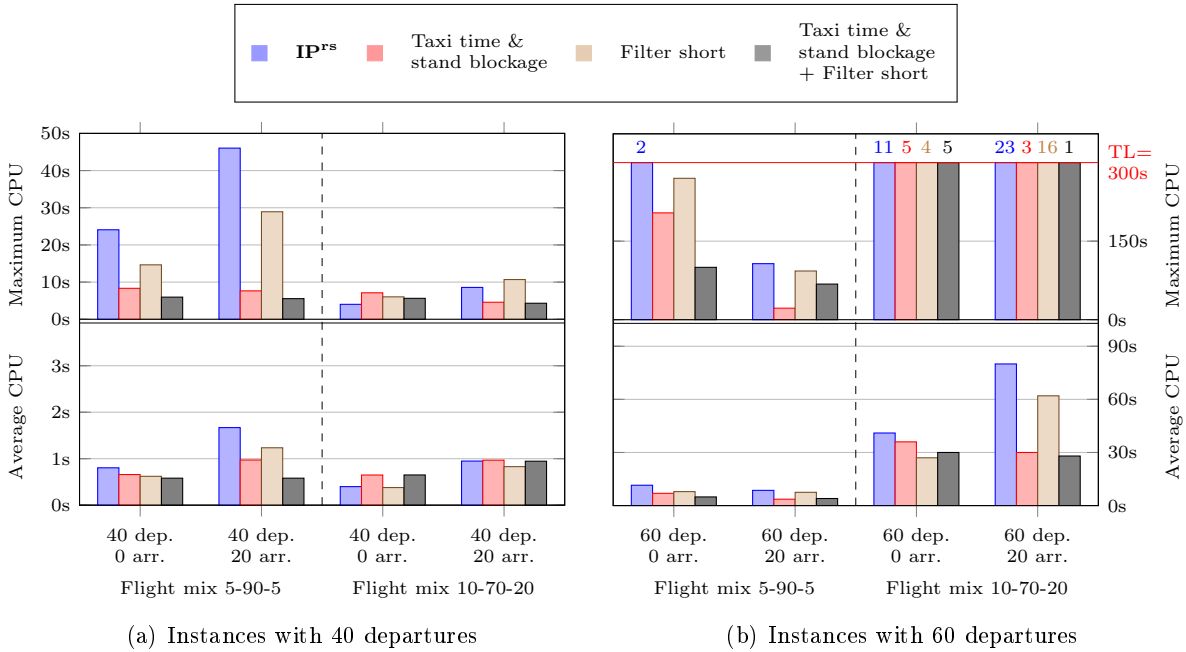


Figure 6: Effect of reformulations and filtering on computation times

For instances with 40 departures, using both techniques is the fastest approach for flight mix 5-90-5, but their effect is more mitigated for flight mix 10-70-20. Nevertheless, it brings the maximum computation times below 10 seconds and average computation times below 1 second.

However, this is not the case of instances with 60 departures and particularly for flight mix 10-70-20. The time limit is reached for some iterations and the average computation times remain excessive, though they are significantly reduced.

Each iteration admits several optimal solutions. Therefore using exact reformulations, such as (31)-(33), does not guarantee to find the same solution. Moreover, our filtering of stand separation constraints is heuristic and finding the same solution is not guaranteed too. Because of the sliding time window

approach, it can potentially lead to significant differences in the final solutions. We observed that solutions are actually different, but within an average gaps of 0.2 % close. Finally, no approach appears to be systematically better than the others.

In conclusion, these experiments highlight that using both our constraints reformulations and our filtering is generally preferable and significantly reduces computation times. Unfortunately, it is not sufficient for an industrial application. In the next paragraph, we try to tackle this problem by using the solution of the previous time window, as explained in Section 6.4.

*Using the previous time window.* As explained in Section 6.4, the solution of the previous time window can be used to heuristically reduce the number of variables and constraints. It is supposed to speed up the solving, but it can also deteriorate the quality of the solution.

Table 4 show how computation times are reduced when implementing this principle. The gaps are computed with respect to the best solutions known so far.

The computation times presented do not account for the first iteration since no previous solution is available, thus they are not representative of the method. The gaps are computed with respect to the best solution known so far. They are not counted for the other method as well.

		Taxi time & stand blockage + Filter short + Using previous time window solution				Taxi time & stand blockage + Filter short			
		Gap		CPU		Gap		CPU	
		Max	Avg	Max	Avg	Max	Avg	Max	Avg
Flight mix 5-90-5	40 dep. 0 arr.	0.86 %	0.44 %	1.1s	0.2s	1.05 %	0.52 %	5.1s	0.5s
	40 dep. 20 arr.	1.66 %	0.74 %	1.7s	0.3s	1.52 %	0.62%	5.6s	0.9s
	60 dep. 0 arr.	1.71 %	0.22 %	17s	0.8s	1.45 %	0.5%	100s	5.1s
	60 dep. 20 arr.	4.12 %	1.38 %	11s	0.9s	2.81 %	0.71 %	68s	4.1s
Flight mix 10-70-20	40 dep. 0 arr.	0.46 %	0.11 %	2.1s	0.2s	0.46 %	0.19 %	4.1s	0.6s
	40 dep. 20 arr.	1.94 %	0.87 %	1.1s	0.3s	2.18 %	0.8 %	4.3s	0.9s
	60 dep. 0 arr.	0.56 %	0.14 %	20s	2.5s	1.11%	0.28 %	300s	32s
	60 dep. 20 arr.	3.14 %	0.88 %	45s	3.0s	2.29 %	0.91 %	300s	29s

Table 4: Using the solution of previous iteration

Table 4 highlights that this process is very efficient. Firstly, almost all test sets are solved with an average gap below 1 %. Note that the gap of one method is not systematically better than the other method. It is again due to the sliding time window scheme. Secondly, the average computation times are appropriate to an industrial application and the time limit is never reached. The maximum computation time over the instances with 60 departures and 20 arrivals are still a bit long for flight mix 10-70-20, but it remains reasonable. Note that a fine tuning of  $\Delta_i$  can allow to better control computation times, but it may further deteriorate the quality of the solutions.

## 8. Conclusion

This paper has focused on the management of the departure process, from push back to take-off, through an integration of the GRP and the RSP. We have shown that an integration of both problems results in a better synchronization of ground movements and take-offs. We have proposed a heuristic sequential algorithm based on an innovative IP formulation of the RSP that takes into account stand area conflicts. In a numerical study based on Copenhagen Airport (CPH) layout, we have shown that

our approach provides high quality solutions, while offering significantly shorter computation times than an exact approach from the literature which integrates both problems in a single MIP. In the end, our approach fits the industrial requirements.

Numerical experiments were performed on a specific layout and one may wonder to what extent our results could be generalized to other airports. Our approach relies on the fact that the stand area and the runway are the main bottlenecks during departure peaks. It is the case for many airports (see e.g. [Atkin et al. \(2012\)](#), [Guépet et al. \(2016\)](#)) but may not be true for some other layouts. Another perspective of research would be to include the optimization of landing times, which are considered as an input in our numerical experiments. We have explained how to adapt the IP formulation of the RSP to deal with arrivals, but computational complexity will certainly be increased. It would be also of interest to model more finely stand separation constraints. The numerical study could also be refined by including a fairness criterion, speed limits that depend on the aircraft type or additional take-off constraints (e.g. SID separation times).

However, the methodology we followed to design our method could be applied to other airports: (1) analyze the bottlenecks of the traffic through an analysis of ground movements ([Guépet et al., 2016](#)), (2) propose a simplified model of integration focusing on these bottlenecks (current paper). It is consequently more likely to be tractable than a global and direct integration.

## AppendixA. Computing the bounds $e_{iu}$ and $l_{iu}$

A flight  $i$  is subject to an earliest time at origin  $T_{o_i}$  and a latest time at destination  $L_{d_i}$ . Note that flight  $i$  cannot reach node  $u_k \in P_i = (o_i, u_2, \dots, u_k, \dots, u_{|P_i|-1}, d_i)$  before

$$T_{o_i} + \sum_{k'=1}^{k-1} T_{iu_{k'}u_{k'+1}}^{min} \quad (\text{A.1})$$

because of the speed restrictions. Also note that flight  $i$  must have left node  $u_k \in P_i$  before

$$L_{d_i} - \sum_{k'=k}^{|P_i|-1} T_{iu_{k'}u_{k'+1}}^{min} \quad (\text{A.2})$$

to meet latest time at destination  $L_{d_i}$ . Also, arrival  $i$  has to free the runway as soon as the landing is over, thus it necessarily leaves node  $u_k \in P_i$  before

$$T_{o_i} + \sum_{k'=1}^{k-1} T_{iu_{k'}u_{k'+1}}^{max} \quad (\text{A.3})$$

Therefore, flight  $i$  is subject to slot restrictions  $[e_{iu}, l_{iu}]$  at every node  $u$  of its path, where

$$e_{iu} = (\text{A.1}) \quad \forall i \in \mathcal{F}, \forall u \in V_i \quad \text{and} \quad l_{iu} = \begin{cases} (\text{A.2}) & \forall i \in \mathcal{F}_{dep}, \forall u \in V_i \\ \min\{(\text{A.2}), (\text{A.3})\} & \forall i \in \mathcal{F}_{arr}, \forall u \in V_i \end{cases}$$

For a departure  $i$ , note that  $e_{io_i} = T_{o_i}$  and  $l_{id_i} = L_{d_i}$ . For an arrival  $i$ , note that  $e_{io_i} = l_{io_i} = T_{o_i}$  and that  $l_{id_i} \leq L_{d_i}$ .

## Appendix B. Identifying the stand area conflicts

Conflicts in the stand area have been described in Section 1 (see Figure 1). Consider two flights  $i$  and  $j$ , we will determine if they are in conflict in the stand area, i.e. compete for the same taxiway segment(s) or node(s). In that case, we will determine a conflicting interval, i.e. two bounds  $a_{ij}$  and  $b_{ij}$  such that if  $i$  pushes back / parks in at time  $t$  then  $j$  cannot push back / park in during the interval  $[t + a_{ij}, t + b_{ij}]$ . The interval must be as large as possible in order to capture the whole conflict. It must be remarked that  $a_{ij}$  and  $b_{ij}$  do not depend on time  $t$  since separation times  $S_{iju}$  neither. Hence, we assume that  $i$  pushes back / parks in at time 0 in what follows.

For each flight  $i$ , we will note  $V_i^r$  and  $E_i^r$  the node and edges sub-set of  $V_i$  and  $E_i$  in the stand area. We thus have  $V_i^r = \{o_i, u_2, \dots, u_{k_i}\}$  if  $i$  is a departure, and  $V_i^r = \{u_{k_i}, \dots, u_{|P_i|-1}, d_i\}$  if  $i$  is an arrival.

Algorithms 1 and 2 compute respectively  $a_{ij}$  and  $b_{ij}$ . They use four methods:

- *buildSchedule*( $i, t$ ) computes the shortest schedule in the stand area such that flight  $i$  pushes back / parks in at time  $t$ . More precisely, it returns the schedule  $(t_u)_{u \in V_i^r}$  defined as follows:

- if  $i$  is a departure,  $t_{i o_i} = t$  and  $t_{i u_k} = t_{i u_{k-1}} + T_{i u_{k-1} u_k}^{min}, \forall k = 2, \dots, k_i$
- if  $i$  is an arrival,  $t_{i d_i} = t$  and  $t_{i u_k} = t_{i u_{k+1}} - T_{i u_k u_{k+1}}^{min}, \forall k = |P_i| - 1, \dots, k_i$  (reverse order)

Note that the duration of the shortest schedule does not depend on  $t$  since minimum travel times  $T_{iuv}^{min}$  neither. This duration is equal to  $\sum_{uv \in E_i^r} T_{iuv}^{min}$  and is hereafter referred to as *duration*( $i$ ).

- *isFeasible*(*schedule* $_i$ , *schedule* $_j$ ) indicates if *schedule* $_i$  and *schedule* $_j$  (outputs of the previous *buildSchedule* method) are compatible in the stand area, i.e. respect minimum separation at each node and overtake and head-on constraints on each edge of the stand area.
- *lb*( $i, j$ ) returns a lower bound of  $a_{ij}$ . For example,  $-(duration(i) + duration(j) + \sum_{u \in V_i^r \cap V_j^r} S_{jiu})$  fits: it is clear that if  $j$  pushes back / parks at this time, it uses the stand area first and does not interfere with  $i$  (pushing back / parking in at time 0). Note that any other lower bound can be used.
- Similarly, *ub*( $i, j$ ) is an upper bound of  $b_{ij}$ , e.g.  $duration(i) + duration(j) + \sum_{u \in V_i^r \cap V_j^r} S_{iju}$ .

The principle of Algorithm 1 is to look for the earliest time of conflict  $a_{ij}$  iteratively from *lb*( $i, j$ ) to *ub*( $i, j$ ). If upper bound *ub*( $i, j$ ) is reached, flights  $i$  and  $j$  are not in conflict in the stand area. In the other case, Algorithm 2 is called for computing the latest time of conflict  $b_{ij}$ . Its principle is similar to Algorithm 1.

Note that depending on the structure of the stand area, there could be compatible times in the interval  $[t + a_{ij}, t + b_{ij}]$ , but they would require an accurate and tight synchronization, which is not robust and explain why we do not consider them. Algorithm 1 and 2 can be adapted to find these times and compute multiple conflicting intervals.

J.A.D. Atkin, E.K. Burke, J.S. Greenwood, and D. Reeson. A metaheuristic approach to aircraft departure scheduling at London Heathrow Airport. In *Electronic proceeding of the 9th international conference on computer-aided scheduling of public transport*, San Diego, California, USA, 2004.

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

**Data:** flights  $i$  and  $j$   
**Result:**  $a_{ij}$   
 $schedule_i = buildSchedule(i, 0);$   
 $a_{ij} = lb(i, j);$   
 $schedule_j = buildSchedule(j, a_{ij});$   
**while**  $isFeasible(schedule_i, schedule_j)$  **and**  $a_{ij} \leq ub(i, j)$  **do**  
     $a_{ij} = a_{ij} + 1;$   
     $schedule_j = buildSchedule(j, a_{ij});$   
**end**

**Algorithm 1:** Computing  $a_{ij}$

**Data:** flights  $i$  and  $j$ ,  $a_{ij}$   
**Result:**  $b_{ij}$   
 $schedule_i = buildSchedule(i, 0);$   
 $b_{ij} = ub(i, j);$   
 $schedule_j = buildSchedule(j, b_{ij});$   
**while**  $isFeasible(schedule_i, schedule_j)$  **and**  $b_{ij} \geq a_{ij}$  **do**  
     $b_{ij} = b_{ij} - 1;$   
     $schedule_j = buildSchedule(j, b_{ij});$   
**end**

**Algorithm 2:** Computing  $b_{ij}$

- J.A.D. Atkin, E.K. Burke, J.S. Greenwood, and D. Reeson. An examination of take-off scheduling constraints at London Heathrow Airport. *Public Transport*, 1:169–187, 2009a.
- J.A.D. Atkin, E.K. Burke, and S. Ravizza. The airport ground movement problem: past and current research and future directions. In *proceedings of the 4th International Conference on Research in Air Transportation, Budapest, Hungary*, 2010.
- J.A.D. Atkin, G. De Maere, E.K. Burke, and J.S. Greenwood. Addressing the pushback time allocation problem at Heathrow airport. *Transportation Science*, 47(4):584–602, 2012.
- Jason AD Atkin, Edmund K Burke, John S Greenwood, and Dale Reeson. An examination of take-off scheduling constraints at london heathrow airport. *Public Transport*, 1(3):169–187, 2009b.
- H. Balakrishnan and B. Chandran. Algorithms for scheduling runway operations under constrained position shifting. *Operations Research*, 58(6):1650–1665, 2010.
- J. E. Beasley, M. Krishnamoorthy, Y. M. Sharaiha, and D. Abramson. Scheduling aircraft landings - the static case. *Transportation Science*, 34(2):180–198, 2000.
- J. A. Bennell, M. Mesgarpour, and C. N. Potts. Airport runway scheduling. *4OR - A Quarterly Journal of Operations Research*, 9:115–138, 2011.
- C. Bosson, M. Xue, and S. Zelinski. Optimizing integrated terminal airspace operations under uncertainty. In *Digital Avionics Systems Conference (DASC), 2014 IEEE/AIAA 33rd*, pages 1A3–1, 2014a.

- C. Bosson, M. Xue, and S. Zelinski. Optimizing integrated arrival, departure and surface operations under uncertainty. 2014b.
- G.L. Clare and A.G. Richards. Optimization of taxiway routing and runway scheduling. *Intelligent Transportation Systems, IEEE Transactions on*, 12(4):1000–1013, 2011.
- R. Deau, J.-B. Gotteland, and N. Durand. Runways sequences and ground traffic optimization. In *proceedings of the 3rd International conference on Research in Air Transportation, Fairfax, USA*, 2008.
- R. Deau, J.-B. Gotteland, and N. Durand. Airport surface management and runways scheduling. In *proceedings of the 8th USA/Europe Air Traffic Management R&D Seminar, Napa, USA*, 2009.
- Eurocontrol. Airport CDM leaflet, January 2009. URL [http://www.euro-cdm.org/library/cdm\\_leaflet.pdf](http://www.euro-cdm.org/library/cdm_leaflet.pdf).
- T. Fahle, R. Feldmann, S. Gotz, and B. Monien. The aircraft sequencing problem. *Computer Science in perspective*, pages 152–166, 2003.
- J. Guépet, O. Briant, J.P. Gayon, and R. Acuna-Agost. The aircraft ground routing problem: Analysis of industry punctuality indicators in a sustainable perspective. *European Journal of Operational Research*, 248(3):827–839, 2016.
- Julien Guepet. *Optimization of airport operations : stand allocation, ground routing and runway sequencing*. PhD thesis, Université Grenoble Alpes, December 2015. URL <https://tel.archives-ouvertes.fr/tel-01257637>.
- Y. Jung, T. Hoang, J. Montoy, G. Gupta, W. Malik, and L. Tobias. A concept and implementation of optimized operations of airport surface traffic. In *10th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference, Fort Worth, TX*, 2010.
- Y. Jung, T. Hoang, J. Montoya, G. Gupta, W. Malik, L. Tobias, and H. Wang. Performance evaluation of a surface traffic management tool for Dallas/Fort Worth International Airport. In *Ninth USA/Europe Air Traffic Management Research and Development Seminar*, pages 1–10, 2011.
- G. Keith and A. Richards. Optimization of taxiway routing and runway scheduling. In *proceedings of AIAA Guidance, Navigation and Control Conference, Honolulu, Hawaii, USA*, 2008.
- B. Kim, L. Li, and J.P. Clarke. Runway assignment by minimizing emissions in terminal airspace. In *Proceedings of AIAA Guidance, Navigation and Control Conference, Toronto, Canada*, 2010.
- H. Lee and H. Balakrishnan. A comparison of two optimization approaches for airport taxiway and runway scheduling. In *Digital Avionics Systems Conference (DASC), 2012 IEEE/AIAA 31st*, pages 4E1–1, 2012.
- A. Lieder, D. Briskorn, and R. Stolletz. A dynamic programming formulation for the aircraft landing problem with aircraft classes. *European Journal of Operational Research*, 243:61–69, 2015.
- W. Malik, G. Gupta, and Y. Jung. Managing departure aircraft release for efficient airport surface operations. In *AIAA Guidance, Navigation, and Control Conference*, 2010.
- S. Rathinam, Z. Wood, B. Sridhar, and Y.C. Jung. A generalized dynamic programming approach for a departure scheduling problem. In *AIAA Guidance, Navigation, and Control Conference*, pages 10–13, 2009.