Robust Resource Planning for Aircraft Ground Operations

Yagmur S. Gök[1], Daniel Guimarans[2], Peter J. Stuckey[2],
Maurizio Tomasella[1], and Cemalettin Ozturk[3]

[1] Business School, The University of Edinburgh, 29 Buccleuch Place, EH8 9JS,
Edinburgh, United Kingdom
{Yagmur.Gok, Maurizio.Tomasella}@ed.ac.uk
[2] Faculty of Information Technology, Monash University, 900 Dandenong Road,
Melbourne, VIC 3145, Australia
{daniel.guimarans, peter.stuckey}@monash.edu
[3] United Technologies Research Center, 4th Floor, Penrose Business Centre, Penrose
Wharf, Cork, T23 XN53, Ireland
OzturkC@utrc.utc.com

Abstract. Aircraft turnaround scheduling and airport ground services team/equipment planning directly concern both the airport operator and service providers. We first ensure airport-wide global optimality by solving a resource-constrained project scheduling problem (RCPSP) for minimal overall delays. We then support decentralized allocation of teams/vehicles to flights, independently by each service provider. Either a multiple traveling salesman problem with time-windows (mTSPTW), or a vehicle routing problem with time-windows (VRPTW) are solved for this purpose, by taking advantage of both constraint programming (CP) and mixed integer programming (MIP) solvers. We also exploit these models in a matheuristic approach based on large neighborhood search used to reach good solutions in reasonable time for real-world instances. Unlike the classical VRP objective of minimizing traveling time, we maximize the total slack time between team visits, and show that doing this fosters robustness of the generated plans. We assess the robustness of solutions through a discrete-event simulation model, and conclude by validating our approach with data provided by a major ground handling company for a day of operations at Barcelona El Prat Airport.

Keywords: Optimization · Scheduling · Routing · Aviation · Airport Operations

1 Introduction

Effective planning and scheduling is crucial in many areas of airport operations, where decisions are interconnected with each other and the potential for flight delays due to knock-on effects is rather high. Careful planning for the day of operations is essential. With many factors out of the control of any airport decision maker —weather conditions, aircraft technical faults, delays, late passengers
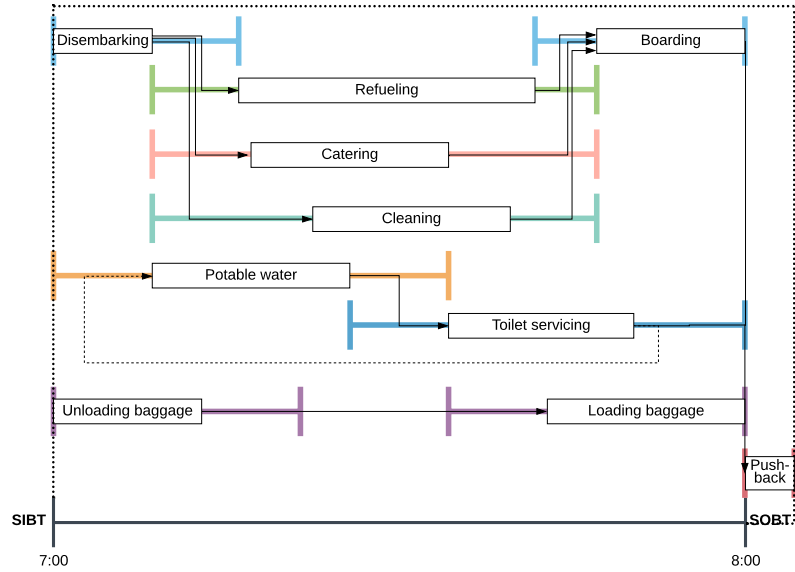
Fig. 1. An example of aircraft turnaround operations and precedence relationships between them within a timeline

on boarding, etc.—, plans that are in some form *robust* should be sought. This is particularly true for aircraft turnaround and airport ground services. When an aircraft lands, it proceeds to a parking stand. Here, it undergoes a sequence of operations to get ready for the following take-off. A mix of operations may be needed: passenger disembarking/boarding, baggage unloading/loading, refueling, cabin cleaning, catering, toilet and potable water servicing, and aircraft push-back. Precedence relations do apply —e.g. refueling often cannot start before passengers have disembarked due to safety regulations. Figure 1 shows an example of a single aircraft turnaround. Conjunctive arcs represent the precedence relations. There exist specific time windows through which each activity should be performed. Each time window shortens/stretches out depending on what happens to other related operations. The turnaround should ideally start as soon as an aircraft arrives at its stand, ideally at the time for which it was scheduled to arrive there —or start of *in-block time* (*SIBT*)— and should be completed by the time it was scheduled to be pushed-back into the taxiway on its way to the runway —also called its start of *off-block time* (*SOBT*).

Turnaround operations are often handled by different service providers (SPs). A great deal of coordination is required to make sure they do not delay any aircraft, and for delays not to propagate to other aircraft. Cross-turnaround delay propagation may happen if teams from SPs are scheduled back-to-back between subsequent aircraft: team A finishing late on aircraft turnaround X will surely start late on turnaround Y if it was immediately assigned to it. Team A will also be occupying the stand for longer, with turnaround Z of the aircraft

next assigned to the same stand likely to start late, and operations of team B (possibly of a different SP), that was assigned to Z also impacted. Propagation of delays across the airport parking lot —*apron* for short— is a certainty.

Organizations in aviation have long been aware of such issues and have devised an approach named Airport Collaborative Decision Making (A-CDM) ([5]), which is currently in place in just a minority of airports in Europe —albeit some of the biggest ones in terms of passenger numbers per year. According to A-CDM, most of the actors involved in aircraft operations must share certain pieces of information with one another to keep tighter levels of coordination. A central piece of information is the *Target Off-Blocks Time* (TOBT), which is calculated on the day of operations and used as a reference for all other ground service operations. A-CDM focuses on the coordination of aircraft movements, seen, probably correctly, as the center-piece of airport operations. But coordination of the movements of ground service staff and related equipment across busy aprons is equally critical, as it contributes to delays, but also because certain pieces of costly equipment are shared by multiple turnaround teams working for separate SPs. While A-CDM is an appealing concept, to work as expected, a certain degree of information sharing among the airport and the various SPs need to be in place, which happens typically only if enforced. Busier hub airports represent the typical example where the enabling conditions are met. In non-A-CDM airports, though, coordination is also needed, but without data sharing mechanisms it is virtually impossible to achieve. The remainder of this paper proposes a feasible and robust way to support this more general scenario.

2   Related Literature

The Operations Research (OR) community has, in the past three decades, worked on the modeling and solution of problems related to coordinating the movements, usage, and sharing of turnaround teams and equipment, but only partially.

The body of work by Norin and colleagues [19, 18] shares quite a few elements with our study. As we, and others do (e.g., [1] and [21]), the problem is modeled as a form of Vehicle Routing Problem with Time Windows (VRPTW) [22]. Similarly to us, they work in an A-CDM setting, albeit with a different objective function: they minimize the weighted sum of flight delays and traveling distance. Due to the computational complexity of the underlying problem, they also focus on finding reasonable solutions in short times adopting a form of Greedy Randomized Adaptive Search Procedure (GRASP). They also make use of simulation to test the robustness of their heuristic plans under a range of uncertain conditions. Differently from us, the authors focus on one turnaround operation only: de-icing, with one type of vehicle/related staff to operate them. Stockholm Arlanda provides for their data set and motivating example.

In another closely related study [12], the authors did not see an option to adopt a fully centralized solution process, as the ground service providers are effectively separate legal entities making independent decisions. They also did not see the point of adopting a complex negotiation mechanism between dif-

ferent service providers, essentially ruling out in their assumptions any form of information sharing. This is a very relevant paper based on the same problem as ours, but examined through a substantially different perspective.

Padron *et al* [21] probably represents the closest work to our study. They also consider scheduling all the ground handling vehicles at an airport, and they combine CP with heuristic search techniques such as Large Neighborhood Search (LNS) and Variable Neighborhood Descent (VND). Unlike us, their turnarounds have a fixed sequence of tasks. In a subsequent paper [20], as we do here, they integrate simulation at the bottom-end of their methodology to investigate the robustness of the generated heuristic solutions.

Among the remaining studies, less related to ours, [7] represents a more recent example of VRPTW formulation, but focuses on individual workers and their synchronization with vehicles, and has no view on collaborative mechanisms. Works such as [1], [10], and [23] provide different heuristic approaches to solve the vehicle routing problems implied by aircraft turnaround operations at airports. Finally, many older studies had approached apron resource allocation problems from a multi-agent distributed planning perspective. Among these, [16], [11],[14] and [13] modeled their problems —as we also do in one of our steps — after the Resource-Constrained Project Scheduling Problem (RCPSP) [2], while [6] modeled the problem as a form of job shop scheduling.

3   Planning Process and Related Models

There are two levels to the turnaround planning problem studied in this paper— *long-term* and *short-term* (Figure 2). All decisions are of tactical nature, as they take place ahead of the day of operation. Real-time management of apron operations and related resources is out of scope for the present paper.

In the longer term, flight schedules for the next few months are known to both the airport operator (AO) and the SPs. The AO needs this information to coordinate the scheduling of all the turnaround operations for each of the days in the planning horizon, aiming at minimum delays. The main decision maker in the longer-term is the AO, wanting to fix time windows for all operations whilst keeping overall resource requirements for the airport reasonably contained. After these decisions are made centrally, each SP can start thinking about their own resource requirements for each day of operation, and run their own staff rostering processes.

In the shorter term, approximately a week before the day of operations, the updated flight schedule is shared with all SPs and the same tactical reasoning can be rerun, with time windows revisited by the AO and rosters updated by the SPs.

In the even shorter term, closer to the start of the day of operations, time windows remain fixed and turnaround teams are known to SPs with a high degree of certainty. Then, each SP makes sure they can optimally route their staff through all turnaround tasks, by keeping some slack time to compensate
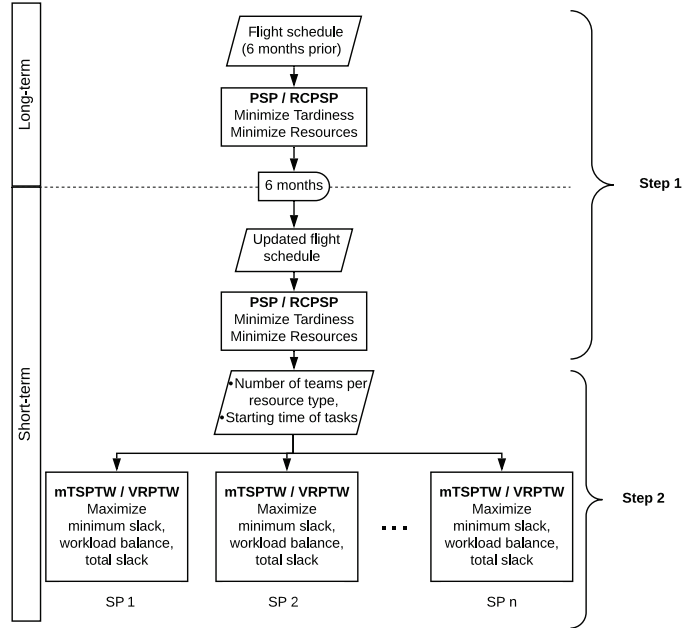
Fig. 2. Proposed turnaround planning process

for unforeseeable delays. At this stage, the objectives and constraints can differ by SP, but all SPs still have to stick to the time windows set by the AO.

Although our approach would ultimately help to plan for all kinds of resources in apron operations, we focus on planning for 'teams' of handling agents of each SP, i.e. we focus on human resources. Each of the turnaround operations is normally executed by small teams of employees, of size known to each SP, and planning for the sequence of turnarounds to be visited and serviced by each team during the day of operation is of utmost importance. In the following, we assume that all needed equipment is either carried over by the teams as they move from one turnaround to the next, or sourced across the apron area as they move through their jobs for the day, and as such is not modeled directly.

From an OR perspective, two classes of problems are involved in the description given above: *project scheduling* provides a convenient framework to the AO's problem of fixing time windows and keeping resource requirements under control, while *vehicle routing* comes to the rescue of SPs for optimal routing of their turnaround teams.

We consider (Subsection 3.1) both versions of project scheduling problems (PSPs), with and without constraints in the number of resources (e.g., teams for unloading/loading baggage, cleaning, catering service, refueling, etc.), the latter class of problems taking the name of resource-constrained project scheduling problems (RCPSP). The first objective we seek is to minimize the overall tar-

diness for the airport. After achieving a minimal-delay *ideal* schedule, the total resource requirement for the airport is minimized by enforcing any optimal solution found at this stage to maintain at least the same level of tardiness coming from the PSP.

Moving to optimal routing of teams for each SP (Subsection 3.2), the frameworks of reference are two: multi-traveling salesmen problem with time windows (mTSPTW) [4] and vehicle routing problem with time windows (VRPTW). The choice depends on whether the vehicles that are operated by teams are *capacitated* or not. Most resources/teams use a vehicle with limited capacity in order to service an aircraft, such as catering trucks, where the catering team is responsible for loading a certain number of galley trolleys into the aircraft. Other resources are uncapacitated and have no limitations on the number of aircraft that can be serviced before returning to the depot (e.g., push-back vehicles). Unlike the typical objective of both mTSPTW and VRPTW, which is minimizing the traveling distance, we first maximize the slack between tasks, in order to give enough time to absorb any small disruptions. This is meant to enforce a certain degree of robustness to the plan. After that, we try and maximize the workload balance among teams of the same SP, as much as possible, to foster fairness of the plan. Finally, we maximize the total slack time, when the minimum slack time cannot be improved further, with the effect of increasing all slack times between tasks except the minimum among all.

Details of our models are provided in the next two sub-sections. Many powerful global constraints from the Constraint Programming (CP) community are available for PSP/RCPSP and mTSPTW/VRPTW, hence our CP formulations. By employing such global constraints and taking advantage of the strength of different solvers, one would expect better computational performance with respect to, say, Mixed-Integer Programming (MIP) formulations and solvers. All models were developed in MiniZinc [17], a solver-independent modeling framework that allows the model to be run on many different solvers. This feature was crucial to enable our solution approach (Section 4).

3.1   Project Scheduling Models

For a given day of operations, we set the start time $start_i \in [0, t_{max}]$ of all tasks $i \in I = \{1 \ldots \phi\}$ that cover all aircraft turnarounds expected at the given airport (t_{max} is the length of the day of operation). We do so in two steps. In the first (PSP, equations (1a) and (2)—(6)), we aim at minimum costs resulting from tardy turnarounds, assuming unlimited resources. In the second (RCPSP, equations (1b), (2)—(6), (7b) and (8b)), we aim for minimal resource needs whilst maintaining tardiness performance established in the first step.

Each task has an expected processing time $duration_i$. Based on the known flight timetables, both the Scheduled Time of Arrival (STA) and Scheduled Time of Departure (STD) of each aircraft are known. As a result of this and of the precedence relations among all tasks (Figure 1), earliest start times sta_i and earliest end times std_i of all tasks are also known in advance. The set of all tasks $j \in I$ which can only start after a given task $i \in I$ is completed is denoted

as S_i. This set will be empty for push-back tasks, which represent the natural conclusion of the related turnarounds. In between any two tasks, a fixed setup time $setup_i$ will ensure resources can effectively be gathered and moved from one location to another across the apron. This parameter can be estimated as a function of the maximum distance among all stands.

Each task represents a specific activity $A = \{1 \dots \alpha\}$, e.g. baggage loading. By AT_a we denote the set of all tasks of type $a \in A$. Sets SO and SI represent, respectively, activities that are only allowed to start a certain time before STD, or after STA (mostly due to process specifications).

Certain pairs of tasks cannot be performed simultaneously, e.g. potable water and toilet servicing of the same turnaround. $P = \{1 \dots N\}$ is the set of such forbidden pairings, and D_p is the set of (two) tasks for each $p \in P$.

Each task requires, uninterruptedly from start to end, a given amount rr_{i_k} of a given type of resource $k \in K = \{1 \dots \kappa\}$. Resource types effectively represent teams of handling agents providing services of different nature. Resource capacity per resource type rc_k also needs to be decided.

In the joint CP formulation of the PSP/RCPSP steps that follows, we also denote (Objective Z_1, see (1a)) the cost of tardy turnarounds per unit of time as $costtardy$, while parameter $sobt_a$—see constraint (4)— states that certain activities need to be completed within given bounds from the planned departure time. Finally, we employ two global constraints: global constraint (6) ensures non permitted task pairs are scheduled separately, while global constraint (8b) ensures resource levels are not exceeded at any time.

CP Formulation

$$Z_1^* = \min \ \ Z_1 = \min \sum_{\substack{i \in I \\ \text{where} \\ S_i = \{\}}} costtardy \times max\{0, start_i + duration_i - std_i\} \tag{1a}$$

$$Z_2^* = \min \ \ Z_2 = \min \sum_{k \in K} rc_k \tag{1b}$$

subject to

$$start_i \geq sta_i \ \ \forall i \in I \tag{2}$$

$$start_j \geq start_i + duration_i \ \ \forall i \in I, \forall j \in S_i \tag{3}$$

$$start_i + duration_i \geq std_i - sobt_a \ \ \forall a \in SO, \ \ \forall i \in AT_a \tag{4}$$

$$start_i = sta_i \ \ \forall a \in SI, \ \ \forall i \in AT_a \tag{5}$$

$$disjunctive([start_i | i \in D_p], [duration_i | i \in D_p]) \ \ \forall p \in P \tag{6}$$

$$Z_1 = Z_1^* \tag{7b}$$

$$cumulative([start_i \mid i \in I], [duration_i + setup_i \mid i \in I], [rr_{k,i} \mid i \in I], rc_k)$$
$$\forall k \in K \ \text{ where } rc_k < \sum_{i \in I} rr_{k,i} \tag{8b}$$

3.2   Routing Models

After solving the above project scheduling problems, both the AO and all SPs know all time windows in which tasks should be executed on the given day of operation, as well as the number of teams of all types that are likely required to do the job. Each SP then takes this information to optimally schedule for their own teams to cover all turnaround tasks they are contracted to service. In the following we will see how this single-SP decision can be supported.

As in subsection 3.1, we provide a joint formulation following a lexicographic approach, with three objectives/sub-problems in this case. The sequence of three is then repeatedly solved as many times as the resource types managed by the given SP. For some resource types (teams), the vehicles used, e.g. re-fueling trucks, have finite capacity, hence they may need to be replenished before visiting the next turnaround. For these resource types, capacity constraints (20)—(22) will need to be included in the three models. The sub-problems take then the form of a VRPTW, irrespective of the objective/step in the sequence. For other resource types (teams), e.g. push-back trucks, capacity is not an issue, constraints (20)—(22) are excluded, and the sub-problems take the form of an mTSPTW.

The objective of utmost importance, and the one to pursue first (equations (9) and (12)—(23)), is to maximize the minimum slack time between any two tasks, in an attempt to absorb short delays and prevent minor knock-on effects. The second step in the sequence (equations (10), (12)—(23) and (24b)) looks at maximizing the workload balance among teams, to enforce some form of fairness in the plan, something which would be required in highly-unionized settings. Workload equity and its calculation is a subject of interest in the literature [15] on routing problems. The general suggestion points at minimizing the maximum distance in order to achieve a balanced workload while still ensuring the minimization of traveling distances. However, we are not as concerned with traveling times in between tasks as we are with the much higher processing times for each task. The last step (equations (11), (12)—(23), (24b) and (24c)) then seeks to maximize the total slack time in the plan, in a way to increase its robustness.

On the given day of operation, we focus on a given working shift $S = [startshift, endshift]$ for which a staff roster of the given SP is available. Within that, we know the number of teams $t \in T_{SP,k} = \{1 \ldots t_{SP,k}\}$ of resource type k, who need to cover, overall, a known number of tasks $i \in I_{SP,k} = \{1 \ldots \phi_{SP,k}\} \subset I$, by moving across a given number of parking stands $h \in H = \{1 \ldots \eta\}$, where tasks are performed. The SP wants to set, for each task i:

- the start time of the task, or $stime_i \in S$;
- the team $rt_i \in T_{SP,k}$ assigned to i;
- the task s_i immediately following i;
- whether replenishment is needed prior to moving to s_i.

For each available team, a specific route needs to be set up for the given shift (hence the letters 'rt' in rt_i), where the first task is a dummy task the label of which is a function of the label/index of the team in question, while all other tasks are 'genuine' tasks from set $I_{SP,k}$. Task labels then, whether genuine or

dummy, take on value in set $N = I_{SP,k} \cup \{\phi_{SP,k} + 1, \ldots, \phi_{SP,k} + t_{SP,k}\}$. As a result, any task that is not the first task in each route is a task $s_i \in I_{SP,k}$. Constraint (14) ensures all the dummy nodes represent the start of individual routes. Constraint (15) makes sure both i and s_i belong to the same route.

CP Formulation

$$z_1^* = \max \quad z_1 = \max \quad \min_{i \in I_{SP,k}} \, slack_i \tag{9}$$

$$z_2^* = \max \quad z_2 = \max \quad (\min_{t \in T_{SP,k}} \, workload_t - \max_{t \in T_{SP,k}} \, workload_t) \tag{10}$$

$$z_3^* = \max \quad z_3 = \max \quad \sum_{i \in I_{SP,k}} slack_i \tag{11}$$

subject to

$$circuit \; ([s_i \mid \;\; i \in N]) \tag{12}$$

$$alldifferent \; ([s_i \mid \;\; i \in N]) \tag{13}$$

$$rt_{\phi+t} = t \;\; \forall t \in T_{SP,k} \tag{14}$$

$$rt_{s_i} = rt_i \;\; \forall i \in I_{SP,k} \tag{15}$$

$$stime_i = start_i \;\; \forall i \in I_{SP,k} \tag{16}$$

$$busy_i = \begin{cases} stime_i + duration_i + \\ traveltime_{h_i,h_j} + x_i \times replenish, & \forall i \in I_{SP,k} \mid s_i \in I_{SP,k} \\ endshift, & otherwise \end{cases} \tag{17}$$

$$stime_{s_i} \geq busy_i \;\; \forall i \in I_{SP,k} \tag{18}$$

$$slack_i = stime_{s_i} - busy_i \;\; \forall i \in I_{SP,k} \tag{19}$$

$$q_i = cap \;\; \forall i \in N \setminus I_{SP,k} \tag{20}$$

$$q_{s_i} = q_i - demand_{s_i} \;\; \forall i \in N \setminus I_{SP,k} \tag{21}$$

$$q_{s_i} = \begin{cases} cap - demand_{s_i}, & if \;\; x_i = 1 \\ q_i - demand_{s_i}, & otherwise \end{cases} \;\; \forall i \in I_{SP,k} \tag{22}$$

$$workload_t = \sum_{\substack{i \in I_{SP,k} \\ where \; rt_i = t}} duration_i \;\; \forall t \in T_{SP,k} \tag{23}$$

$$z_1 = z_1^* \tag{24b}$$

$$z_2 \geq z_2^* \tag{24c}$$

Tasks should start at a time $stime_i$ that is no earlier than the $start_i$ that was assigned at the PSP/RCPSP stage. Later starts could be advisable/needed,

e.g. because of resource limitations. Hence, although potentially contributing to causes of delays, at this routing stage decisions on task start times could be reconsidered, in principle at least. In our approach we simplify this aspect by fixing $start_i$ exactly as in constraint (16), as this is more akin to maximizing the form of slack in the system that we have in two out of three objectives.

Tasks should also not start until their immediate predecessor has been completed, as in constraints (18) and (17). Any time available in between tasks i and s_i is defined as slack—see constraint (19). The replenishment decision is enacted through binary decision variable x_i, which takes on value 1 if replenishment needs to happen between task i and task s_i, 0 otherwise. Each replenishment takes *replenish* time. Moving between two consecutive tasks requires $traveltime_{h_i,h_j}$, with $h_i, h_j \in H$, $h_i \neq h_j$. Task duration is again denoted as $duration_i$. The sum of the duration of all tasks assigned to a given team contribute to defining the total workload $workload_t$ for the team, as in constraint (23).

For capacity constrained turnaround services, initial capacity q_i for the team and related vehicle is set to *cap*—constraint (20); capacity then depletes as required by the demand from each subsequent task s_i but is topped-up any time a replenishment decision is made—see constraints (21) and (22).

Global constraint (12) builds a single overall sequence for all tasks of all routes, with dummy tasks signposting the start of each team's own route. Global constraint (13) is redundant and added to help propagation.

4   Solution Approach

From the above discussion, we know that our overall approach develops as in Figure 2, with perspectives from both the AO and all the SPs supported by the models presented, respectively, in Sub-sections 3.1 and 3.2.

Table 1. Solution Steps

Step	Model	Obj.	Additional Input	Solver	Search Strategy*	Time Limit(s)	Warm Start
1.1	PSP	Z_1	—	Chuffed	smallest, indomain_ min	—	—
1.2	RCPSP	Z_2	Z_1^*	Chuffed	smallest, indomain_min	—	—
2.1.1	mTSPTW/ VRPTW	z_1	$t_{SP,k}, start$	Gecode	first_fail/smallest, indomain_max/min	10	–
2.1.2	mTSPTW/ VRPTW	z_1	$t_{SP,k}, start,$ s, rt	Gurobi	—	—	Yes
2.2	mTSPTW/ VRPTW	z_2	$t_{SP,k}, start,$ z_1^*	Gecode	first_fail/smallest, indomain_max/min	20	—
2.3	mTSPTW/ VRPTW	z_3	$t_{SP,k}, start,$ z_1^*, z_2^*	Gecode	first_fail/smallest, indomain_max/min	90	—

Algorithm 1: Large Neighborhood Search

Input: rt, s, z_3 from CP model (step 2.3)
$rt^b \leftarrow rt$
$s^b \leftarrow s$
$z_3^b \leftarrow z_3$
while *iter* < *maxIter* **do**
    k1 $\leftarrow$ a random number from $\{1 \ldots t_{SP,k}\}$
    k2 $\leftarrow$ a random number from $\{1 \ldots t_{SP,k}\} \setminus \{k1\}$
    for $i \leftarrow 1$ **to** ϕ **do**
        while $rt_i \in [k1,k2]$ **do**
        | Destroy rt_i and s_i
        end
    end
    Repair rt, s, z_3 with CP model(step 2.3)
    $rt^t \leftarrow rt$
    $s^t \leftarrow s$
    $z_3^t \leftarrow z_3$
    if $z_3^t < z_3^b$ **then**
        $z_3 \leftarrow z_3^t$
        $rt \leftarrow rt^t$
        $s \leftarrow s^t$
    else
        $z_3^b \leftarrow z_3^t$
        $rt^b \leftarrow rt^t$
        $s^b \leftarrow s^t$
    end
end

Table 1 shows further details around solvers and search strategies we used, as well as additional parameters around any time limits adopted as stopping criterion, or whether we made use of a warm-start. Numbers in the first column to the left correspond to component steps of Steps 1 and 2 from Figure 2. Variable names in the table refer to the formulations from Section 3.

All models were implemented in MiniZinc, which enabled us to test the performance of different solvers for each model, and ultimately select the most suitable for use in each case. In the case of steps 1.1 and 1.2, Chuffed [3] clearly outperformed all other available solvers and we were able to prove optimality for all instances. The first sub-problem of Step 2 though, that is the maximization of minimum slack time, proved slightly different, and had to be broken down into two further component steps. We first used CP with a specialized search strategy (step 2.1.1) which allowed us to reach the maximum as quickly as possible with Gecode [8]. However, it took very long for CP to prove optimality in almost all cases. Thus, by taking advantage of the warm-start possibility, we used the same model and the solution provided by Gecode as a warm start for a MIP solver (Gurobi [9]) (2.1.2), thanks to which we managed to prove optimality for all instances in a very short time.

Choosing the right search strategy also proved decisive in terms of solving times wherever we adopted a CP approach. In steps 1.1 and 1.2, we chose the $start_i$ variables to lead the search. The variable selection strategy *smallest* means a variable is chosen with the smallest value in its domain, and the assignment of the value to that variable is done using *indomain_min*, meaning that it will get assigned the minimum value in its domain. On the other hand, for the mTSPTW/VRPTW component, we noticed the model was unable to solve quickly without specifying any search strategies. We then noticed that the two forms of the problem claim for different choices of search strategy. We observed that *first_fail*, where the variable with the smallest domain is chosen, outperforms other strategies for all mTSPTWs, while *smallest* performed better for all VRPTWs. On the variable assignment for mTSPTW, *indomain_max* ruled out the rest, meaning the assignment was made with the maximum value in its domain. For VRPTW, on the other hand, *indomain_min* performed better.

The very last step of our approach involves adopting a Large Neighborhood Search (LNS) schema (Algorithm 1) to further improve the solutions obtained from each of the routing sub-problems composing step 2.2. In our implementation of LNS, we take the solution from maximizing z_3 as a starting point, then 'destroy' two routes, chosen at random from the given solution, and finally use again the same model from step 2.3 to 'repair' it. If the new solution is better than the incumbent, we update the record of the best solution, and repeat the process for up to 200 iterations.

5   Experiments

In this study we used real data coming from Europe's 6th busiest commercial airport, Barcelona - El Prat (BCN). Our data relate to one given day of operation and include seven resource types, with each type handled by a different SP, and ten different turnaround activity types, for a total of 914 tasks to be scheduled at the PSP/RCPSP stage. At the mTSPTW/VRPTW stage, we considered the two shifts per day as currently adopted at the given airport, irrespective of resource type/SP. There are approximately 50 turnarounds in each shift, amounting to approximately the same number of tasks to be assigned per resource type and shift. In some cases from out data set, two teams are required to perform a task —e.g., baggage loading/unloading for wide-body aircraft. In these cases, the tasks are duplicated to ensure two teams, not one, perform the same task.

We ran all models on a personal laptop (1.6 GHz Intel Core i5) running mac-OS High Sierra. The overall integration was achieved in Python 3.7 using MiniZinc Python (MiniZinc version 2.3.2), which allowed us to solve the models in an incremental way, whilst providing a platform for easy integration of our LNS implementation. Computation times are not of primary concern when dealing with our problem, as clear from Figure 2, hence the average 45 minutes taken to run the whole solution approach end to end does not represent a problem, to start with. In reality though, the AO will only run the PSP/RCPSP stage, which will take only approximately 7 minutes. Each SP will instead run the

mTSPTW/VRPTW separately, on its own, for two shifts, which will take up to around 6 minutes for each SP, LNS step included. The RCPSP is proven to be optimal for minimizing tardiness, as well as for minimizing total resources/teams per SP. Moreover, in the second stage, we prove optimality for the maximum minimum slack time for the given tardiness and resource levels. Work balance is between 0 and 66 minutes, and 15 minutes on average, within the allowed time limit. Ensuring optimal workload balance proved too challenging, hence we limited the time available for this stage.

The most time-consuming part of the whole approach was to find the optimal solution for total slack time for each VRPTW. This is due to the workload balance objective and constraints. When we relaxed this and maximized it for z_1 and z_3, we could get an optimal solution using Gurobi for almost all resource types and shifts, with only a few instances not proven in 5 hours of solving time. We compared the results of the objective bounds with our lexicographic approach (excluding workload balance objective and constraint). The average gap was 0.68% for 14 instances of a mix of mTSPTW and VRPTW (each for one shift). The maximum gap was 2.72%. The average solution time for our approach was 1.78 minutes, while it required 60 minutes for Gurobi to prove optimality, if reached.

Padron and Guimarans [20] tackled an extremely similar ground-handling problem on the exact same data set. Compared to them, our approach was able to reduce the number of resources used per resource type when tested in the same instance for BCN. This was largely because in the RCPSP stage we give flexibility to the ordering of tasks for an individual turnaround, rather than using a preset plan for each turnaround as they do.

To test our optimization results under the uncertainty that normally permeates real airport settings, we developed, validated and used a discrete event simulator. Uncertain factors in our problem include: aircraft arrival time, task duration, traveling time between stands, and replenishment time. A bounded exponential probability distribution was used for the traveling time, and triangular distributions for the rest.

In addition to the real case from BCN, we generated several instances with different mix of aircraft and frequencies of arrivals and departures. Instances are presented as ta[θ]_t[$tmax$], where θ is the number of turnarounds and $tmax$ is the planning horizon in minutes. We study these scenarios, together with the one from Barcelona Airport, by setting different levels of variability (normal and high variability). We use this experimental setting to evaluate our approach using two different objectives: maximizing slack time and minimizing traveling time, as in typical VRPTWs. Ten independent replications were produced for each scenario. Table 2 provides a summary of relevant indicators. Σ is the total delay time, N is the number of delays, $N_\%$ is the percentage of delays and $\Sigma_{>15}$ shows the sum of delays that are in excess of 15 minutes. Superscripts s and t refer to the two approaches: maximizing slack and minimizing traveling distance, respectively. Computation time, including solving the RCPSP plus the average solving time (in seconds) of all VRPs and excluding simulation for a deterministic bound, are

indicated by τ^s and τ^t for the respective approaches. Times for the simulated instances with _normal and _high variability are not provided in the table since the deterministic instances' computation times (_det) are the main indicators of the solution approach and simulation is not part of the solution but only used as a tool for evaluation.

Table 2. Simulation Key Performance Indicators (KPIs) for slack time maximization and travel time minimization approaches

instance	$N^s_\%$	$N^t_\%$	Σ^s	Σ^t	$\Sigma^s_{>15}$	$\Sigma^t_{>15}$	$\Delta\Sigma_{>15}$	τ^s	τ^t
ta24_t120_normal	**62.50**	67.50	**87.67**	106.89	**4.61**	6.58	1.97	-	-
ta48_t240_normal	**66.04**	76.25	**262.98**	362.66	**15.67**	36.18	20.51	-	-
ta72_t360_normal	**60.28**	69.44	**376.97**	512.09	**21.65**	55.86	34.21	-	-
ta96_t480_normal	**69.69**	78.23	**570.39**	792.53	**25.79**	88.98	63.19	-	-
bcn_normal	**46.67**	51.51	**200.36**	242.1	**0.08**	0.82	0.74	-	-
ta24_t120_high	**86.67**	92.50	290.36	**279.45**	77.69	**48.21**	-29.48	-	-
ta48_t240_high	**83.54**	89.38	**462.05**	670.61	**71.76**	165.66	93.90	-	-
ta72_t360_high	**84.17**	91.67	**708.97**	1128.36	**131.77**	337.09	205.32	-	-
ta96_t480_high	**85.52**	92.81	**1015.62**	1620.01	**154.56**	509.48	354.92	-	-
bcn_high	**61.72**	69.35	**423.13**	541.47	**9.65**	21.67	12.02	-	-
ta24_t120_det	25.00	25.00	32.00	32.00	1.00	1.00	0.00	21	33
ta48_t240_det	41.67	41.67	164.00	164.00	2.00	2.00	0.00	558	120
ta72_t360_det	40.28	40.28	252.00	252.00	2.00	2.00	0.00	961	415
ta96_t480_det	53.13	53.13	370.00	370.00	3.00	3.00	0.00	1171	322
bcn_det	22.58	22.58	71.00	72.00	0.00	0.00	0.00	327	270

In the deterministic case, the KPIs are the same since there are enough teams to perform the given tasks on time, no matter what the objective is. In the normal and high variability cases, except one, we observe that our approach maximizing slack outperforms the typical minimization of traveling time. In the real case of BCN, we only have partial information provided by a ground handling company, and the instance does not correspond to the whole operation at the airport —i.e., it only includes the flights corresponding to the airlines currently having a contract with the ground handler. This implies a lower arrival frequency in the instance, causing long idle times between the majority of tasks. In these cases, our approach is not significantly better than simply minimizing traveling time, as long as variability remains low. However, as aircraft arrival frequency and variability increase, we observe a significant difference between the two objectives. Figure 3 shows how our approach is able to outperform travel time minimization, reducing the total delay across all turnarounds. This figure also shows that our approach provides more predictable delays, with a more contained spread across simulations for all instances.

Table 2 also includes other important key performance indicators (KPIs), such as percentage of delayed turnarounds and total minutes of delay exceeding the *on time* threshold of 15 minutes. The latter is a big concern for SPs, since
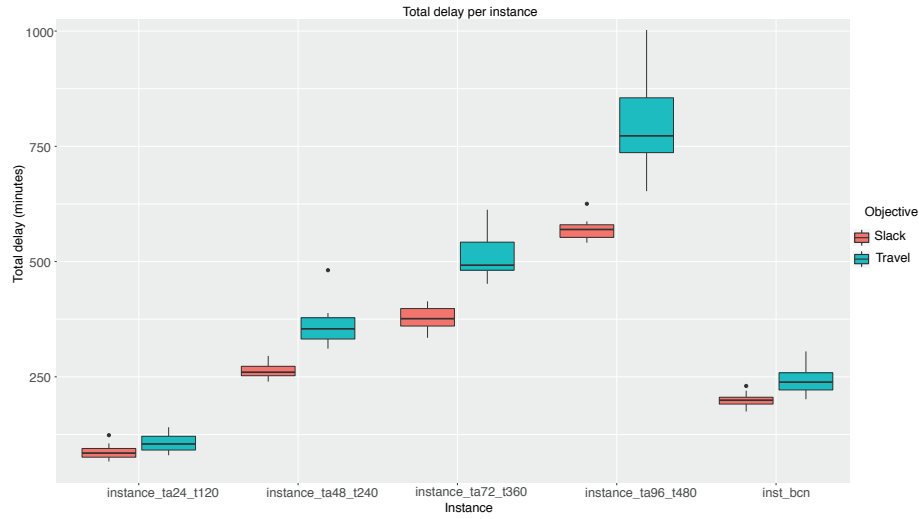
Fig. 3. Total delay per instance among 10 simulation replications

failing to meet this target carries penalties and potential further delays due to air traffic management. Our approach clearly reduces the total delay over 15 minutes, for up to 70% over the day of operation, except for one instance under high variability. Considering each minute of delay incurs a cost, deploying our approach in real-life scenarios could potentially result in significant cost savings.

6   Conclusion

In this work, we proposed a novel two-step solution approach to the airport ground service scheduling and team planning problem. With respect to earlier approaches, the RCPSP step allows service providers to operate their busy schedules with potentially fewer human resources. Our focus on maximizing minimum slack in the second step ensures they can do so efficiently. Our simulation proves the robustness of our approach. Still, tighter links between simulation and the optimization components could help, in the future, to enhance the performance of our LNS-based approach, e.g., by generating cuts from the simulation results for the benefit of the heuristic search component or using simulation within the CP search strategy.

References

1. Andreatta, G., Capanna, L., De Giovanni, L., Monaci, M., Righi, L.: Efficiency and robustness in a support platform for intelligent airport ground handling. Journal of Intelligent Transportation Systems: Technology, Planning, and Operations **18**(1), 121–130 (2014). https://doi.org/10.1080/15472450.2013.802160

2. Blazewicz, J., Lenstraand, J., Rinnooy Kan, A.: Scheduling subject to resource constraints: classification and complexity. Discrete Applied Mathematics **5**, 11–24 (1983)

3. Chu, G.: Improving combinatorial optimization. Ph.D. thesis, The University of Melbourne (2011), http://hdl.handle.net/11343/36679

4. Desrosiers, J., Dumas, Y., Solomon, M.M., Soumis, F.: Chapter 2 time constrained routing and scheduling. In: Network Routing, Handbooks in Operations Research and Management Science, vol. 8, pp. 35 – 139. Elsevier (1995). https://doi.org/https://doi.org/10.1016/S0927-0507(05)80106-9

5. Eurocontrol: Airport Collaborative Decision Making (A–CDM) (2018), http://www.eurocontrol.int/articles/airport-collaborative-decision-making-cdm

6. Fan, W., Xue, F.: Optimize cooperative agents with organization in distributed scheduling system. In: International Conference on Intelligent Computing. pp. 502–509. Springer (2006). https://doi.org/10.1007/978-3-540-37275-2_61

7. Fink, M., Desaulniers, G., Frey, M., Kiermaier, F., Kolisch, R., Soumis, F.: Column generation for vehicle routing problems with multiple synchronization constraints. European Journal of Operational Research **272**(2), 699–711 (2019). https://doi.org/10.1016/j.ejor.2018.06.046

8. Gecode Team: Gecode: generic constraint development environment (2017), http://www.gecode.org

9. Gurobi: Gurobi software, http://www.gurobi.com/

10. Ip, W.H., Wang, D., Cho, V.: Aircraft ground service scheduling problems and their genetic algorithm with hybrid assignment and sequence encoding scheme. IEEE Systems Journal **7**(4), 649–657 (2013). https://doi.org/10.1109/JSYST.2012.2196229

11. Kuster, J., Jannach, D.: Handling airport ground processes based on resource-constrained project scheduling. In: Advances in Applied Artifical Intelligence. pp. 166–176 (2006). https://doi.org/10.1007/11779568_20

12. van Leeuwen, P., Witteveen, C.: Temporal decoupling and determining resource needs of autonomous agents in the airport turnaround process. In: 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology. vol. 2, pp. 185–192 (2009). https://doi.org/10.1109/wi-iat.2009.149

13. Mao, X., Roos, N., Salden, A.: Distribute the selfish ambitions. In: Belgian/Netherlands Artificial Intelligence Conference. pp. 137–144 (2008)

14. Mao, X., Ter Mors, A., Roos, N., Witteveen, C.: Agent-based scheduling for aircraft deicing. In: Proceedings of the 18th Belgium-Netherlands Conference on Artificial Intelligence. pp. 229–236. BNVKI (2006)

15. Matl, P., Hartl, R., Vidal, T.: Workload equity in vehicle routing problems: A survey and analysis. Transportation Science **52**(2), 239–260 (2018). https://doi.org/10.1287/trsc.2017.0744

16. Neiman, D.E., Hildum, D.W., Lesser, V.R., Sandholm, T.W.: Exploiting meta-level information in a distributed scheduling system. In: Proceedings of the National Conference on Artificial Intelligence. vol. 1, pp. 394–400 (1994)

17. Nethercote, N., Stuckey, P., Becket, R., Brand, S., Duck, G., Tack, G.: Minizinc: Towards a standard CP modelling language. In: Bessiere, C. (ed.) Proceedings of the 13th International Conference on Principles and Practice of Constraint Programming. LNCS, vol. 4741, pp. 529–543. Springer-Verlag (2007)

18. Norin, A., Yuan, D., Granberg, T.A., Värbrand, P.: Scheduling de-icing vehicles within airport logistics: a heuristic algorithm and performance evalua-

tion. Journal of the Operational Research Society **63**(8), 1116–1125 (2012). https://doi.org/10.1057/jors.2011.100

19. Norin, A., Granberg, T.A., Värbrand, P., Yuan, D.: Integrating optimization and simulation to gain more efficient airport logistics. In: Eighth USA/Europe Air Traffic Management Research and Development Seminar (2009)
20. Padron, S., Guimarans, D.: Using simulation for evaluating ground handling solutions reliability under stochastic conditions. In: 2018 ROADEF Lorient, France. pp. 1–6 (2018)
21. Padron, S., Guimarans, D., Ramos, J.J., Fitouri-Trabelsi, S.: A bi-objective approach for scheduling ground-handling vehicles in airports. Computers & Operations Research **71**, 34–53 (2016). https://doi.org/10.1016/j.cor.2015.12.010
22. Solomon, M.M., Desrosiers, J.: Survey paper—time window constrained routing and scheduling problems. Transportation Science **22**(1), 1–13 (1988). https://doi.org/10.1287/trsc.22.1.1
23. Trabelsi, S.F., Mora-Camino, F., Padron, S.: A decentralized approach for ground handling fleet management at airports. In: 2013 International Conference on Advanced Logistics and Transport, ICALT 2013. pp. 302–307 (2013). https://doi.org/10.1109/ICAdLT.2013.6568476