



THE UNIVERSITY *of* EDINBURGH

Edinburgh Research Explorer

Constraint-based robust planning and scheduling of airport apron operations through simheuristics

Citation for published version:

Gök, YS, Padrón, S, Tomasella, M, Guimarans, D & Ozturk, C 2022, 'Constraint-based robust planning and scheduling of airport apron operations through simheuristics', *Annals of Operations Research*.
<https://doi.org/10.1007/s10479-022-04547-0>

Digital Object Identifier (DOI):

[10.1007/s10479-022-04547-0](https://doi.org/10.1007/s10479-022-04547-0)

Link:

[Link to publication record in Edinburgh Research Explorer](#)

Document Version:

Publisher's PDF, also known as Version of record

Published In:

Annals of Operations Research

General rights

Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy

The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.





Constraint-based robust planning and scheduling of airport apron operations through simheuristics

Yagmur S. Gök¹ · Silvia Padrón² · Maurizio Tomasella¹ · Daniel Guimarans³ · Cemalettin Ozturk⁴

Accepted: 18 January 2022
© The Author(s) 2022

Abstract

Scheduling aircraft turnarounds at airports requires the coordination of several organizations, including the airport operator, airlines, and ground service providers. The latter manage the necessary supplies and teams to handle aircraft in between consecutive flights, in an area called the airport ‘apron’. Divergence and conflicting priorities across organizational borders negatively impact the smooth running of operations, and play a major role in departure delays. We provide a novel simulation-optimization approach that allows multiple service providers to build robust plans for their teams independently, whilst supporting overall coordination through central scheduling of all the involved turnaround activities. Simulation is integrated within the optimization process, following simheuristic techniques, which are augmented with an efficient search driving mechanism. Two tailored constraint-based feedback routines are automatically generated from simulation outputs to constrain the search space to solutions more likely to ensure plan robustness. The two simulation components provide constructive feedback on individual routing problems and global turnaround scheduling, respectively. Compared to the state-of-the-art approach for aircraft turnaround scheduling and routing of service teams, our methodology improves the apron’s on-time punctuality, without the

✉ Yagmur S. Gök
yagmur.gok@ed.ac.uk

Silvia Padrón
s.padron@tbs-education.fr

Maurizio Tomasella
maurizio.tomasella@ed.ac.uk

Daniel Guimarans
daniel.guimarans@gmail.com

Cemalettin Ozturk
cemalettin.ozturk@mtu.ie

¹ Business School, University of Edinburgh, 29 Buccleuch Place, Edinburgh EH8 9JS, UK

² TBS Business School, 20 Boulevard Lascrosses, 31068 Toulouse, France

³ Amazon, 22 Rue Edward Steichen, 2540 Luxembourg, Luxembourg

⁴ Process, Energy and Transport Engineering, Munster Technological University, Bishopstown, Cork T12 P928, Ireland

need for the involved organizations to share sensitive information. This supports a wider applicability of our approach in a multiple-stakeholder environment.

Keywords Simheuristics · Simulation · Optimization · Large neighborhood search · Robust scheduling · Constraint programming

1 Introduction

Airports involve many stakeholders and require tight coordination between them to achieve safe and efficient operations. Third-party service providers (SPs) normally supply ground handling services such as refueling, baggage loading/unloading, catering, etc., whilst satisfying given service levels, as agreed between themselves and the airlines. Competition among SPs at the same airport is often fierce, and risks of losing a contract with a carrier are not negligible. Legislative frameworks, such as the European Union's Ground Handling Directive (Official Journal 1996), have been in place to foster fair competition at the level of each airport. Before 1996, airlines tended to suffer from the monopoly of a single handling company and high prices, and switching SP due to reasons such as poor service delivery was not as easy. In the USA, airline-owned SPs are often employed to serve their own fleet. Irrespective of ownership structure, tight coordination is required across the many SPs covering, at the same airport, the various services needed by an aircraft between two consecutive flights. This set of activities goes by the name of *aircraft turnaround*, and possesses the features of a typical resource-constrained project scheduling problem (RCPSP) (Blazewicz et al., 1983), with precedence relationships among the composing tasks and the urge to minimize the makespan of all individual projects, to facilitate aircraft on-time departure. Internal coordination within a single SP is required to efficiently manage both the human resources (generally organized in teams, the size of which depends on the specific service type) and technical assets, as required across the many aircraft to be serviced in a shift/day. Over and above, ground handling operations do strongly depend on decisions made by other stakeholders operating at the same airport, such as the airport operator and the air traffic service providers.

All the above taken, it comes as no surprise that the planning and scheduling of airport ground handling tasks and teams have surfaced in the scholarly literature over the past 25 years—e.g., Norin et al. (2009) and Kuster and Jannach (2006). Especially, robust scheduling ahead of the day of operations is crucial for both SPs and airlines to keep any rescheduling needs on the day of operations within reasonable limits. In an environment as dynamic as an airport, things will surely get out of hand every now and again, due to operational delays, either locally or across the wider air transport network. But some of the reasons for such dependent delays can be modeled, and decisions that are likely to guarantee certain levels of robustness in the sense just stated can be made in advance.

Figure 1 shows an example turnaround. The aircraft remains at its parking stand—in an area of the airport called *apron*—between the scheduled in-blocks time (SIBT) and the scheduled off-blocks time (SOBT), at which point it is normally pushed back into the taxiway to then proceed to the runway for take-off. These are the times when chocks (large rubber or wooden wedges) are placed (SIBT) and removed from (SOBT) around each wheel, to keep the aircraft from moving. Color-coded thin bars represent the time windows, according to plans and any other constraints (e.g., boarding may start only as far back from the SOBT as considered desirable) within which activities should take place. Thicker bars show planned durations, start and end times for all activities. These 'as-planned' durations follow, in gen-

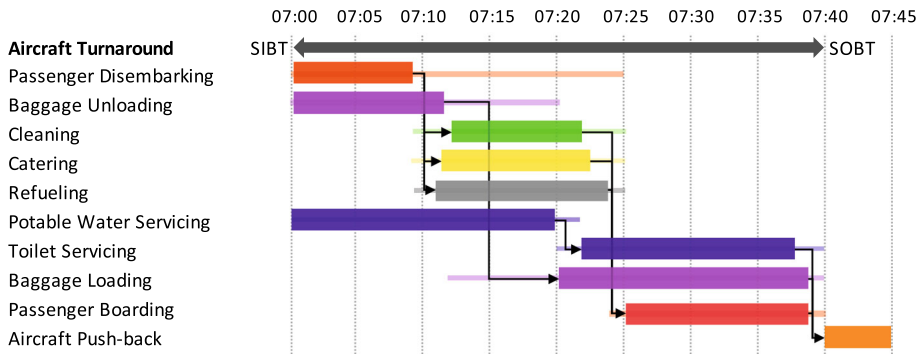


Fig. 1 A typical turnaround (arcs denote precedence relations)

eral, the standard operating procedures known for each aircraft. Precedence relations are shown as arrows linking activities. To guarantee on-time departures, tardiness of all such turnarounds/projects must be minimized. The On-Time Departure Performance of an airport overall (OTDP, roughly the % of aircraft leaving up to 15 minutes past their scheduled time of departure) is one of the most visible indicators of how well it functions, and is affected by factors that go beyond activity dependence and delays within each turnaround, thus becoming a complex issue for airport managers to look at.

From each SP's viewpoint, achieving timely completion of every service to every aircraft requires accurate orchestration of both the human resources (teams of staff) available on shift, as well as the related technical assets (e.g., baggage loading/unloading belts, push-back trucks, etc.). The latter go by the name of ground support equipment (GSE). Given the tight competition in the ground handling market, costs become crucial, beginning with both staffing and GSE levels, and resources are often scarce. Robust scheduling of all resources becomes key to each SP. But this is not enough. The dependencies among the activities of each turnaround, together with the involvement of multiple SPs, suggests that cross-turnaround coordination should be sought. This might help, if not to eliminate delay propagation (unlikely, in real world environments), at least to manage it in a way that minimizes it and/or to help to regain control of operations quicker after major operational disruptions (Ball et al. 2007; Evler et al. 2021). In this paper, we focus on the management of turnaround teams, their timely delivery of each activity, and movement between turnarounds. This adds another level to the RCPSM mentioned above. More precisely, it requires the modeling and solution of vehicle routing problems with time-windows (VRPTW) (Solomon and Desrosiers 1988), covering the many types of resources (teams) involved. Hence, each of the SP teams needs to be routed robustly in a way that keeps at least the knock-on effects in delay propagation within reasonable control.

In the last two decades, an approach to handle airport operations, including apron operations, has been put in place at some of the major European airports (30 in total, at the time of writing). This is called Airport Collaborative Decision Making (A-CDM) (Eurocontrol 2018), and aims at enhanced information sharing between all airport stakeholders. Information to be shared includes relevant time stamps around the status of all activities, and the establishment of certain milestones or key times for each turnaround, around which everything else is organized. This scenario is viable where it can be enforced, such as the biggest and most complex airports. Most airport realities do not implement A-CDM, which raises

the question of how to support robust scheduling of apron resources outwith A-CDM. For state-of-the-art approaches to A-CDM airport settings, see Evler et al. (2021).

We propose a new approach which helps the SPs to make individual decisions about the optimal routing of their turnaround teams, whilst achieving airport-wide coordination through robust scheduling of the same activities across all SPs. We handle these two optimization components via deterministic optimization approaches that involve constraint programming (CP) (Hentenryck 2002) and metaheuristics, in a framework that will be clarified later. We ensure desired degrees of plan robustness by accounting for the non-deterministic elements of both problems, through stochastic simulation of each component. We achieve the hybridization of optimization and simulation by borrowing ideas from the family of techniques known as Simheuristics (Juan et al. 2015), which we enhance through a tighter link between the simulation and the optimization processes. In particular, we set a feedback mechanism between each of the two simulation stages in our procedure and corresponding points in the optimization sub-routines, to intelligently guide the next stage of optimization towards a likely more robust solution. Earlier versions of our ideas are available in Gök et al. (2020a) and Gök et al. (2020b), but only in this paper we fully develop the two feedback mechanisms and show their benefits to the robust planning of apron operations.

In Sect. 2 we discuss related works along two lines: (a) the different ways in which simulation and heuristics have been hybridized, within Simheuristics; and (b) modeling and solution of scheduling and planning problems related to airport ground handling services in non-A-CDM settings, which represents our focus. We also clarify our contributions in both these respects. Sect. 3 proposes compact mathematical models for both the scheduling of turnarounds and routing of all teams of SPs concurrently operating at the same airport. In Sect. 4, we discuss our simheuristic approach and emphasize the novel aspect that centers on our feedback mechanism from simulation back to optimization. Before that, we present and justify our choice of robustness metrics in detail. Our extensive experimental analysis is provided in Sect. 5, in which we also compare our simheuristic algorithm to the closest approach from previous literature (Padrón et al. 2016). Finally, Sect. 6 wraps up major learning points from our study.

2 Related literature and paper contributions

2.1 Simheuristics and simulation-optimization

Simheuristics have become a prominent field of research and many scientific works have adopted them to solve stochastic combinatorial problems efficiently, routing problems in particular (Juan et al. 2015; Chica et al. 2020). Hybridizing simulation with metaheuristics provides a framework for assessing solutions under uncertainty and guidance through the search space based on this assessment. This way, large-scale, complex problems are solved with less computational effort than classical stochastic optimization methods (Pagès-Bernaus et al. 2019; Juan et al. 2021).

In many works within the simheuristics domain, simulation is embedded in two steps of the heuristic method: (i) in the move acceptance component, to drive the search by roughly measuring the quality of the deterministic solution in stochastic scenarios, and (ii) at the end of the optimization process, to perform an even more detailed assessment. In the former, simulation is launched after completing each local search process, when the deterministic cost of the obtained solution is improved. A small number of simulation runs are conducted to

estimate the stochastic cost, by exchanging the deterministic parameters with pseudo-random numbers. Should the stochastic version of the objective function result in an improvement, the solution is set aside to be intensively simulated in the second step, once the ‘optimization’ process has been completed.

This recurring scheme is employed e.g., by Gragas et al. (2016), where a Monte Carlo (MC) sampling method is embedded within iterated local search (ILS). The combination of MC and ILS is also adopted by Guimarans et al. (2018) to solve the vehicle routing problem with packing constraints and varying travel times. In de Armas et al. (2017) the facility location problem with uncertain service costs is addressed in a similar fashion. Besides determining the best solution in terms of expected costs, the simulation cycle produces solutions with the highest performance according to standard statistical measures such as standard deviation or quantiles. Likewise, this same strategy has been used to incorporate simulation within other heuristics or metaheuristics, such as the savings-based heuristic (Gonzalez-Martin et al. 2018), variable neighborhood search (Gruler et al. 2018, 2020), the greedy randomised adaptive search procedure (Ferone et al. 2019), and genetic algorithms (Rabe et al. 2020). In the latter though, the realizations of the random variables are obtained by applying a discrete-event simulation (DES).

Irawan et al. (2021) follow a similar configuration to schedule wind turbines’ maintenance combining large neighborhood search (LNS) with MC sampling at the usual two stages, with slight differences. Stochastic variables are replaced by the value corresponding to a given quantile. In addition, only a single optimal solution for the stochastic problem is stored and further tested. A one-stage combination of MC sampling and a genetic algorithm is presented by Rabbani et al. (2019) in the context of a multi-objective problem. Stochastic scenarios generated by a sampling procedure are used to evaluate the fitness functions each time a new population of solutions is created. Latorre-Biel et al. (2020) also enclose MC sampling in the acceptance criterion to solve the vehicle routing problem with correlated stochastic demands. Since customers’ requests are mutually dependent, a Petri net-based process is run, using the generated local optimal solution to predict the expected demands for the next iteration.

In our view, the main shortcoming in the existing simheuristics literature is that simulation is deployed to accept or reject local minima on the basis of expected costs. This way, it provides a limited mechanism to guide the exploration of the solution space, which is likely to lead to a larger number of replications without any guarantee of reaching an improvement. Less than a handful of works exploit a deeper integration between the simulator and the heuristic method. Guimarans et al. (2015) hybridize LNS, CP and MC sampling to address the aircraft recovery problem with stochastic flight and turnaround durations. During the exploration phase, a solution is accepted if a robustness criterion is satisfied. If it is rejected, aircraft are re-scheduled by constraining the search space to solutions with a worse objective function but providing a more robust performance. Keskin et al. (2021) looks at solving electric vehicles’ recharge scheduling with random waiting times at the recharge stations. The authors develop an adaptive LNS (ALNS) with different destroy and repair operators to improve the initial routing solution. Through these operators, customers are ‘unrouted’ and reinserted depending on the probability and the cost of violating the time windows, calculated by means of a DES.

Besides simheuristics, other simulation-based optimization approaches provide valuable alternatives to tackle our problem. The discrete event optimization methodology from Pedrielli et al. (2018) includes the stochastic events in the mathematical optimization model using decision variables. Even though Benders cuts can be applied to restrict the feasible region (Zhang et al. 2018), the need to include all relevant simulation events in the optimization makes this methodology impractical for our apron planning problem due to its

size and the high dimensionality of the state space. Scala et al. (2021) suggest influencing the optimization outputs making a predefined adjustment of the input parameters of the optimization model and the solving method if aircraft movement conflicts exceed a specified threshold after simulation. However, guiding exploration by applying general calibrations—i.e., modifying weighted objective functions, relaxing optimization constraints or increasing search intensification—without targeting problematic allocations can result in excessive execution times and sub-optimal solutions.

With the present study, we seek to improve simheuristic feedback effectiveness through an enhanced search driving structure, picking up on some of the ideas shown by Guimarans et al. (2015) and developing them further. We take advantage of the ability of CP to discard unpromising regions and manage constraints efficiently. Our algorithm will then learn and generate tailored constraints from the simulation outcomes, during execution time, to prevent schedules being produced with unsuitable assignments or features likely to affect robustness. We also integrate two simulation methods for reasons we will clarify in Sect. 4, while the simheuristic literature conventionally adopts only one.

2.2 Apron operations scheduling

Research efforts around apron equipment and staff (teams) planning have been broadly evolving in two directions: *non-collaborative* vs. *collaborative* approaches. The former look to achieve optimized plans for a particular apron service individually, such as transfer buses (Zhao et al. 2021), push-back tractors (Bubalo et al. 2017), assistants for reduced-mobility passengers (Grahn and Jacquillat 2020), or ground handling workers in general (Hur et al. 2019). They are most suited to cater for the needs of a single SP. Collaborative approaches pursue effective coordination between operations, and has been recognized as an important subject for upcoming investigation (Schultz et al. 2020; Wang et al. 2021). Padrón et al. (2016) introduce the most prominent collaborative resource allocation approach of the airport ground handling literature, later enhanced by Padrón and Guimarans (2019) to reduce computational times. Several types of apron services are optimized independently, but local decisions are combined with a central view to guarantee the optimization of an overall objective. Apron resources are allocated in sequence, solving a VRPTW per resource type, given a time window configuration. Time windows are centrally established, employing a constraint satisfaction model, and are systematically adapted to ensure coherence across the local routing solutions. The method measures the impact of the optimal allocation of one apron service on the other services' available starting time. Depending on this impact, the VRPTW sequence is rearranged to generate different trade-off solutions that improve the global handling schedule, whilst minimizing the completion time of all involved turnarounds. As a downside, routing decisions become highly dependent, making the approach suitable for the cases where a unique SP supplies all handling operations to flights. Our research intends to be an evolution of this methodology, particularly by removing the latter limitation.

We discussed from the outset that multiple SPs normally operate at the same airport, something which Padrón et al. (2016) does not support directly. Cross-SP collaboration has been handled elsewhere, with the SPs, airport operator, and air traffic service providers modeled as a set of interactive agents. Within this scheme, centralized management is implemented by Fitouri-Trabelsi et al. (2015) via a leader agent, ensuring a smooth running of all apron-wide operations. Kabongo et al. (2016) propose a distributed control system for the same setting.

Planning multiple kinds of resources from different SPs requires sharing sensitive knowledge and information (e.g., asset location in real-time) between the concerned actors—a significant limitation. An alternative protocol is discussed by Tomasella et al. (2019) to plan turnaround teams under uncertainty. A central, airport-wide, entity verifies the suitability of the schedule developed independently by each SP, and feeds back to them certain performance estimates to suggest where a different plan would suit each SP, or the airport, any better. Simulation is enclosed in the local optimization process at each SP, using a simheuristic under the classical scheme extensively discussed above.

Building upon the learning points from the cited papers, in the present work we propose a hybrid planning structure that supports multiple SPs. Each SP allocates its resources (teams) in a decentralized manner, without the need to disclose to competing SPs delicate details such as workforce composition or capacity of each technical turnaround asset, whilst following centrally-generated time window constraints to ensure coordination at the airport level. Contrary to Padrón et al. (2016), time windows are fixed and are optimally set through an RCPSP model targeting on-time performance of all the involved turnarounds. In turn, each SP looks at maximizing slack times of their teams' routes without affecting the consistency or the punctuality of the overall handling schedule.

3 Problem formulations

In this section, we formulate the two sub-problems that together characterize the robust scheduling of aircraft turnaround tasks and routing of the involved teams of staff across an apron. We introduce updated formulations based on those originally presented in Gök et al. (2020a). Both the overall problem and the two sub-problems we formulate are, in practice, of stochastic nature. In this section, we follow the approach that is typical of simheuristics—see Figures 1 and 3 in Juan et al. (2015), and present simplified deterministic formulations of the two sub-problems. We transform the two original stochastic combinatorial optimization problems into their deterministic equivalent versions, and explain how we obtained the related deterministic parameters. We will reintroduce the stochastic aspect of both sub-problems in when discussing our specific simheuristic algorithm in Sect. 4.

Both sub-problems are modeled through CP, which has a proven track record in solving scheduling and routing problems. More specifically to our work, CP has also been proved (Guimarans et al. 2015) to work very well within simheuristic approaches, especially if paired with metaheuristics such as LNS. Furthermore, CP allows the use of 'global constraints' to make the model formulation more efficient and the solution process more effective (Beldiceanu et al. 2012), a characteristic we will also exploit.

Figure 2 illustrates an example of how scheduling turnaround tasks and team routing sub-problems are deployed together to solve the apron operations problem for a subset of four turnarounds running concurrently. Each turnaround requires four types of tasks to handle the corresponding aircraft, where TA_1 represents the turnaround 1 and $t_{1,2}$ the task 1 from turnaround 2. Scheduling the tasks takes the form of an RCPSP and determines the starting times of all the operations involved that minimize tardiness and the required number of resources (teams), through a lexicographic approach. Keeping the obtained start times fixed, teams belonging to the same type and the same SP will be routed independently, by sequentially maximizing the minimum slack, the workload balance and the total slack of the routes. The outcome of this step is pictured on the right of the figure, where $\tau_{1,2,1}$ is a team

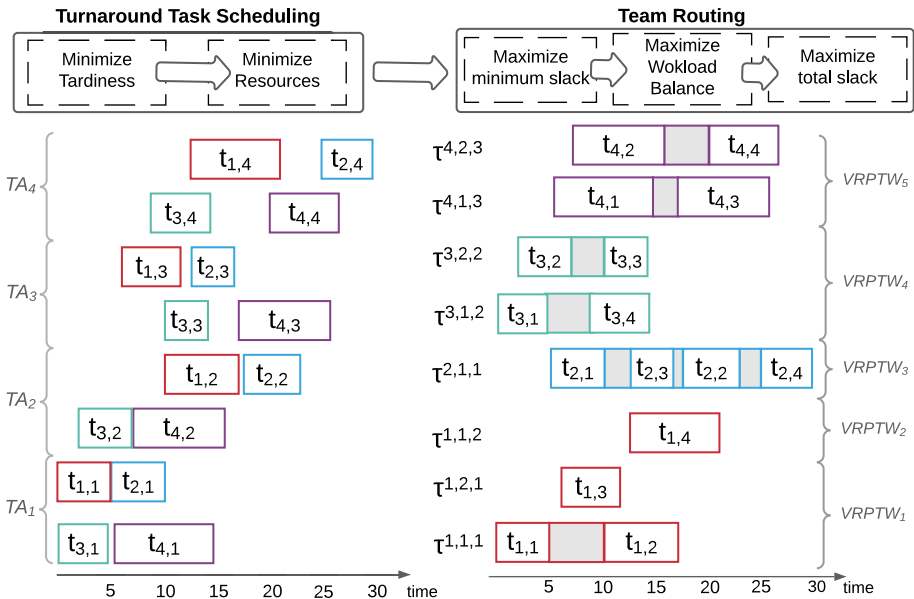


Fig. 2 Turnaround task scheduling and team routing for planning apron operations

of type 1, and more specifically is number 2 in the set of teams of type 1 provided by the SP 1.

Eight teams, across the four team types, and belonging to three SPs, are allocated to the planned tasks, given that SP 1 supplies services 1 and 2 and SP 2 supplies services 1 and 3, whilst service 4 is provided by only SP 3. For instance, routing decisions related to $t_{1,1}$, $t_{1,2}$, and $t_{1,3}$ are made by solving a single VRPTW, as all such tasks carried out by the same team type (i.e., team type 1) and supplied by the same SP (i.e., SP 1). On the other hand, task $t_{1,4}$ is allocated through a separate VRPTW, as it relates to a turnaround serviced by another SP (specifically, by SP 2). As a result, two teams of type 1 from SP 1 are required for performing $t_{1,1}$, $t_{1,2}$, and $t_{1,3}$ whereas one team of SP 2 processes $t_{1,4}$. The gray box between the tasks stands for the resulting slack between two consecutive visits in a team route, which represents the principal decision made at the team routing stage.

3.1 Turnaround task scheduling

We look at all composing tasks of all aircraft turnarounds happening at a given airport over a time horizon of length t_{max} —say between a few hours and a whole day of the airport timetable. The aim is to achieve early coordination—ahead of the day of operation—by scheduling task start times, denoted as $start_i \in [0, t_{max}]$, $i \in I = \{1 \dots v\}$, where v is the number of tasks, to minimize tardiness of aircraft turnarounds whilst assessing the required staffing levels to be guaranteed by all involved SPs. Each task represents a specific activity type $a \in A = \{1 \dots u\}$ such as refueling, baggage loading/unloading, passenger boarding/disembarking, etc., where u is the number of activity types. $I_a \subset I$ denotes the set of all tasks of type a . The flight timetable at the given airport provides the needed Scheduled Time of Arrival (STA) and Scheduled Time of Departure (STD) of all aircraft, off of which earliest start sta_i and latest finish times std_i for all tasks can be worked out. In most cases, it will be $sta_i = STA$

and $std_i = STD$ for all i that belong to the same aircraft. Parameters $duration_i$ denote the estimates for the processing time of the related tasks, represented by the expected value of the related probability distributions. The precedence relations between tasks are denoted through a successor set S_i linked to each task i , hence $j \in S_i$ iff task i precedes task j ($i \rightarrow j$). In practice, in any aircraft turnaround there exists only one task for which $S_i = \emptyset$, and that is the push-back task (Fig. 1). Some task types, such as clean water supply and waste disposal, cannot be carried out concurrently, as they both need access to the same physical space. $P = \{1 \dots d\}$ is the set of such forbidden pairings, where d is the total number of pairs, and D_p is the set of (two) tasks for each $p \in P$.

Sets SO and SI represent, respectively, activity types that must be completed no earlier than a certain time (which we denote as $sobt_a$) before std_i , or begin right after sta_i . For instance, boarding cannot start earlier than a set time—say 40 minutes—before the STD, while disembarking could in theory start as soon as the aircraft reaches its stand.

The scheduling sub-problem is tackled in two stages. In the first stage, we formulate it as a CP model as follows:

$$W_1^* = \min W_1 = \min \sum_{i \in I: S_i = \emptyset} c_t \times \max\{0, start_i + duration_i - std_i\} \quad (1a)$$

$$start_i \geq sta_i, \quad \forall i \in I \quad (2)$$

$$start_j \geq start_i + duration_i, \quad \forall i \in I, \forall j \in S_i \quad (3)$$

$$start_i + duration_i \geq std_i - sobt_a, \quad \forall a \in SO, \forall i \in I_a \quad (4)$$

$$start_i = sta_i, \quad \forall a \in SI, \forall i \in I_a \quad (5)$$

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

Objective function (1a) minimizes the cost of tardy turnarounds, with c_t representing the unit cost per aircraft per minute of tardiness, which occurs, for any aircraft, when the STD is exceeded by the finish time of the push-back task. Constraints (2) set the lower bound to $start_i$, while constraints (3) enforce task precedence relations. Constraints (4) and (5) further define the feasible start times of certain tasks, according to the related sta_i and std_i . The CP global constraint *disjunctive* (6) restricts certain pairs of tasks from taking place simultaneously (Beldiceanu et al. 2012).

To this point, an uncapacitated project scheduling problem (PSP) has been solved, providing the minimum possible tardiness for the overall system (i.e., the apron). SP resources are clearly finite, though, which makes the actual problem an RCPSP. We tackle this in the second stage.

Each activity type—hence each task—requires a certain number of teams $rr_{k,i}$ of some type $k \in K = \{1 \dots nRes\}$ —often one team per activity type, but some activities (e.g., baggage unloading/loading) might require the coordination of different teams. Activity types of the same nature — say refueling— but carried out by separate SPs correspond to separate team types. This allows for covering, in this second stage, the scheduling of all team types of all SPs (the total number of which is $nRes$). Also, any team of any SP can belong to only one team type or, equivalently, performs only one activity type. This means that the human resources in our problem are somewhat dedicated to a specific activity type, on the given day of operation. Decision variables rc_k represent the number of teams of type k required throughout the planning horizon. Setup times $setup_i$ represent the sum of two components: (a) the travel time for a team to move to the next stand where it will perform task i ; and (b) the setup time for the same team to get ready to start actual operations at the new stand (e.g.,

due to positioning of equipment). Parameters $setup_i$ can be estimated as follows. From the two distributions of components (a) and (b) for task i the overall distribution of $setup_i$ can be obtained, e.g., via convolution. At that point, $setup_i$ can be set equal (as we did) to the expected value of the related distribution. In our case study, we only consider component (a) for $setup_i$ and further simplify matters by substituting $setup_i$ for all i requiring a resource, with a constant, which is the median value of the traveling times between every gate that is used during the given period across the apron.

The objective of this second stage aims at minimizing the overall number of teams needed to perform all the required tasks across all turnarounds, while respecting the minimum tardiness obtained through solving the first stage.

$$W_2^* = \min W_2 = \min \sum_{k \in K} rc_k \quad (1b)$$

$$W_1 = W_1^* \quad (7)$$

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

Objective function (1b) minimizes requirements for turnaround teams across all SPs, subject to constraints (2)–(6) and the addition of constraints (7) and (8). Constraint (7) ensures tardiness is kept at the minimum based on the previous result with objective (1a), while the CP global constraint *cumulative* (Beldiceanu et al. 2012) ensures resource capacity is not exceeded at any time, for any resource type (8).

3.2 Team routing

With the schedule of all turnaround tasks centrally developed, each SP knows at what time any one of its teams must be ready to start performing each task (i.e., $start_i$, for all i). They also know how many teams are required for each of their services, for the airport overall to meet desired performance levels. Each SP can now optimally route its own teams to cover the planning horizon. We assume that all teams carry the GSE they need with them from one aircraft to another or, if the GSE is a vehicle (true in most cases), they move with the vehicle across the apron all the time. Therefore, we do not model GSE directly.

We support team routing decisions by modeling the problem as a VRPTW or a traveling salesman problem with time-windows (TSPTW). The former is adopted when the GSE operated by the team has limited capacity—e.g., catering trucks—and is likely to require replenishment over the planning horizon. The latter is adopted otherwise. The initial GSE capacity is set to cap . We denote as q_i the capacity left at the end of task i , after supplying its demand $demand_i$.

Following on from PSP/RCPS, we consider strict time windows based on task duration, where the start time $start_i$ is enforced. We solve one team routing problem separately for each team type k with a route devised for each team $t \in T_k = \{1 \dots rc_k\}$, where rc_k is the total number of routes (teams) per team type. The tasks to be covered are denoted as $i \in I_k = \{1 \dots l_k\} \subset I$, where l_k represents the total number of tasks for teams of type k . We denote with set $N_k = I_k \cup \{l_k + 1, \dots, l_k + rc_k\}$ all regular visits plus some dummy visits which are the end visit of each route. For each task i , the SP wants to set: (1) the start time of the task, or $stime_i$; (2) the route rt_i allocated to i (note: each team has its own route); (3)

the task s_i immediately following i ; and, (4) whether to replenish or not the associated GSE before moving to s_i (through the binary variable x_i).

We use variables $slack_i$ to track the available idle time between tasks; i.e., the time between the planned completion of task i and the planned start of task s_i that is not consumed for travel or replenishment. Since our goal is to foster robustness, we use $slack_i$ variables to introduce buffers able to absorb small perturbations in the planned operations. Similarly to the PSP/RCPSPP above, we also proceed here through a lexicographic approach, this time in three stages, with the first stage aimed at maximizing the minimum slack between tasks across all turnarounds.

$$Z_1^* = \max Z_1 = \max_{i \in I_k} \min slack_i \tag{9a}$$

$$circuit \ ([s_i \mid i \in N_k]) \tag{10}$$

$$alldifferent \ ([s_i \mid i \in N_k]) \tag{11}$$

$$rt_{i+t} = t, \quad \forall t \in T_k \tag{12}$$

$$rt_{s_i} = rt_i, \quad \forall i \in I_k \tag{13}$$

$$stime_i = start_i, \quad \forall i \in I_k \tag{14}$$

$$busy_i = \begin{cases} stime_i + duration_i + \\ traveltime_{i,s_i} + x_i \times replenish, \forall i \in I_k & | \ s_i \in I_k \\ endshift, & otherwise \end{cases} \tag{15}$$

$$stime_{s_i} \geq busy_i, \quad \forall i \in I_k \tag{16}$$

$$slack_i = stime_{s_i} - busy_i, \quad \forall i \in I_k \tag{17}$$

$$q_i = cap, \quad \forall i \in N_k \setminus I_k \tag{18}$$

$$q_{s_i} = q_i - demand_{s_i}, \quad \forall i \in N_k \setminus I_k \tag{19}$$

$$q_{s_i} = \begin{cases} cap - demand_{s_i}, & \text{if } x_i = 1 \\ q_i - demand_{s_i}, & otherwise \end{cases} \quad \forall i \in I_k \tag{20}$$

We maximize the minimum slack value across all routes (teams) and tasks (9a). As discussed, short delays are absorbed and minor knock-on effects should be prevented this way (at least in the plan). The first stage constraints are (10)–(20), but constraints (18)–(20) are only considered in the VRPTWs. Global constraint *circuit* (10) builds a single overall sequence for all tasks of all routes (Beldiceanu et al. 2012). Within the sequence, dummy tasks are added to indicate the end of each route, the label of which is a function of the index of the team associated to the route, while all other tasks are regular tasks from set I_k . Dummy tasks are assigned to each route (12), and equation (13) makes sure that both i and its successor are allocated to the same route. Global constraint *alldifferent* (11) is redundant and added to help propagation by making sure all assignments for s_i are different (Beldiceanu et al. 2012). Tasks are set to start at the time imposed by the earlier centralized plan (14). Constraints (15) set, for each task, the time until which the related team will be busy, that is $busy_i$. Parameters $traveltime_{i,s_i}$ indicate the time needed for the team to move between the stand of task i and the stand of task s_i . These parameters can be easily derived from the same data and distributions that relate to component (a) of parameters $setup_i$. As before, even for $traveltime_{i,s_i}$ we adopt the expected value of the related distributions. Parameter *replenish* represents the time to replenish the GSE before visiting the next aircraft. We assume that distributions for the replenishment tasks of all GSEs are available. The deterministic parameters that are the

replenishment times for each activity type/GSE can be set to the expected value of the related distributions. In practice, we further simplified matters in our case study and, as done in previous stages for setup times, we adopt a single constant value for *replenish*, computed as the average of all replenishment parameters from across all activities/GSEs. Parameter *endshift* indicates the end of the planning horizon. The start time of the successor task s_i is set to no earlier than the time by when the related team is set free from the previous task i (16). The extent of the slack available after each task i is defined through constraints (17). Capacity constraints (18), (19) and (20) ensure that replenishment is executed only when necessary.

In the second stage, we aim at enforcing some level of balance across all teams in terms of the respective workloads (9b), which are defined as in constraints (21). Meanwhile, the value of the first stage objective function (i.e., minimum slack across routes) is kept at its maximum through constraint (22). This model also includes all constraints from (10)–(20), with equations (18)–(20) only applied in the VRPTWs. As we demonstrated in Gök et al. (2020a), workload balance does not always solve to optimality. Hence we set a time limit for solving Z_2 . On reaching this time limit, the solution process stops and we take the best solution so far, which we call \hat{Z}_2 .

$$Z_2^* = \max Z_2 = \max_{t \in T_k} (\min_{i \in I_k} workload_t - \max_{i \in I_k} workload_t) \quad (9b)$$

$$workload_t = \sum_{\substack{i \in I_k \\ \text{where } r_{i,t} = t}} duration_i, \quad \forall t \in T_k \quad (21)$$

$$Z_1 = Z_1^* \quad (22)$$

In the third and last stage, we maximize the total slack time across all tasks with objective (9c), whilst maintaining the lower bound of the second-stage objective by adding constraint (23). This final model contains all constraints from (10) to (23), with constraints (18)–(20) only enabled for the VRPTWs. Similarly to \hat{Z}_2 , we also have a time limit for solving Z_3 , at the end of which we keep the best solution so far and show it as \hat{Z}_3 . This \hat{Z}_3 is then further improved through our algorithm which will be explained in sect. 4.3.

$$Z_3^* = \max Z_3 = \max_{i \in I_k} \sum slack_i \quad (9c)$$

$$Z_2 \geq \hat{Z}_2 \quad (23)$$

4 Simulation-optimization for the robust scheduling of apron operations

4.1 Apron operational robustness and related metrics

A standard approach to assess airport operational performance is looking at figures such as the previously mentioned OTDP, which summarizes the effectiveness of airport operations from a passenger perspective. What these figures alone do not help much with is developing a thorough understanding of the real causes behind, say, a 25 minute delay on departure of a specific B737 aircraft, as many factors normally interact in determining this value. Some

of these factors may be related to the status, around the time, of the air transport network (local, national and beyond), and some others may be due to non-apron-related operations at the same airport. As a result, it may be difficult to assess the effect of apron operations *per se*, as well as where problems may lie, e.g., what SP may be more ineffective because of resource capacity constraints or a sub-par schedule of their operations. The uncertainty that is an integral component of every apron operation only complicates this performance measurement exercise.

Our solution procedure will need to assess the robustness of the turnaround schedules and team routing plans against a variation of scenarios that are representative of problem uncertainty, as they are devised by our algorithm. When robustness is achieved through the airport, our procedure terminates. We also wish to achieve robustness at each team type level. Two assessments of robustness will then take place.

After we obtain a solution for the three-stage VRPTW/TSPTW for a given team type k , we run independent replications of a stochastic simulation of the operations of that specific team type alone. We do not simulate the operations of the whole apron at this point. We refer to this simulation as $SIM(k)$. From each replication of such experiments, we measure all late starts of tasks of all teams of a given type. A delay is accounted for as such when the actual task start time of activity i exceeds $start_i$ by more than the aircraft arrival delay, if any. In fact, a delayed aircraft arrival will shift, *ceteris paribus*, the actual start of all tasks of the related turnaround, equally and to the right, something which needs to be discounted from the actual magnitude of a delay as it is due to causes beyond the control of apron operators. For each task allocated to teams of type k , we calculate the average delay across all $SIM(k)$ replications, and then take the maximum of the obtained values, which we call $maxResDelay_{SIM(k)}$.

Once routing plans are available for all team types, independent replications of a comprehensive DES of the overall apron is run based on an automatically generated Petri net model (Guimarans 2021). We refer to this simulation as $SIM(AP)$. We then compute a similar metric to $maxResDelay_{SIM(k)}$, for each team type, for exactly the same reasons, but use in this case a different notation (as the simulation itself has a different scope here), opting for $maxResDelay_k$. Unlike the $SIM(k)$ runs, $SIM(AP)$ runs are able to capture delay propagation across tasks within and across turnarounds. Figure 3 illustrates the point, by showing an example involving potable water and toilet servicing for two aircraft ('AC 1' and 'AC 2'). Time in the figure flows from left to right. Labeled double arrows indicate the late arrival of the team assigned to the task, while dotted arrows represent the transfer of a team from one parking stand to the next. We count 'A' double arrows as delays, calculated as explained for the $SIM(k)$ stage, as they are caused by the related team's late arrival. We do not count the 'B' double arrow as a delay, though. This is also a case of late arrival (of Team 2), but the actual delayed start of the toilet servicing task is in fact due to a late finish of the potable water servicing task (for which Team 1 is responsible). This is, in turn, directly associated to the propagation of the delay that the same team experienced with respect to the potable water servicing for AC 1. With a view on the overall apron operations, the maximum delay across all team types becomes meaningful in terms of assessing the robustness of the plans generated from the previous steps. We denote this as $maxResDelay_{max} = \max_{k \in K} maxResDelay_k$. This metric is especially valuable as it provides a synthetic assessment of the worst-case delay across all the team types, including any knock-on effects as explained above.

We say that a routing plan for a given resource k is *locally* robust when $maxResDelay_{SIM(k)} < \tau$, where τ is a threshold parameter. Similarly, we say that the activity schedules and routing plans for the apron on the whole are *globally* robust when $maxResDelay_{max} < \tau$. The value of τ is indeed subjective. However, we reckon that aiming to limit the maximum delay

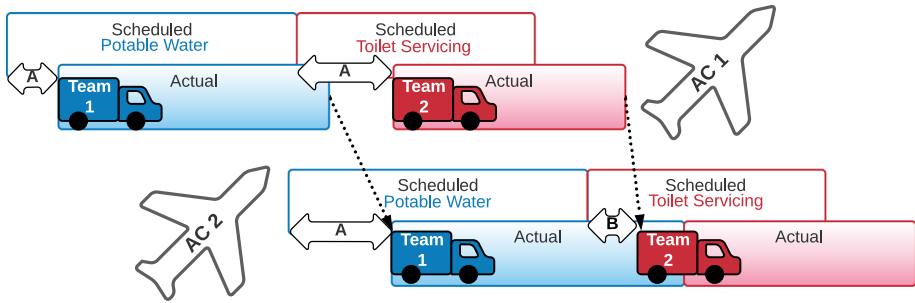


Fig. 3 Example of a delay calculation and propagation

to a set value, for either each team type or across all team types, is meaningful in at least two ways. Let us assume $\tau = 3$ (minutes). First, ensuring that all team types are likely to guarantee a maximum delay at starting their tasks of 3 minutes is a first step towards making on-time departure punctuality more likely itself. Second, this is a way of leveling expected delays across team types and making extreme values of such delays for any specific team type less likely. The latter aspect was not the case in an earlier design of our robustness metrics, called $sumResDelay_{SIM(k)}$, $sumResDelay_k$ and $sumResDelay_{sum} = \sum_{k \in K} sumResDelay_k$, where delays were defined exactly the same way, but the sum of delays across all tasks carried out by teams of type k was computed instead of taking the maximum. The meaning of τ for these versions of the metrics would also be less clear, and more difficult to relate to the standard measures of punctuality used in the industry. Still, we will see in our numerical study section that the latter of these earlier metrics can provide meaningful information to the analyst—we will use $sumResDelay_{sum}$ in comparing our approach with the state-of-the-art approach (Padrón et al. 2016).

Beyond assessing robustness of generated schedules and plans, we defined two sets of additional performance metrics to assess the overall effectiveness of the SPs on timely aircraft departures across the apron. First, for all aircraft in a given replication, we calculate the delay (if any) on starting the push-back task, against the scheduled push-back start time coming from RCPSP results ($start_i$), and then take the average across all replications per aircraft. We call this $acDelay(RCPSP)$. A similar delay calculation is carried out against the STD rather than $start_i$. We call this metric $acDelay(STD)$. In terms of what constitutes a delay and what does not, the reader is referred to the discussion of $maxResDelay_k$ on Fig. 3.

4.2 Simulation-optimization approach

In this section, we describe our solution approach in detail, which falls under the broader category of simulation-optimization techniques (Fu et al. 2005; Figueira and Almada-Lobo 2014). Simulation-optimization approaches provide the required flexibility to model the stochastic nature of real operational problems. Our literature review showed that coupling simulation with metaheuristics search is an efficient method for obtaining near-optimal solutions for complex problems in uncertain environments, including both vehicle routing and project scheduling problems—hence the applicability, in principle, to our apron scheduling problem. As a result, solutions for scheduling handling teams with a more robust performance under variable conditions can be obtained.

Following this research direction, we propose a simheuristic approach, shown in Fig. 4. After solving the task scheduling problem, two nested simheuristic algorithms are executed, which we name $SimLNS^f$ and $SimLoop^f$, and are discussed further in Sect. 4.3. In the $SimLNS^f$ stage, the solution coming from the routing optimization is further improved for each team type, with the help of a targeted constraint feedback at every iteration of the LNS. This feedback is generated through evaluation of SIM(k) and focuses on improving the individual team types' robustness. The overarching $SimLoop^f$ stage is used to improve the overall robustness of the apron operations, also through a constraint feedback. This feedback is generated after evaluation of the apron simulation (SIM(AP)) and improves the robustness at the task scheduling stage for the whole apron, rather than for individual team types.

Hereby, we explain the details of our approach as outlined in Fig. 4. From top to bottom, we start by solving the sequence of two PSP/RCPSP formulated in Sect. 3.1, which produces strict time windows for the start of all turnaround tasks, covering all aircraft turnarounds and SPs for the planning horizon. Both problems are solved to optimality using solver-independent CP platforms and modeling techniques— e.g., Gök et al. (2020a) or Musliu et al. (2018).

Simheuristic algorithms $SimLNS^f$ and $SimLoop^f$ represent enhanced versions of the $SimLNS$ and $SimLoop$ algorithms first presented in Guimarans et al. (2015). The 'f' superscript in their names is to highlight the major difference in our versions. These are augmented with specific feedback mechanisms, building upon results from simulation runs, and automatically generating additional cuts to our optimization models. A second major difference is that the two algorithms are nested into one another, representing an 'inner' loop and an 'outer' loop, upon which the overall simulation-optimization procedure is built.

The overall integrated logic of the $SimLNS^f/SimLoop^f$ combination works as follows. On exit from the RCPSP, two main pieces of information are known: in addition to all task start times, resource levels (number of teams) per type (i.e., rc_k) are set. With that information, it is possible to look at optimally routing the available teams throughout the set tasks. This is the remit of $SimLNS^f$, which, following our assumptions discussed earlier in the paper, can be run independently for each team type (index k). The $SimLNS^f$ algorithm itself involves one or more iterations (index $iter_{in}$), with more iterations being run if either local robustness is not achieved, or additional options still exist to reorganize the known teams across the set tasks, hoping this will either yield local robustness (preferably) or at least improve on the best 'local solution' so far. The local solution for team type k , denoted as $(\hat{Z}_3, rt)_k$, originates at the bottom end of our three-step TSPTW/VRPTW optimization phase, with rt being the collection of routes for all teams of type k and \hat{Z}_3 the related total slack. All three optimization steps are also modeled and solved by solver-independent CP technology, with only Z_1 which can be possibly solved to optimality for instances of real-world size. However, in some cases, the given number of teams for type k might not be sufficient to find a feasible solution. When this happens, until a feasible solution is found, the number of teams are increased by one and model Z_1 is re-solved. These increases can be explained as follows. In the project scheduling stages, we use the median of the traveling times. The subsequent routing stages use the actual times, which may render the problem, for some team types, infeasible. Hence the need to add teams to the specific team type, one by one, until feasibility is reached. The result from the Z_3 step is further improved, due to its critical influence in operational robustness, by means of the immediately following steps, that are the metaheuristic search (LNS) as well as the resource type specific simulation (SIM(k)). Here lies another difference between our version and that previously presented in Guimarans et al. (2015). In our case, we run SIM(k) prior to the destroy/repair steps of LNS, while previously the order was reversed. One reason for this is that, following the provision of the

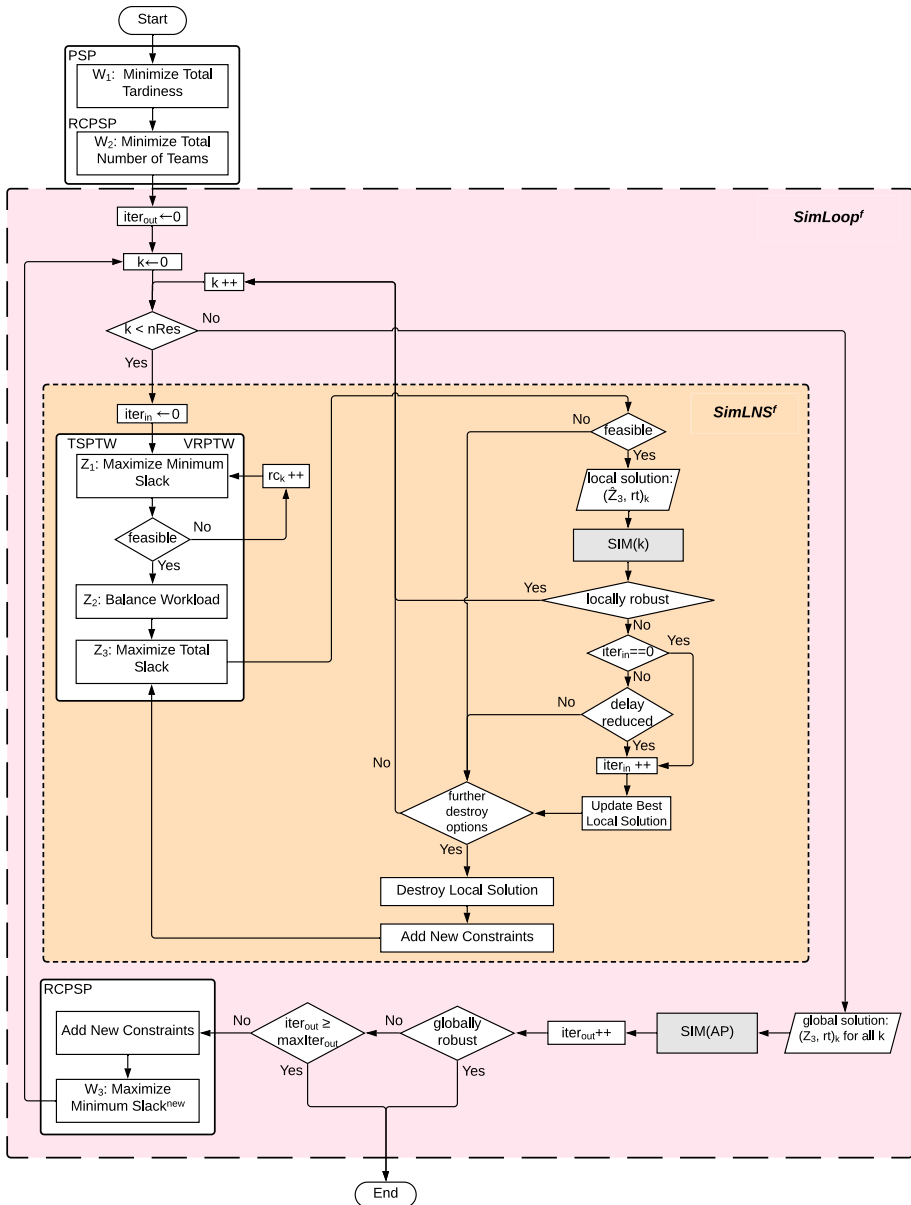


Fig. 4 Simulation-optimization for robust scheduling of apron operations

initial solution by Z_3 , running $SIM(k)$ on it allows the collection of performance measures that immediately can show whether local robustness holds true from the outset, for the given team type. At that point, prior to running the LNS step, we check whether at least the local solution that has just failed the robustness test has improved on the best solution so far by reducing the maximum delay. When this is the case, additional iterations of $SimLNS^f$ for the same team type will be needed ($iter_{in}++$), and we update our records. If there are still routes

available to be destroyed, we destroy the local solution by ‘unassigning’ certain teams from certain tasks (also called visits, following the routing jargon). Figure 5 in Sect. 4.3 will use an example to outline in detail how this is achieved. After that, we modify the formulation of problem Z_3 by adding new constraints that build directly upon our SIM(k) results, enforcing a higher total slack value between certain visits which cause the highest delay, with respect to the one allocated so far. This is also discussed in detail in Sect. 4.3. At this point, we run again Z_3 , the result of which is the production of a slightly modified local solution. This, in practice, implements the ‘repair’ step of our own version of LNS, by exploiting the exact same CP technology. The benefits of this, as opposed to running ad-hoc operators, are known from the CP literature (Guimarans et al. 2015). Running Z_3 under the additional constraints though may result in infeasibility, as there may not exist a solution that has a higher slack value for the given visits. In this case, SIM(k) and the robustness test will be bypassed, in favor of backtracking to the previous local solution and searching for the next, hopefully more promising, best option to destroy it.

After all team types have gone through $SimLNS^f$, all their local solutions are collected into what we call the global solution to our robust scheduling and routing apron operations problem. This kicks off one iteration of $SimLoop^f$, on entry of which we run SIM(AP), in essence simulating all tasks assigned to all teams of all types, for all aircraft. On completion of our DES replicated experiment, we move on to the next iteration $iter_{out}++$ and test global robustness of the schedules and routing plans just simulated, using the related robustness metrics. When global robustness is achieved, or a preset computational budget limit is reached (parameter $maxIter_{out}$), $SimLoop^f$ reaches its completion, and, with it, the overall simulation-optimization routine. Successive iterations of $SimLoop^f$ will need rerunning of $SimLNS^f$, again for all team types and following exactly the same procedure described above, but on different conditions. In fact, just prior to that, we add new constraints to the slightly modified RCPSP problem—now called W_3 instead of W_2 , which are based on the results from the SIM(AP) experiment, and also modify the objective function, now including additional slack from the project scheduling phase. The main difference in the constraints, instead, relates to adding one more team associated to the team type that is most likely to benefit from the addition. More details on this particular feedback will be discussed in Sect. 4.3. A comparison between $SimLoop^f$ and the $SimLoop$ algorithm from Guimarans et al. (2015) reveals that the former features a slightly different version of the destroy/repair and solution update steps, in that these are organized as part of the steps already discussed in relation to $SimLNS^f$.

We now devote the next section to the detailed discussion of our feedback mechanisms. On the opposite, the role of LNS in our approach is fairly standard and hence we omit lengthy discussions. The original source for LNS is Shaw (1998), while Gök et al. (2020a) provides an earlier account of the role of LNS in our solution approach, and the already cited work by Guimarans et al. (2015) provides for an example of how LNS can be employed in a problem different from the one we study.

4.3 Feedback mechanisms

So far, the main features of both $SimLNS^f$ and $SimLoop^f$ have been discussed, as well as the interactions between the two. We now show both algorithms in all details, underlining all the feedback mechanisms mentioned in the previous subsections and upon which robust scheduling of apron operations can be built. Table 1 introduces variables and parameters used for the first time in both Algorithms 1 and 2.

Table 1 Algorithm variables and parameters

Variables/parameters	Definition
$newConstraints$	Set of newly added constraints
$destroyOptions$	Gets true if further destroy options are available
$repSIM(\kappa)$	Current SIM(κ) replications
$maxRepSIM(\kappa)$	Total number of SIM(κ) replication
$outputSIM(\kappa)$	Outputs coming from SIM(κ)
D	Set of routes for which at least one visit is within a set time interval
$nRoute$	Number of routes to destroy besides the route with the most delayed task
$pairSet$	Set of all combinations of $nRoute$ routes from set D ; i.e., creating a set consisting of non-repeating pairs of routes
X	Set of randomly chosen pair of routes from $pairSet$
n	The route with the $maxResDelaySIM(\kappa)$
m	The visit with the $maxResDelaySIM(\kappa)$
o	The immediate successor of visit m
r	Variable used for moving backwards on the visits starting from m
$delay_r$	Delay at the start of visit r
κ	Delay threshold used for determining the visit with the root cause of the $maxResDelaySIM(\kappa)$
$listDelay$	Set of all visits whose next visits' delay are greater than κ that leads to m
$routeStack$	Sum of all $stack_i$ in $listDelay$
$repSIM(AP)$	Current apron simulation replications
$maxRepSIM(AP)$	Total number of apron simulation replications
$outputSIM(AP)$	Outputs coming from the apron simulation

Algorithm 1 starts with first solving model Z_1 . If the initial solution is not feasible, an additional team is added and the same model is solved again. After a feasible solution is found, the updated number of teams are outputted as rc'_k and the next two VRPTW/TSPTW models are solved sequentially in line 7. We get an initial solution for Z_3 — called \hat{Z}_3 , which is always feasible at this point. After several independent replications of SIM(k), while the solution is not robust and there are still further destroy operations left, the LNS begins first updating the current local solution as the best solution. At this stage, variable *routeSlack* is computed from line 30 to be used later on for the feedback constraint. After determining the visit with the maximum delay (m) and the route it belongs to (n), the routes whose visits fall into the defined time interval (30 minutes in our experiments) are added to set D (line 35). In order to prepare for the destroy operation, a combination of $nRoute$ routes in set D , which are the non-repeating pairs of routes that will be destroyed, are added to the set *pairSet*.

Leaving the update stage after line 39, if there are still further destroy options available, our LNS moves on destroying the visits of the most delayed route n and two other random routes of the same type chosen from *pairSet*. This set of routes is then removed from *pairSet*. At this point, a slightly modified version of the CP model for maximizing total slack is run, the result of which is the ‘repair’ of the routing plans for the team type, with all removed tasks reassigned to some team. The CP model is automatically modified by adding two new constraints: (1) updating the problem lower bound—inequality (24)—to the best objective value so far (\hat{Z}_3^*); and, (2) requesting that the total slack corresponding to the portion of tasks previously unassigned from the three selected visits (*listDelay*) to be higher than it was at the previous iteration (*routeSlack*)—inequality (25). If the solution is feasible, then SIM(k) is run and the same loop repeats as long as it is not robust and there are more destroy options left. Otherwise, the *SimLNS^f* ends, when there are no more destroy options left to improve the current solution. If the new solution is found to be robust, the values of rt^* , $slack^*$ and \hat{Z}_3^* are updated.

$$\hat{Z}_3 \geq \hat{Z}_3^* \quad (24)$$

$$\sum_{i \in listDelay} slack_i > routeSlack \quad (25)$$

Figure 5 provides a visual example of the generation of *listDelay* and what happens when the destroy operator is applied. Let us assume that Team 1 is the one with the highest delay, and Teams 2 and 3 are randomly selected among all other teams of the same type as Team 1. For each team, the sequence of labeled intervals represents the sequence of visits, with ‘AC x ’ denoting aircraft x . The visit of Team 1 to AC 53 is affected by the highest delay (3.9 minutes, well above the 3 minutes threshold). The algorithm analyzes the visits of Team 1 that precede AC 53, and determines how far back in the sequence of visits to extend. In the case shown, there appears to be a clear build-up of delays, culminating with the highest delay of AC 53. The backtracking assessment of delays stops with AC 20 because the delay on start of AC 20 is below the threshold parameter of $\kappa = 1$ minute— lower bound under which delays are considered either virtually negligible or easier to recoup. The sequence of AC 20, AC 35 and AC 46 becomes the *listDelay* variable in Algorithm 1. This list is used to automatically generate the constraint in Eq. (25). The destroy operator involves determining what tasks should be unassigned from Teams 1, 2 and 3. In the case shown, our algorithm unassigns (see the crosses in the figure) all tasks that lie fully within the time interval indicated. The time interval goes from 30 minutes prior to the planned start of the task related to AC 20 ($start_{20}$)—first task in *listDelay*—to 30 minutes after the planned end of the task related to

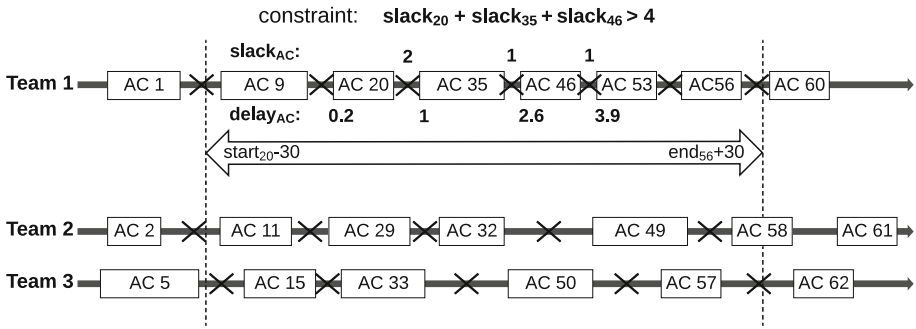


Fig. 5 *SimLNS^f* feedback illustration

AC 56 (end_{56})—task corresponding to the immediate successor of the highest delayed task of all.

At the start of *SimLoop^f* (Algorithm 2, line 9), for each $k \in K$, Algorithm 1 is run and all routing plans for all resource types, which are available at the end of *SimLNS^f*, are gathered. A SIM(AP) replicated experiment (of $maxRep_{SIM(AP)}$ replications) then produces estimates of global robustness, as well as the other performance metrics introduced above. When global robustness is not achieved or a preset computational budget limit is not reached (parameter $maxIter_{out}$), we run a modified RCPSP as follows. The first modification consists of a new objective function (1c) where the minimum value of a new variable $slack_i^{new}$ across all $i \in I$ is maximized. This variable represents a slack time which is added to the cumulative constraint (8c), over and above the *duration* and *setup* times. Notice that the RCPSP formulated in Sect. 3 did not include any slack in between tasks. This addition is meant to have beneficial effects in the subsequent VRPTW/TSPTW optimization stages.

$$W_3^* = \max W_3 = \max_{i \in I} (\min slack_i^{new}) \tag{1c}$$

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

Second, the model now forces to have an additional team of the type that maximizes the objective by adding constraint (26). We also set the lower bound for all team types to its previous value with constraint (27). In order to find new routing plans based on the new scheduling solution, Algorithm 1 is run once again followed by SIM(AP). The outer loop continues until either global robustness is achieved, or the maximum number of iterations $maxIter_{out}$ is reached.

$$\sum_{k \in K} rc_k = \sum_{k \in K} rc'_k + 1 \tag{26}$$

$$rc_k \geq rc'_k, \quad \forall k \in K \tag{27}$$

5 Experiments

In this section, we first discuss our instances (Sect. 5.1) and analyze the performance of our approach, both in its own right (Sect. 5.2) and by comparison (Sect. 5.3) against the closest competing algorithm from the literature (Padrón et al. 2016).

Algorithm 1: $SimLNS^f$

```

Input:  $start, duration, k$ 
Output:  $rt^*, slack^*, \hat{Z}_3^*, rc'_k$ 
1  $iter_{in} \leftarrow 0$ 
2 solve model  $Z_1$ 
3 while  $Z_1$  is infeasible do
4    $rc_k \leftarrow rc_k + 1$ 
5   solve model  $Z_1$ 
6  $rc'_k \leftarrow rc_k$ 
7 Solve models  $Z_2, Z_3$ 
8  $rt, stime, slack, \hat{Z}_3 \leftarrow output(Z_3)$ 
9  $newConstraints \leftarrow \emptyset$ 
10  $destroyOptions \leftarrow True$ 
11  $output_{SIM(k)} \leftarrow \emptyset$ 
12  $rep_{SIM(k)} \leftarrow 0$ 
13 while  $rep_{SIM(k)} < maxRep_{SIM(k)}$  do
14    $output_{SIM(k)} \leftarrow output_{SIM(k)} \cup SIM(k)$ 
15    $rep_{SIM(k)} \leftarrow rep_{SIM(k)} + 1$ 
16  $maxResDelay_{SIM(k)} \leftarrow max(average(output_{SIM(k)}))$ 
17  $maxResDelay^*_{SIM(k)} \leftarrow maxResDelay_{SIM(k)}$ 
18 while  $(maxResDelay_{SIM(k)} \geq \tau) \wedge (destroyOptions == True)$  do
19   if  $(Z_3$  is feasible)  $\wedge$ 
20      $((iter_{in} == 0) \vee (maxResDelay_{SIM(k)} < maxResDelay^*_{SIM(k)}))$  then
21      $iter_{in} \leftarrow iter_{in} + 1$ 
22      $rt^* \leftarrow rt, slack^* \leftarrow slack, \hat{Z}_3^* \leftarrow \hat{Z}_3$ 
23      $maxResDelay^*_{SIM(k)} \leftarrow maxResDelay_{SIM(k)}$ 
24      $X \leftarrow \emptyset$ 
25      $D \leftarrow \emptyset$ 
26      $m \leftarrow$  the visit with  $maxResDelay_{SIM(k)}$ 
27      $o \leftarrow$  the immediate successor of visit  $m$ 
28      $n \leftarrow$  route of  $maxResDelay_{SIM(k)}$ 
29      $r \leftarrow m$ 
30     repeat
31        $r \leftarrow$  immediate predecessor of  $r$ 
32       add  $r$  to  $listDelay$ 
33     until  $delay_r < \kappa$ ;
34      $routeSlack \leftarrow \sum_{i \in listDelay} slack_i$ 
35     for each visit  $j \in I_k$  do
36       while  $(stime_r - 30 < stime_j < stime_o + duration_o - duration_j + 30) \wedge$ 
37          $(rt_j^* \notin D)$  do
38         add  $rt_j^*$  to  $D$ 
39      $pairSet \leftarrow$  all combinations of  $nRoute$  routes from set  $D$ 
40     if  $pairSet == \emptyset$  then
41        $destroyOptions \leftarrow False$ 
42     else
43        $X \leftarrow Random(pairSet)$ 
44        $pairSet \leftarrow pairSet - \{X\}$ 
45       for each route pair  $p \in X \cup n$  do
46         for each visit  $j \in p$  do
47           if  $stime_r - 30 < stime_j < stime_o + duration_o - duration_j + 30$  then
48             Destroy  $j$ 
49       add constraints (24) and (25) to  $newConstraints$ 
50       Repair  $Z_3$  CP model with  $newConstraints$ 
51        $rt, stime, slack, \hat{Z}_3 \leftarrow output(Z_3)$ 
52       if  $Z_3$  is feasible then
53          $output_{SIM(k)} \leftarrow \emptyset$ 
54          $rep_{SIM(k)} \leftarrow 0$ 
55         while  $rep_{SIM(k)} < maxRep_{SIM(k)}$  do
56            $output_{SIM(k)} \leftarrow output_{SIM(k)} \cup SIM(k)$ 
57            $rep_{SIM(k)} \leftarrow rep_{SIM(k)} + 1$ 
58            $maxResDelay_{SIM(k)} \leftarrow max(average(output_{SIM(k)}))$ 
59 if  $maxResDelay_{SIM(k)} < \tau$  then
60    $rt^* \leftarrow rt, slack^* \leftarrow slack, \hat{Z}_3^* \leftarrow \hat{Z}_3$ 

```

Algorithm 2: *SimLoop*^f

Input: *start, duration*

```

1  $rt \leftarrow \emptyset$ 
2  $iter_{out} \leftarrow 0$ 
3 repeat
4   if  $iter_{out} \neq 0$  then
5      $newConstraints \leftarrow \emptyset$ 
6     add constraints (8c), (26) and (27) to  $newConstraints$ 
7     Solve  $W_3$  with objective (1c) and with  $W_2$  constraints  $\cup newConstraints$ 
8      $start \leftarrow output(W_3)$ 
9   for each team type  $k \in K$  do
10     $rt^*, slack^*, \hat{Z}_3^*, rc'_k \leftarrow \text{Algorithm 1}(start, duration, k)$ 
11     $rt \leftarrow rt \cup rt^*$ 
12  generate Petri net model
13   $rep_{SIM(AP)} \leftarrow 0$ 
14  while  $rep_{SIM(AP)} < maxRep_{SIM(AP)}$  do
15     $output_{SIM(AP)} \leftarrow output_{SIM(AP)} \cup SIM(AP)(rt)$ 
16     $rep_{SIM(AP)} ++$ 
17   $maxResDelay_k \leftarrow max(average(output_{SIM(AP)}))$ 
18   $maxResDelay_{max} \leftarrow \max_{\forall k \in K}(maxResDelay_k)$ 
19   $iter_{out} ++$ 
20 until  $(maxResDelay_{max} < \tau) \vee (iter_{out} \geq maxIter_{out})$ ;
```

5.1 Instance generation

Before running our experiments, we designed a combination of problem instances with the aim to cover as varied a portion of real-world airport operational settings as possible in the subsequent analysis. The planning horizon is set to 8 hours of operation, which represents between a third and half of the duration of a day of operation at most airports. We denote our instances as $\tau\alpha_beta_gamma_theta_sigma$.

Parameter α gives the total number of aircraft arrivals/turnarounds in the planning horizon. We investigated two levels for α , namely 100 and 250. Between them they cover, roughly, airport sizes of all but the top-20 commercial airports by yearly passenger traffic (60+ million passengers/year) (ACI 2020), out of about 10,000 commercial airports worldwide (according to IATA codes (IATA n.d.)).

Parameter β characterizes the profile of aircraft arrivals/turnarounds over the planning horizon. We investigated five levels for this factor: F, P, PP, FP and PF, where each 'P' represents a related 'peak' of operations in terms of number of turnarounds. See Fig. 6 for the example related to $\alpha = 250$.

Parameter γ gives the number of SPs operating at the airport's apron, which is largely a function of α , with values taken directly from information that is freely available on the web. For instance, in the case with about 250 turnarounds, the size of the airport is such that, on average, about five SPs normally operate in it, for each resource type/service. Each airline will normally use the services of several SPs, i.e., for airline 'a' the baggage loading/unloading and push-back is handled by SP#1, the catering could be contracted to SP#13, and potable water supply and toilet servicing to SP#27, while for airline 'b' it may be SP#2, SP#15 and SP#30, respectively. The number of SPs per service does vary considerably from airport to airport, in the real world.

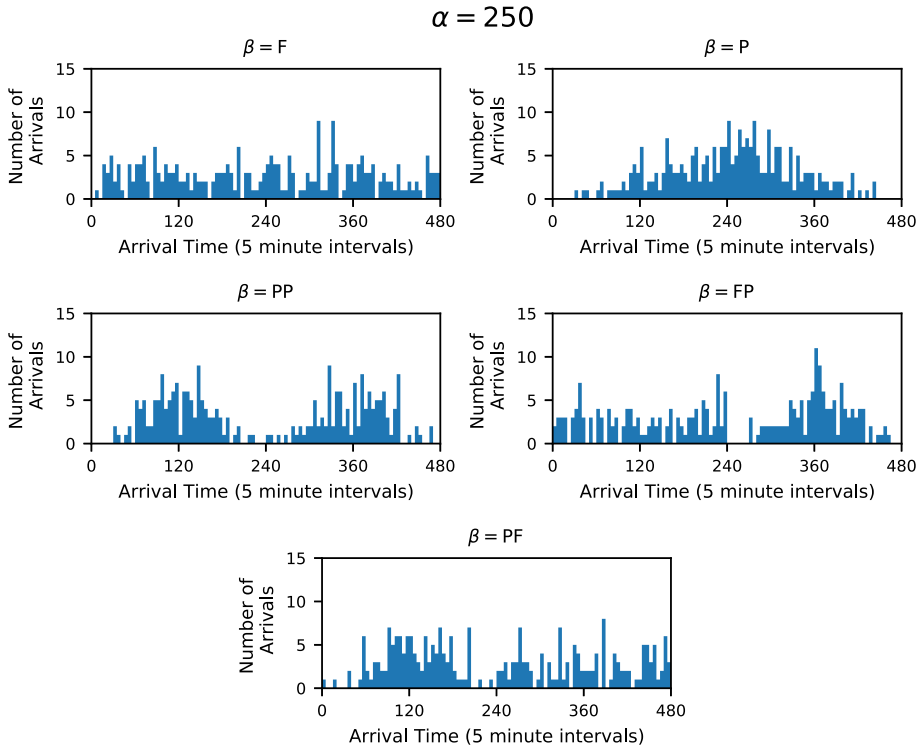


Fig. 6 Different aircraft arrival profiles across the planning horizon

Parameter θ states whether the distribution of turnarounds across the SPs is either even or not. In the case of ‘even’, the percentage of aircraft handled by the SPs are equal. Example distributions in ‘uneven’ cases may be: for 100 aircraft with two SPs, we consider a 20%–80% split; for 250 aircraft with five SPs, we have a 5%–10%–15%–30%–40% split.

Finally, parameter σ represents the overall variability in the considered probability distributions. Two levels were investigated: medium and high variability. For task duration and replenishment times, the upper limit of the used triangular distributions is set to 30% above the mode for medium variability, and 60% above for high variability. The lower limit is set to 20% below the mode for medium variability and 10% below for high variability. By keeping the lower bound of the parameters slightly higher in the high variability case compared to the medium case, we target higher delays on the high variability scenario. The mode is kept constant across the two variability scenarios. As for the traveling times, exponential distributions of lower and higher variability were adopted for the medium and high variability cases respectively. These times were generated, following exponential distributions of parameter λ , as $1/\lambda + \text{expo}(1/\lambda \cdot \text{variability})$, where $1/\lambda$ is the deterministic value from our case study, and *variability* is equal to 0.6 and 0.3 for the high and medium variability cases, respectively. Aircraft arrival times vary between 5 minutes before to 5 minutes after STA in all cases following a triangular distribution for which STA is the most likely value.

In total, 40 instances were tested. These resulted from the eight possible combinations of parameters α , θ and σ , each combined with the five template profiles we created.

Table 2 Values of algorithmic parameters

Parameters	Values
τ (minutes)	3
κ (minutes)	1
$maxRep_{SIM(\kappa)}$	200
$maxRep_{SIM(AP)}$	10
$maxIter_{out}$	15

5.2 Algorithmic performance

Table 2 summarizes the values of the most relevant algorithmic parameters used across the instances we investigated. SIM(AP) replications take longer than SIM(k) replications, hence the difference in the related parameter values.

Tables 3 and 4 provide an overview of the effectiveness of our approach. They summarize the likely operational performance of both the SPs and the apron overall, across all instances. The resource commitment for all instances are shown in Table 3, thus giving an idea of process efficiencies involved in achieving the related performance. The same table also shows for which instances overall apron robustness was achieved on completion of our overall algorithm ('R' and 'NR' standing for 'Robust' and 'Not Robust', respectively), as well as how many iterations of $SimLoop^f$ were required ($iter_{out}$). Table 4 shows the values for selected performance metrics at both the initial and final iterations of $SimLoop^f$. We provide 95% confidence intervals for $acDelay$ (RCPSP), whilst for $maxResDelay$ we show the values of $maxResDelay_{max}$ (column 'Max') as well as its 90th percentile across all team types (column 'P90'). We also provide the mean and the half-width of the difference between the final and the initial solution for $maxResDelay$.

Table 3 confirms that our feedback-enhanced simheuristic yields robust solutions in 80% of the cases, within the set limit of 15 iterations of the outer loop. The exceptions are related solely to the biggest apron size considered, and in eight out of the ten high-variability instances. Even then, our simheuristic managed to improve, often considerably, on the robustness metric. To see this, compare the values in the 'Max' columns in Table 4—see instance ta250_FP_5_UE_H. Overall, the values in column 'P90' for the final iteration of $SimLoop^f$ confirm that our approach also 'compresses' resource delays, often quite considerably: P90 reaches values that are in excess of half a minute lower than Max in more than 60% of the instances. Of all the 'non robust' cases, only instance ta250_PF_5_UE_H has a value of P90 above the set threshold, whilst in instance ta250_F_5_UE_H, the value of P90 falls below 2 minutes (against a value of Max in excess of 5 minutes).

The $acDelay$ (RCPSP) figures from Table 4 appear to highlight an overall *physiological* level of delay on starting push-back for all instances. On completion of the first iteration of $SimLoop^f$, $acDelay$ (RCPSP) is always within 3 minutes before or after the time originally set at the project scheduling stage, which means that overall apron operations tend to deliver to plan even after uncertainty is accounted for. The results seem to provide some reassurance that, even on the actual days of operations, where uncertainty levels will potentially reach new heights, the delay on starting push-back is still far below the industry standard threshold of 15 minutes (although this standard applies more directly to metric $acDelay$ (STD)).

The difference between the final and the initial delay on starting push-back is not substantial, which is explained by the characteristics of flight schedules. Turnarounds must be scheduled within extremely tight time windows imposed by airlines. Under realistic con-

Table 3 Total resource requirements towards achieving the robustness target

Instances	$iter_{out}$	Robustness	$\sum rc_k$		Δ
			Initial	Final	
ta100_F_2_E_H	2	R	75	77	2
ta100_F_2_UE_H	4	R	70	75	5
ta100_FP_2_E_H	1	R	87	87	0
ta100_FP_2_UE_H	5	R	82	90	8
ta100_P_2_E_H	10	R	92	105	13
ta100_P_2_UE_H	7	R	98	105	7
ta100_PF_2_E_H	1	R	84	84	0
ta100_PF_2_UE_H	5	R	90	97	7
ta100_PP_2_E_H	2	R	102	106	4
ta100_PP_2_UE_H	8	R	90	102	12
ta100_F_2_E_M	1	R	68	68	0
ta100_F_2_UE_M	3	R	64	67	3
ta100_FP_2_E_M	3	R	76	78	2
ta100_FP_2_UE_M	2	R	76	81	5
ta100_P_2_E_M	1	R	88	88	0
ta100_P_2_UE_M	4	R	84	90	6
ta100_PF_2_E_M	2	R	80	81	1
ta100_PF_2_UE_M	2	R	82	86	4
ta100_PP_2_E_M	2	R	98	99	1
ta100_PP_2_UE_M	6	R	81	89	8
ta250_F_5_E_H	6	R	202	217	15
ta250_F_5_UE_H	15	NR	188	209	21
ta250_FP_5_E_H	15	NR	203	226	23
ta250_FP_5_UE_H	15	NR	220	243	23
ta250_P_5_E_H	15	NR	231	253	22
ta250_P_5_UE_H	15	NR	240	264	24
ta250_PF_5_E_H	15	NR	222	242	20
ta250_PF_5_UE_H	15	NR	208	229	21
ta250_PP_5_E_H	12	R	258	280	22
ta250_PP_5_UE_H	15	NR	233	255	22
ta250_F_5_E_M	1	R	185	185	0
ta250_F_5_UE_M	5	R	169	178	9
ta250_FP_5_E_M	4	R	189	197	8
ta250_FP_5_UE_M	2	R	197	206	9
ta250_P_5_E_M	2	R	211	219	8
ta250_P_5_UE_M	13	R	214	234	20
ta250_PF_5_E_M	6	R	199	210	11
ta250_PF_5_UE_M	3	R	189	195	6
ta250_PP_5_E_M	1	R	234	234	0
ta250_PP_5_UE_M	1	R	205	205	0

Table 4 Selected performance metrics at the initial and final iterations

Instances	<i>max ResDelay</i> (min)			<i>acDelay</i> (RCPSP) (min)		
	Initial	Final	Final-Initial	Initial	Final	Final-Initial
	Max	Max	P90	Initial	Final	Final-Initial
ta100_F_2_E_H	6.09	2.58	2.29	1.94 ± 0.19	1.92 ± 0.19	-0.02 ± 0.07
ta100_F_2_UE_H	3.25	2.47	1.60	2.01 ± 0.19	1.78 ± 0.16	-0.23 ± 0.08
ta100_FP_2_E_H	1.88	1.88	1.64	1.79 ± 0.17	1.79 ± 0.17	0.00 ± 0.00
ta100_FP_2_UE_H	7.40	2.48	2.39	2.09 ± 0.26	2.05 ± 0.20	-0.03 ± 0.05
ta100_P_2_E_H	6.31	2.99	2.84	2.39 ± 0.27	2.19 ± 0.18	-0.2 ± 0.08
ta100_P_2_UE_H	3.55	2.70	2.36	1.89 ± 0.17	2.09 ± 0.16	0.21 ± 0.07
ta100_PF_2_E_H	2.56	2.56	1.86	1.83 ± 0.16	1.83 ± 0.16	0.00 ± 0.00
ta100_PF_2_UE_H	3.16	2.39	2.12	2.08 ± 0.19	1.99 ± 0.17	-0.09 ± 0.07
ta100_PP_2_E_H	8.83	2.97	2.51	2.10 ± 0.18	1.96 ± 0.18	-0.14 ± 0.08
ta100_PP_2_UE_H	7.34	2.61	1.15	1.98 ± 0.19	1.75 ± 0.16	-0.24 ± 0.05
ta100_F_2_E_M	1.79	1.79	1.49	1.27 ± 0.11	1.27 ± 0.11	0.00 ± 0.00
ta100_F_2_UE_M	3.76	2.81	1.91	1.39 ± 0.14	1.37 ± 0.14	-0.01 ± 0.06
ta100_FP_2_E_M	5.44	1.50	1.20	1.16 ± 0.11	1.10 ± 0.11	-0.06 ± 0.03
ta100_FP_2_UE_M	3.90	2.83	1.76	1.40 ± 0.17	1.33 ± 0.12	-0.08 ± 0.04
ta100_P_2_E_M	2.09	2.09	1.87	1.26 ± 0.12	1.26 ± 0.12	0.00 ± 0.00
ta100_P_2_UE_M	19.03	2.06	1.35	1.40 ± 0.13	1.41 ± 0.12	0.01 ± 0.04
ta100_PF_2_E_M	3.19	1.66	1.55	1.34 ± 0.14	1.24 ± 0.10	-0.09 ± 0.05
ta100_PF_2_UE_M	10.80	1.72	1.27	1.25 ± 0.13	1.21 ± 0.11	-0.04 ± 0.04
ta100_PP_2_E_M	3.62	1.97	1.59	1.36 ± 0.12	1.19 ± 0.12	-0.18 ± 0.04
ta100_PP_2_UE_M	12.54	2.72	1.74	1.31 ± 0.16	1.20 ± 0.12	-0.11 ± 0.04

Table 4 continued

Instances	<i>maxResDelay</i> (min)				<i>acDelay</i> (RCPSP) (min)			
	Initial		Final		Initial		Final	
	Max	P90	Max	P90	Initial	Final	Initial	Final
ta250_F_5_E_H	3.66	2.17	2.76	1.92	1.85 ± 0.11	1.85 ± 0.11	1.85 ± 0.11	1.85 ± 0.11
ta250_F_5_UE_H	11.02	3.40	5.22	1.89	2.07 ± 0.13	2.01 ± 0.12	2.01 ± 0.12	-0.06 ± 0.05
ta250_FP_5_E_H	4.96	3.09	4.35	2.71	2.09 ± 0.12	2.02 ± 0.12	2.02 ± 0.12	-0.06 ± 0.03
ta250_FP_5_UE_H	11.78	3.51	4.43	2.86	2.24 ± 0.15	2.23 ± 0.13	2.23 ± 0.13	-0.01 ± 0.05
ta250_P_5_E_H	5.41	3.58	3.61	2.98	2.02 ± 0.12	1.97 ± 0.12	1.97 ± 0.12	-0.05 ± 0.06
ta250_P_5_UE_H	8.36	4.39	3.84	2.00	2.17 ± 0.13	1.98 ± 0.12	1.98 ± 0.12	-0.19 ± 0.05
ta250_PF_5_E_H	8.87	2.46	4.36	2.04	2.00 ± 0.12	1.96 ± 0.13	1.96 ± 0.13	-0.04 ± 0.04
ta250_PF_5_UE_H	10.45	3.11	5.84	3.48	2.06 ± 0.12	2.04 ± 0.13	2.04 ± 0.13	-0.03 ± 0.04
ta250_PP_5_E_H	9.99	2.65	2.96	2.42	2.11 ± 0.12	2.00 ± 0.11	2.00 ± 0.11	-0.1 ± 0.04
ta250_PP_5_UE_H	4.25	3.42	3.42	2.82	2.21 ± 0.12	2.15 ± 0.12	2.15 ± 0.12	-0.07 ± 0.06
ta250_F_5_E_M	2.40	2.02	2.40	2.02	1.19 ± 0.07	1.19 ± 0.07	1.19 ± 0.07	
ta250_F_5_UE_M	12.87	3.11	2.22	1.65	1.31 ± 0.09	1.15 ± 0.07	1.15 ± 0.07	-0.15 ± 0.02
ta250_FP_5_E_M	3.42	1.97	2.28	1.88	1.32 ± 0.08	1.28 ± 0.07	1.28 ± 0.07	-0.04 ± 0.03
ta250_FP_5_UE_M	7.41	3.14	2.44	1.84	1.33 ± 0.09	1.31 ± 0.09	1.31 ± 0.09	-0.02 ± 0.03
ta250_P_5_E_M	4.21	2.36	2.83	1.97	1.30 ± 0.08	1.28 ± 0.08	1.28 ± 0.08	-0.02 ± 0.03
ta250_P_5_UE_M	9.19	3.49	2.73	1.70	1.38 ± 0.10	1.30 ± 0.07	1.30 ± 0.07	-0.08 ± 0.03
ta250_PF_5_E_M	6.61	1.46	2.90	2.44	1.26 ± 0.07	1.28 ± 0.08	1.28 ± 0.08	0.03 ± 0.03
ta250_PF_5_UE_M	3.30	2.04	2.89	1.82	1.31 ± 0.09	1.34 ± 0.08	1.34 ± 0.08	0.03 ± 0.02
ta250_PP_5_E_M	2.69	1.72	2.69	1.72	1.21 ± 0.08	1.21 ± 0.08	1.21 ± 0.08	
ta250_PP_5_UE_M	2.99	0.64	2.99	0.64	1.34 ± 0.08	1.34 ± 0.08	1.34 ± 0.08	

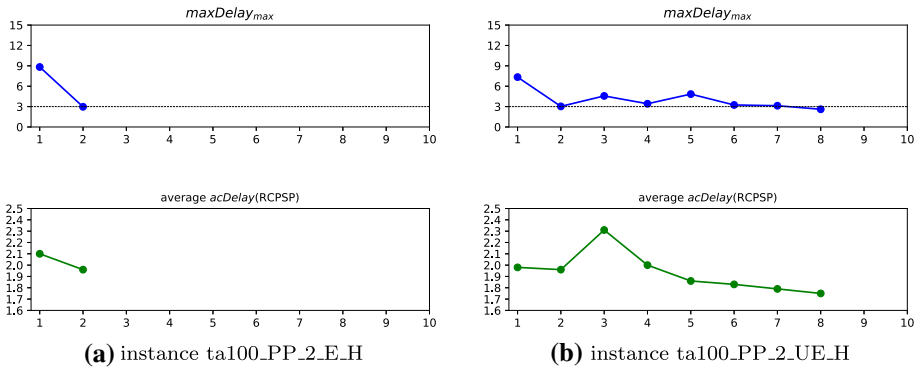


Fig. 7 Performance metrics of the instances with 100 aircraft, double peak aircraft arrival pattern and high variability. The horizontal axis is indexed according to the iterations of the outer loop

ditions, the actual duration of operations in the critical path is likely to extend beyond the planned completion time, regardless of task configuration or resource allocation. This situation results in a lower bound on $acDelay$, which is already achieved by the initial solution of our algorithm, and kept after the final step while increasing system predictability. The significant reduction in the tail end of resource delays after the overall completion of the algorithm makes the turnaround planning less sensitive to further disturbances, and hence more robust. Moreover, a more compact distribution of resource delays has a strong positive impact on the performance of service providers by mitigating their operational strain to comply with service level agreements.

Back to Table 3, one can assess the overall increase of resources (column Δ) needed to meet the robustness requirements. Let us now recall that, at each iteration of the outer loop (with the exception of the first iteration), a new team is added to the overall set—where it appears to more likely enhance operational robustness. In all cases where $\Delta > iter_{out}$, the positive difference between columns Δ and $iter_{out}$ is due to the addition of teams happening within the routing optimization stages, where a team is automatically added when the Z_1 problem becomes infeasible (more on this aspect to appear in Fig. 8). The number of resources shown in the table refer to teams, not individuals. Different turnaround services will then have different team sizes (number of individuals in a team), many of which will indeed have size one or two. The figures as shown, then, do not translate easily into an equivalent number of individuals. Having double-checked these figures with some SPs we have worked with recently, we found they are a realistic approximation of the number of individuals these SPs employ in similar situations. The fact that our numbers tend to be higher than theirs may help to explain another factor in the poorer on-time performance figures from real world apron settings.

We now turn to discussing the performance of our simheuristic algorithm *per se*. As an illustration, we focus on the two instances with 100 turnarounds, 2 SPs, and double peak of demand (PP), differing only by whether or not resources are distributed evenly. Figure 7 adds details to the results from Tables 3 and 4, and confirms the more general trend according to which unevenly distributed resources usually take more iterations of $SimLoop^f$ to hit the set robustness threshold.

Figure 8 shows more about the specific dynamics around resource increase from one iteration of the outer loop to the next. The differences between ‘at start’ and ‘at end’ represent resources added during the various iterations of the inner loop within the given iteration of

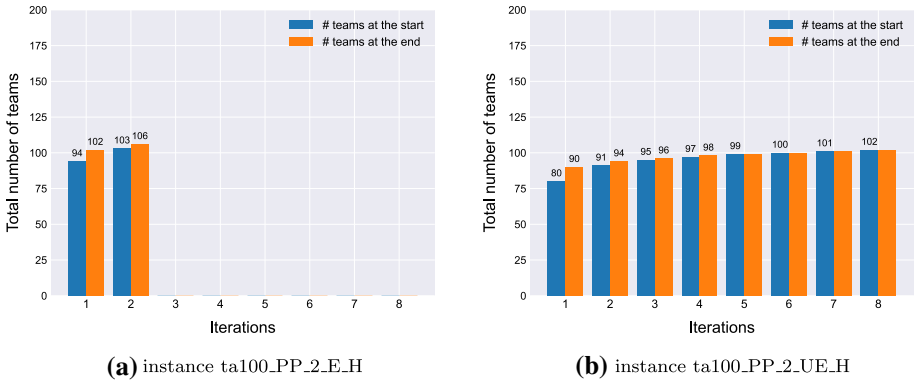


Fig. 8 Increase in the number of teams for the instances with 100 aircraft, double peak aircraft arrival pattern and high variability

Table 5 Changes in $maxResDelay_{SIM(k)}$ at each LNS iteration, for each global iteration, for unloading/loading teams in instance ta100_P_2_UE_H

$iter_{out}$	$iter_{in}$	$maxResDelay_{SIM(k)}$ (minutes)
1	1 → 2	4.04 → 2.62
2	1 → 2	3.51 → 2.67
3	1 → 2	3.64 → 2.86
4	1 → 2	3.11 → 2.87

the outer loop. Clearly, as more iterations of $SimLoop^f$ are needed, there will be less need to add resources because of infeasibility of the routing problems (of Z_1 , more precisely), and any additional resource addition will indeed be assigned to the team type that is most constraining to the overall apron robustness.

The local feedback mechanism operating at the routing level, combining LNS with $SIM(k)$, is triggered in a few but not all instances. One such instance is the peak aircraft arrival profile with high variability and unevenly distributed resources (instance ta100_P_2_UE_H). This was expected as with tighter aircraft arrivals it is harder to recover from any delays, considering the high number of visits each team has to make. In this case, the maximum delay of one of the team types (unloading/loading, more precisely) is 4.04 minutes, which is above the given threshold of 3 minutes. In order to reduce this delay below the threshold, the local feedback is triggered and new constraints are added when repairing the destroyed routes. The new solution’s maximum delay immediately drops below 3 minutes in a single LNS iteration. Table 5 provides all details about this chosen example. To be noticed that resource delay values in Table 5 should not be confused with resource delays in Table 4, as they are computed from two separate and quite different simulation studies ($SIM(k)$ for the former, $SIM(AP)$ for the latter).

Finally, a few considerations around the runtimes of our simheuristic. As an approach involving two separate simulation stages (particularly $SIM(AP)$ to mimic the entire apron dynamics), five optimization problems and a metaheuristic, altogether organized in two nested feedback loops, our approach was unlikely to ever become extremely fast. We ran all our experiments on Edinburgh University’s ECDF Linux Compute Cluster (Eddie) (ECDF 2021) which is 2.00GHz Intel(R) Xeon(R) Gold, CPU with 6138, with an allocated node consisting of 40 cores and 768GB RAM. Each instance was allocated a maximum of 24GB RAM to

be run in the same node simultaneously. At the end of the run, we have seen that instances required between 14.8GB and 15.2GB RAM. For the instances corresponding to the smaller apron size, runtimes ranged between 44 minutes and about 7 hours. For the bigger apron size instances, runtimes ranged between just over 3 hours to about 2 days and 12 hours. While there surely exists room for improvement, for instance by exploiting parallel computing features, we believe these runtimes demonstrate the applicability of our simheuristic. The tactical nature of our problem permits its solution to take place several days before the specific day of operation. Ideally however, given that airport flight timetables feature slight adjustments from day to day, it would be ideal for the problem to be solved during the evening or the night immediately before the day of operation, to exploit the latest timetable updates. Our results prove that for the smaller apron operation instances this may already be possible. The word ‘smaller’ shall not distract the reader here, as in those cases we are still talking about airport equivalents of around 20M passengers a year, which already represent sizeable realities in the airport industry. For the bigger apron size instances, our results also prove the applicability of the approach, with the caveat that until faster runtimes can be achieved, the problem may have to be solved about three days ahead of a given day of operation, i.e., based on a slightly different timetable. The resulting task schedules and resource routing plans will thus require specific adjustments on the day of operation, on a backbone solution that is already available and likely to be performing well.

5.3 Comparison with the state of the art

We have formed a comparison between the performance of our approach and that presented by Padrón et al. (2016), the closest approach to ours in terms of problem scope— see Sect. 2. However, their methodology had to be extended, for the purpose of the comparison, to cope with stochastic traveling and service times. More precisely, the weighted sum of tardiness cost, waiting, and traveling times (parameters a , b , and c in Padrón et al. (2016)), are now minimized by aiming to reach a balance between turnaround duration and resource usage. SIM(k) has been enclosed inside the optimization algorithm to decide at each step the ordering of the VRPTWs according to the $maxResDelay_k$ metric, where the most delayed task is scheduled first. Solutions are accepted if the maximum delay across all team types is below the given threshold—i.e., 3 minutes —, whilst improving the overall simulated waiting and travel times. A set of trade-off solutions also accounting for turnaround completion times are obtained, and the best solution in terms of the simulated cost is kept for comparison. In addition, the model in Padrón et al. (2016) has been also adapted to make a more accurate comparison. A minimum workload is imposed on each team to keep routes balanced, forcing the total service time of each route to be higher than 90% of the total average. Refilling is required in the case of capacitated resources servicing operations which consume supplies. Finally, the synchronization of resources is considered in the case of operations requiring more than one team to carry them out.

Both approaches have been tested for high and medium variability with 100 simulation replications over a set of five new instances, which in turn vary by aircraft arrival profile. The instances were generated in a similar way to those studied so far, but only a single SP is considered—Padrón et al. (2016) only caters for a single SP (hence the ‘E’ in the instance notation used here probably represents a slight abuse of notation). Each shift consists of 60 aircraft, which represents a real-world case with a single SP in a medium- to large-size airport.

Table 6 Comparison with Padrón et al. (2016)

Instances	<i>acDelay</i> (<i>STD</i>) (min)		Robustness		$\sum r_{ck}$	
	SIM ^e	simFeed	SIM ^e	simFeed	SIM ^e	simFeed
ta60_F_1_E_H	16.43 ± 4.25	15.96 ± 4.26	NR	R	42	43
ta60_FP_1_E_H	11.34 ± 3.44	11.07 ± 3.42	R	R	49	47
ta60_P_1_E_H	10.32 ± 3.31	10.02 ± 3.33	NR	R	55	54
ta60_PF_1_E_H	13.11 ± 3.88	12.95 ± 3.88	NR	R	49	48
ta60_PP_1_E_H	11.91 ± 3.69	11.47 ± 3.63	NR	R	55	56
ta60_F_1_E_M	11.28 ± 3.68	11.23 ± 3.68	R	R	37	38
ta60_FP_1_E_M	6.81 ± 2.90	6.63 ± 2.95	R	R	40	43
ta60_P_1_E_M	5.97 ± 2.82	5.77 ± 2.84	R	R	47	47
ta60_PF_1_E_M	8.65 ± 3.31	8.51 ± 3.33	R	R	41	42
ta60_PP_1_E_M	7.36 ± 3.09	7.37 ± 3.11	R	R	52	53

The results of our comparison are provided in Table 6 (where confidence intervals are 95%) and Figs. 9 and 10, where the approach presented in the present paper is referred to as Simheuristic with feedback (simFeed), while the Sequence Iterative Method (SIM^e) refers to the extension of Padrón et al. (2016) presented above. For SIM^e, the cost function parameters have been set to $a = 0.9$, $b = 0.05$, and $c = 0.05$, looking to mainly target the cost of tardiness, i.e., the primary optimization objective of our methodology. The travel time objective's relevance can be increased to reduce the number of required resources without affecting the SP robustness, since the stability of the solution is not compromised. Because this configuration can result in more delayed departures for turnarounds with minimal scheduled slacks, the solution with the shortest completion time from the obtained Pareto front can be privileged instead.

Under high variability, SIM^e could not reach robustness in four out of five instances. This proves the importance of more accurate feedback targeting critical assignments, instead of following an unconstrained search around neighborhoods. With respect to the *acDelay*(*STD*) metric, the two approaches fair similarly (with 95% confidence level), although ours leads to delays that are, on average, half a minute lower in most of the high variability instances. The total number of teams required to run operations are comparable across the two methods. However, in 80% of the medium-variability cases, SIM^e was able to find robust solution by deploying fewer teams. Last but not least, our approach has shown significant improvements in terms of *sumResDelay_{sum}* for all instances (see Fig. 9), as well as better values of *maxResDelay* for the push-back task in 80% of the instances (see Fig. 10). The latter two sets of results prove therefore that SPs would be able to run their operations under less 'stretched' conditions when adopting simFeed.

6 Conclusion

In this paper, we introduced a new constraint-based feedback approach, in the form of simheuristics, for solving the scheduling of airport turnaround tasks and routing of ground handling teams. In order to test the performance of our approach, we ran an extensive number of experiments representing real-world scenarios, and conducted a comparison with the state-of-the-art approach by Padrón et al. (2016) (extended version), which added an extra

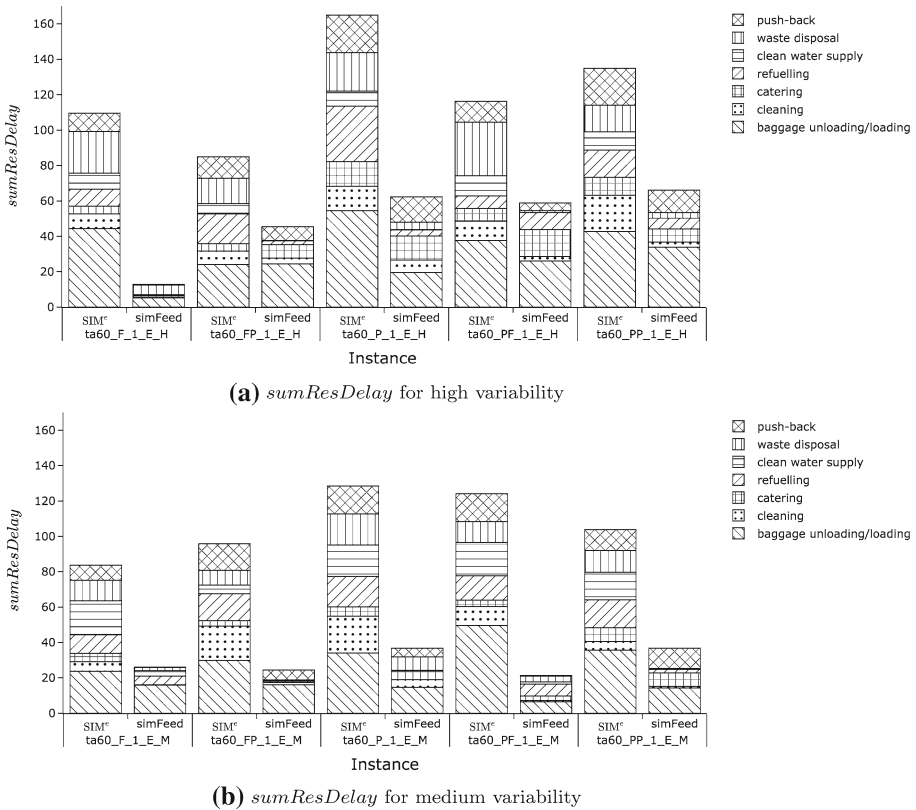


Fig. 9 $sumResDelay$ per team type

layer of confidence on our proposed solution technique. Although Padrón et al. (2016) provide a flexible method which is able to obtain a variety of solutions responding to conflicting priorities, our approach can be considered superior to the state of the art in two respects:

1. The introduction of specific feedback mechanisms that automatically modify the optimization models with additional cuts, which in turn are derived from two simulation processes and help achieve robustness;
2. Its feasibility in real-world airport domains with several SPs per type of service, as we have shown in our experiments, and following what the regulation dictates in all medium to large-size airports.

Contrasting with other existing simheuristic approaches, we proposed a new feedback mechanism from simulation to optimization, which consists of the generation of supplementary constraints that are added to the existing CP models at two different stages of the solution approach. The quick convergence to a robust solution through the inner loop, without compromising the additional resource costs, is one of the major strengths of our approach. This complements the performance of our optimization algorithms, in terms of the level of robustness for the initial solutions fed to the inner loop. In addition, the outer loop also manages to bring the solution to targeted levels of robustness within few iterations, by updating resource requirements and objective bounds in the cases where the inner loop is not enough to guarantee robustness for the apron overall.

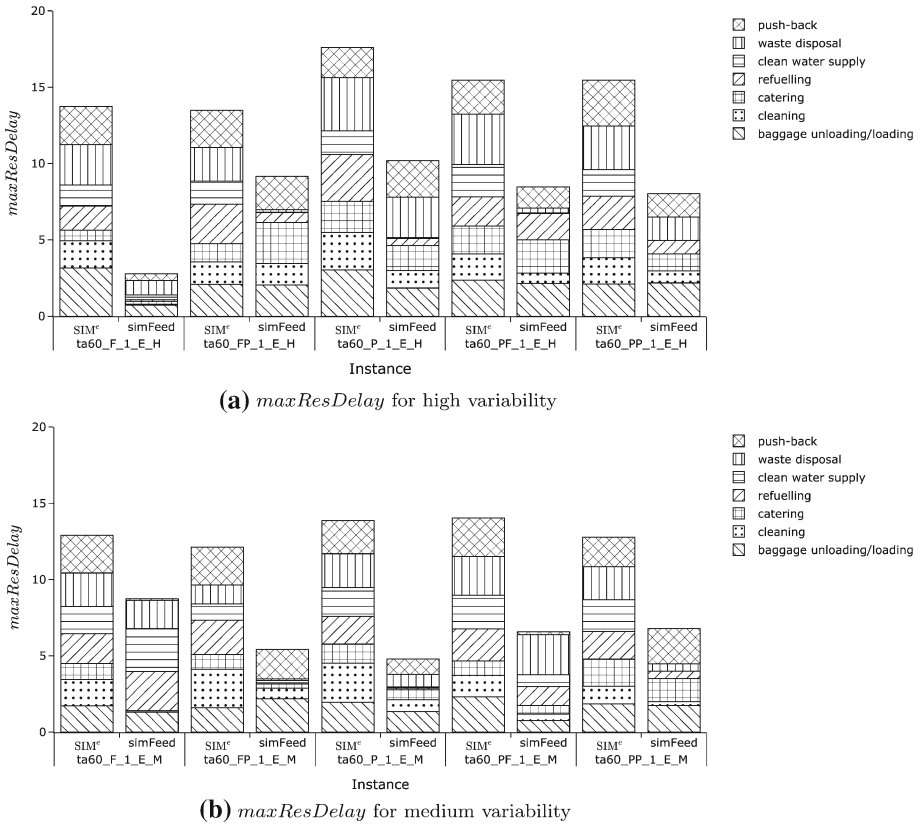


Fig. 10 *maxResDelay* per team type

Our work can be further improved in terms of computational efficiency in the CP-based optimization component. Exploiting parallelization of the constraint solving process in a cloud computing infrastructure, an active research topic (Menouer et al. 2019), has a great potential in large-scale optimization problems such as the one in this paper.

Recent developments at the crossroads between machine learning and combinatorial optimization also open new opportunities, particularly for complex and large-scale problems consisting of various technologies engineered to work together, as in our case. Incorporating machine learning in the solution approaches to such problems has potential for: (1) reducing the computational effort by learning promising solutions; and, (2) making the developed solution methodologies more adaptable to new domains by reducing the effort required for engineering (Bengio et al. 2021). Particularly for our problem, reinforcement learning can be used to further speed up the routing component (Bello et al. 2017) in our framework. Furthermore, it can be used for the online extension of our problem in which the solution developed in this paper would be used as the initial daily plan, while control and decision making need to be performed in real time (Kuhnle et al. 2019).

Acknowledgements This work has made use of the resources provided by the Edinburgh Compute and Data Facility (ECDF) (<http://www.ecdf.ed.ac.uk/>). We would especially like to thank Angela Chitzanidi for all her

help setting up the resources for us. The contributions of Daniel Guimaraes to this paper are not related to his role at Amazon.

Declarations

Conflict of interest All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- ACI (2020) Aci reveals top 20 airports for passenger traffic, cargo, and aircraft movements. <https://aci.aero/news/2020/05/19/aci-reveals-top-20-airports-for-passenger-traffic-cargo-and-aircraft-movements/>. Accessed: 22.07.2021.
- Ball, M., Barnhart, C., Nemhauser, G., & Odoni, A. (2007). Air transportation: Irregular operations and control. *Handbooks in operations research and management science*, 14, 1–67.
- Beldiceanu, N., Carlsson, M., & Rampon, JX. (2012). Global constraint catalog, 2nd edition (revision a). Tech. Rep. 2012:03. Computer Systems Laboratory.
- Bello, I., Pham, H., Le, QV., Norouzi, M., & Bengio, S. (2017). Neural combinatorial optimization with reinforcement learning. <https://openreview.net/pdf?id=Bk9mxISFx>
- Bengio, Y., Lodi, A., & Prouvost, A. (2021). Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research*, 290(2), 405–421. <https://doi.org/10.1016/j.ejor.2020.07.063>, <https://www.sciencedirect.com/science/article/pii/S0377221720306895>
- Blazewicz, J., Lenstraand, J., & Rinnooy Kan, A. (1983). Scheduling subject to resource constraints: Classification and complexity. *Discrete Applied Mathematics*, 5, 11–24.
- Bubalo, B., Schulte, F., & Voß, S. (2017). Reducing airport emissions with coordinated pushback processes: A case study. In: *Computational Logistics*, pp 572–586.
- Chica, M., Angel, A. J., Christopher, B., Oscar, C., & David, K. (2020). Why simheuristics? Benefits, limitations, and best practices when combining metaheuristics with simulation. *SORT-Statistics and Operations Research Transactions*, 44(2), 311–334. <https://doi.org/10.2436/20.8080.02.104>
- de Armas, J., Juan, A. A., Marquès, J. M., & Pedroso, J. P. (2017). Solving the deterministic and stochastic uncapacitated facility location problem: from a heuristic to a simheuristic. *Journal of the Operational Research Society*, 68(10), 1161–1176. <https://doi.org/10.1057/s41274-016-0155-6>
- ECDF (2021) Edinburgh compute and data facility. u of edinburgh. <https://www.ecdf.ed.ac.uk>. Accessed: 10-07-2021.
- Eurocontrol (2018) Airport collaborative decision making (A-CDM). <http://www.eurocontrol.int/articles/airport-collaborative-decision-making-cdm>
- Evler, J., Asadi, E., Preis, H., & Fricke, H. (2021). Airline ground operations: Optimal schedule recovery with uncertain arrival times. *Journal of Air Transport Management*, 92, 102021.
- Ferone, D., Gruler, A., Festa, P., & Juan, A. A. (2019). Enhancing and extending the classical grasp framework with biased randomisation and simulation. *Journal of the Operational Research Society*, 70(8), 1362–1375. <https://doi.org/10.1080/01605682.2018.1494527>
- Figueira, G., & Almada-Lobo, B. (2014). Hybrid simulation-optimization methods: A taxonomy and discussion. *Simulation Modelling Practice and Theory*, 46, 118–134.
- Fitouri-Trabelsi, S., Mora-Camino, F., Nunes-Cosenza, C. A., & Weigang, L. (2015). Integrated decision making for ground handling management. *Global Journal of Science Frontier Research: F (Mathematics and Decision Sciences)*, 15(1), 17–31.

- Fu, MC., Glover, FW., April, J. (2005). Simulation optimization: A review, new developments, and applications. In: *Proceedings of the Winter Simulation Conference*, 2005., IEEE.
- Gök, YS., Guimarans, D., Stuckey, P.J., Tomasella, M., & Ozturk, C. (2020a). Robust resource planning for aircraft ground operations. In: Hebrard E, Musliu N (eds) *Integration of constraint programming, artificial intelligence, and operations research*. Springer International Publishing, pp. 222–238.
- Gök, YS., Tomasella, M., Guimarans, D., & Ozturk, C. (2020b). A simheuristic approach for robust scheduling of airport turnaround teams. In: Bae KH, Feng B, Kim S, Lazarova-Molnar S, Zheng Z, Roeder T, Thiesing R (eds) *Proceedings of the 2020 Winter Simulation Conference*, IEEE, pp 1336–1347. <https://doi.org/10.1109/WSC48552.2020.9383947>
- Gonzalez-Martin, S., Juan, A. A., Riera, D., Elizondo, M. G., & Ramos, J. J. (2018). A simheuristic algorithm for solving the arc routing problem with stochastic demands. *Journal of Simulation*, 12(1), 53–66. <https://doi.org/10.1057/jos.2016.11>
- Grahn, R., & Jacquillat, A. (2020). Optimal escort dispatch for airport travelers with reduced mobility. *Transportation Research Part C: Emerging Technologies*, 111, 421–438. <https://doi.org/10.1016/j.trc.2019.12.010>
- Grasas, A., Juan, A. A., & Lourenço, H. R. (2016). SimILS: a simulation-based extension of the iterated local search metaheuristic for stochastic combinatorial optimization. *Journal of Simulation*, 10(1), 69–77. <https://doi.org/10.1057/jos.2014.25>.
- Gruler, A., Panadero, J., de Armas, J., Moreno Pérez, J. A., & Juan, A. A. (2018). Combining variable neighborhood search with simulation for the inventory routing problem with stochastic demands and stock-outs. *Computers and Industrial Engineering*, 123, 278–288. <https://doi.org/10.1016/j.cie.2018.06.036>
- Gruler, A., Panadero, J., de Armas, J., Pérez, J. A. M., & Juan, A. A. (2020). A variable neighborhood search simheuristic for the multiperiod inventory routing problem with stochastic demands. *International Transactions in Operational Research*, 27(1), 314–335. <https://doi.org/10.1111/itor.12540>
- Guimarans, D. (2021). petri_net_simulator: Simulation engine based on extended Petri nets. Retrieved July 22, 2021. From https://github.com/dguimarans/petri_net_simulator
- Guimarans, D., Arias, P., & Mujica Mota, M. (2015). *Large neighbourhood search and simulation for disruption management in the airline industry* (pp. 169–201). Cham: Springer.
- Guimarans, D., Dominguez, O., Panadero, J., & Juan, A. A. (2018). A simheuristic approach for the two-dimensional vehicle routing problem with stochastic travel times. *Simulation Modelling Practice and Theory*, 89, 1–14.
- Hentenryck, P. V. (2002). Constraint and integer programming in opl. *INFORMS J on Computing*, 14(4), 345–372.
- Hur, Y., Bard, J. F., Frey, M., & Kiermaier, F. (2019). A stochastic optimization approach to shift scheduling with breaks adjustments. *Computers and Operations Research*, 107, 127–139.
- IATA (n.d.) Airline and location code search. <https://www.iata.org/en/publications/directories/code-search/>. Accessed: 22.07.2021.
- Irawan, C. A., Eskandarpour, M., Ouelhadj, D., & Jones, D. (2021). Simulation-based optimisation for stochastic maintenance routing in an offshore wind farm. *European Journal of Operational Research*, 289(3), 912–926. <https://doi.org/10.1016/j.ejor.2019.08.032>
- Juan, A. A., Faulin, J., Grasman, S. E., Rabe, M., & Figueira, G. (2015). A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems. *Operations Research Perspectives*, 2, 62–72. <https://doi.org/10.1016/j.orp.2015.03.001>. <http://www.sciencedirect.com/science/article/pii/S221471601500007X>
- Juan, A. A., Keenan, P., Martí, R., McGarraghy, S., Panadero, J., Carroll, P., & Oliva, D. (2021). A review of the role of heuristics in stochastic optimisation: from metaheuristics to learnheuristics. *Annals of Operations Research*. <https://doi.org/10.1007/s10479-021-04142-9>. <https://doi.org/10.1007/s10479-021-04142-9>
- Kabongo, PC., Ferreira Ramos, TM., Leite, AF., Ralha, CG., & Weigang, L. (2016). A multi-agent planning model for airport ground handling management. In: *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pp 2354–2359. <https://doi.org/10.1109/itsc.2016.7795935>
- Keskin, M., Çatay, B., & Laporte, G. (2021). A simulation-based heuristic for the electric vehicle routing problem with time windows and stochastic waiting times at recharging stations. *Computers and Operations Research*, 125, 105060. <https://doi.org/10.1016/j.cor.2020.105060>
- Kuhnle, A., Schäfer, L., Stricker, N., & Lanza, G. (2019). Design, implementation and evaluation of reinforcement learning for an adaptive order dispatching in job shop manufacturing systems. *Procedia CIRP*. In: 52nd CIRP Conference on Manufacturing Systems (CMS), Ljubljana, Slovenia, June 12–14, 2019 81, 234–239. <https://doi.org/10.1016/j.procir.2019.03.041>, <https://www.sciencedirect.com/science/article/pii/S2212827119303464>

- Kuster, J., & Jannach, D. (2006). Handling airport ground processes based on resource-constrained project scheduling. *Advances in Applied Artificial Intelligence*, pp 166–176. https://doi.org/10.1007/11779568_20
- Latorre-Biel, J.I., Ferone, D., Juan, A.A., & Faulin, J. (2020). Combining simheuristics with petri nets for solving the stochastic vehicle routing problem with correlated demands. *Expert Systems with Applications*, p 114240. <https://doi.org/10.1016/j.eswa.2020.114240>
- Menouer, T., Sukhija, N., & Darmon, P. (2019). Towards a parallel constraint solver for cloud computing environments. In: *2019 IEEE Fifth International Conference on Big Data Computing Service and Applications (BigDataService)*, pp 195–198. <https://doi.org/10.1109/BigDataService.2019.00033>
- Musliu, N., Schutt, A., & Stuckey, P.J. (2018). Solver independent rotating workforce scheduling. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 10848 LNCS:429–445. https://doi.org/10.1007/978-3-319-93031-2_31
- Norin, A., Granberg, T.A., Värbrand, P., & Yuan, D. (2009). Integrating optimization and simulation to gain more efficient airport logistics. In: *Eighth USA/Europe Air Traffic Management Research and Development Seminar*.
- Official journal L 272, 25/10/1996, p 0036 - 0045 (1996) Council directive 96/67/ec of 15 october 1996 on access to the groundhandling market at community airports. <http://data.europa.eu/eli/dir/1996/67/oj>
- Padrón, S., & Guimarans, D. (2019). An improved method for scheduling aircraft ground handling operations from a global perspective. *Asia-Pacific Journal of Operational Research (APJOR)*, 36(04), 1–25. <https://doi.org/10.1142/S0217595919500209>
- Padrón, S., Guimarans, D., Ramos, J.J., & Fitouri-Trabelsi, S. (2016). A Bi-objective approach for scheduling ground-handling vehicles in airports. *Computers and Operations Research* 71(C):34–53. <https://doi.org/10.1016/j.cor.2015.12.010>
- Pagès-Bernaus, A., Ramalhinho, H., Juan, A. A., & Calvet, L. (2019). Designing e-commerce supply chains: A stochastic facility-location approach. *International Transactions in Operational Research*, 26(2), 507–528. <https://doi.org/10.1111/itor.12433>
- Pedrielli, G., Matta, A., Alfieri, A., & Zhang, M. (2018). Design and control of manufacturing systems: A discrete event optimisation methodology. *International Journal of Production Research*, 56(1–2), 543–564. <https://doi.org/10.1080/00207543.2017.1412532>
- Rabbani, M., Heidari, R., & Yazdanparast, R. (2019). A stochastic multi-period industrial hazardous waste location-routing problem: Integrating nsga-ii and monte carlo simulation. *European Journal of Operational Research*, 272(3), 945–961. <https://doi.org/10.1016/j.ejor.2018.07.024>
- Rabe, M., Deininger, M., & Juan, A. (2020). Speeding up computational times in simheuristics combining genetic algorithms with discrete-event simulation. *Simulation Modelling Practice and Theory*, 103, 102089. <https://doi.org/10.1016/j.simpat.2020.102089>
- Scala, P., Mota, M. M., Wu, C. L., & Delahaye, D. (2021). An optimization-simulation closed-loop feedback framework for modeling the airport capacity management problem under uncertainty. *Transportation Research Part C: Emerging Technologies*, 124, 102937. <https://doi.org/10.1016/j.trc.2020.102937>
- Schultz, M., Evler, J., Asadi, E., Preis, H., Fricke, H., & Wu, C. L. (2020). Future aircraft turnaround operations considering post-pandemic requirements. *Journal of Air Transport Management*, 89, 101886. <https://doi.org/10.1016/j.jairtraman.2020.101886>
- Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In: *International conference on principles and practice of constraint programming*, Springer, pp. 417–431.
- Solomon, M. M., & Desrosiers, J. (1988). Survey paper—time window constrained routing and scheduling problems. *Transportation Science*, 22(1), 1–13. <https://doi.org/10.1287/trsc.22.1.1>
- Tomasella, M., Clare, A., Gök, Y.S., Guimarans, D., & Ozturk, C. (2019). STTAR: a simheuristics-enabled scheme for multi-stakeholder coordination of aircraft turnaround operations. In: *2019 Winter Simulation Conference (WSC)*, IEEE, pp. 488–499.
- Wang, S., Che, Y., Zhao, H., & Lim, A. (2021). Accurate tracking, collision detection, and optimal scheduling of airport ground support equipment. *IEEE Internet of Things Journal*, 8(1), 572–584. <https://doi.org/10.1109/JIOT.2020.3004874>
- Zhang, M., Matta, A., Alfieri, A., & Pedrielli, G. (2018). Simulation-based benders cuts: A new cutting approach to approximately solve simulation-optimization problems. In: *2018 Winter Simulation Conference (WSC)*, pp. 2225–2236. <https://doi.org/10.1109/WSC.2018.8632326>
- Zhao, P., Han, X., & Wan, D. (2021). Evaluation of the airport ferry vehicle scheduling based on network maximum flow model. *Omega*, 99, 102178. <https://doi.org/10.1016/j.omega.2019.102178>