A NOVEL DECOMPOSITION APPROACH FOR HOLISTIC AIRLINE OPTIMIZATION

LUKAS GLOMB, FRAUKE LIERS, FLORIAN RÖSEL

All authors: FAU Erlangen-Nuremberg, Department of Data Science, Cauerstraße 11, 91058 Erlangen, Germany

Abstract. Airlines face many different planning processes until the day of operation. The assignment of aircraft to flights is of central importance and is determined by the optimization of the two planning problems Fleet Assignment and Tail Assignment. The competitiveness of the derived flight sequences strongly depends on the Turnaround Handling, which coordinates the processes on the ground between two flights. All of these planning problems have in common that they often need to be reoptimized on the day of execution due to unplanned events. In many cases, this is still done manually with the expertise of airline operators in the airline operations center. In order to automate this process and to be able to find the best possible reoptimization solution, the medium-term aircraft assignments and the short-term plannable turnaround processes should be optimized in a combined manner. For this purpose we provide a new mixed integer program, which considers Fleet Assignment, Tail Assignment and Turnaround Handling. Due to the size of the model, it cannot be solved efficiently as a complete model using available solvers. Hence, we develop a new decomposition algorithm that alternately solves the combined assignment problems and the turnaround model, and projects the turnaround costs into the aircraft assignment's objective function. In a computational study with realistic airline data, it is shown that for many problem instances this method yields feasible and optimal solutions much faster than comparable benchmark algorithms.

Keywords. OR in Airlines, Mixed Integer Programming, Decomposition, Tail Assignment, Turnaround Optimization

1. Introduction

Airlines face a multitude of different optimization tasks. The resulting optimization problems are usually divided into different mathematical models, which are solved in a cascading manner, since solving the combined optimization problems typically would require too many computational resources. Another reason for the separation of airline planning is that the corresponding decisions have to be done different amounts of time in advance. The Schedule Planning, which determines the set of offered flights, has to take place approximately six months in advance due to ticket sales, while e.g. flight sequences of individual aircraft are typically set a few weeks before the day of operation by the Tail Assignment.

Elementary strategic decisions start with the airline conception, which deals with aircraft procurement and flight network planning. Based on demand prognoses, it must be decided in the medium-term which routes must be served how frequently. This selection is linked to the decision which aircraft type executes which flight and how aircraft rotations can be adequately adapted to maintenance modalities.

The closer the day of execution comes, the more short-term and dynamic planning is required: One important aspect of this is the so-called turnaround, which is defined as the set of all processes on ground that an aircraft must undergo between two flights, like boarding or fueling. The processes must be executed in a specific order, using certain resources. The turnaround optimization coordinates the parallelization and acceleration of ground processes depending on turnaround resource availability. Since operational decisions are usually not included in long- and medium-term planning, airline operators constantly readjust plans at the day of

E-mail address: lukas.glomb@fau.de, frauke.liers@fau.de, florian.roesel@fau.de (corresponding author).

execution to ensure smooth operations and to recover from unforeseen disruptions. These readjustments incorporate medium- and short-term decisions. Hence, an integrated view on these problems enables the organization of re-planning tasks in an automated, coordinated and effective manner that typically will lead to better plans, when compared to solving them in a sequential fashion.

Our goal is hence to bring together the airline's medium- and short-term planning by optimizing aircraft assignments, including turnaround process planning to achieve more effective schedule recovery process.

Therefore our problem definition combines three optimization problems: First of all the *Fleet Assignment*, which deals with the assignment of aircraft types (so-called fleets) to individual flights. Together with the *Tail Assignment*, which refines the Fleet Assignment by assigning individual aircraft to the flights, we denote it as *Aircraft Assignment*. In addition there is the *Turnaround Handling*, which coordinates all processes that an aircraft goes through on the ground between landing and take-off.

Solutions obtained from an optimization model, which considers Aircraft Assignment and Turnaround Handling at the same time, are beneficial for the airline operations. For example, at peak hours at the airport these solutions would generally tend to assign aircraft which require fewer resources. Furthermore, flight delays can be compensated by re-planning the allocation of resources on ground like fueling or catering vehicles or re-assigning flights to less utilized aircraft.

Our Contribution. Hence, the contribution of this paper is the following: Firstly, a mixed-integer program is set up which combines the three optimization problems. To the best of our knowledge, such a model has not been presented in the literature before. It increases the solution quality compared to the sequential consideration of Fleet Assignment, Tail Assignment and Turnaround Handling. Due to its size and structure, it is not reasonable to solve it as one mixed integer problem using available solvers. Hence, a decomposition algorithm is developed in order to solve the model quickly and with a provable solution quality. It considers the Aircraft Assignment model as the master problem and the Turnaround Handling model as the sub problem. Based on the flight connection variables defined by the Aircraft Assignment model, the turnaround costs of the sub problem are calculated and then projected onto the variable space of the master problem. We prove that the algorithm is exact, i.e., it returns the optimal solution after a finite number of calculation steps. To demonstrate its practical relevance, the presented algorithm is implemented and benchmarked on instance sets based on a large German airline's schedule. The computational results are convincing, it is possible to solve a large proportion of the instances to optimality with the proposed approach, which has not been able with the benchmark approaches.

The remainder of this paper is organized as follows. In Section 2, we give an overview of relevant literature. In Section 3, the optimization model is presented. In Section 4, we introduce our decomposition algorithm and prove that it is correct. In Section 5, we present how instances for our computational study are generated from real-world flight plans and which benchmark algorithms are used. An extensive computational study on two different instance sets is performed in Section 6. Section 7 concludes with the main results of this paper and offers suggestions for future research directions.

2. Literature

The individual problem components are represented in numerous elementary forms in the literature: The Fleet Assignment problem was modeled based on a connection network with flights as nodes and flight connections as directed arcs in Abara (1989). The authors of Hane et al. (1995) describe an aggregation to the considered "time-space-network". They propose to solve the Fleet Assignment first and assign the tails based on this solution. Both Fleet Assignment and Tail Assignment are often solved with Column Generation, which solves pricing problems to obtain sequences of flights, called flight strings as stated in Desaulniers et al. (1997) or in Barnhart et al. (1998). This approach is effective in large networks with high connectivity. Up until

today, the field's literature already deals with the integration of dependent optimization problems, such as schedule planning [Sherali et al. (2013), Yan et al. (2020)], often with maintenance planning Gopalan and Talluri (1998), Sarac et al. (2006), and likewise with the crew assignment problem in all its variations, e.g., Mercier et al. (2005), Dunbar et al. (2012), Parmentier and Meunier (2020). These model combinations have been exhaustively discussed in the literature.

Since the considered problems result in large-scale mathematical problems, many publications discuss specific solution procedures which speed up the solution process: In Cordeau et al. (2001) Benders Decomposition (based on Benders (1962)) is used to solve the aircraft routing problem combined with the task of scheduling crews to flights. Therefore the decomposition approach iterates between a master problem that optimizes the aircraft routing, and a sub problem that determines the crew pairing. Both problems are solved using column generation and a heuristic branch-and-bound method ensures that integer solutions are obtained. Mercier et al. (2005) studied the very same problem varying the set up of Benders Decomposition: One uses the aircraft routing problem as the master problem, one in turn the crew scheduling problem. In addition to Benders Decomposition, Sandhu and Klabjan (2007) also use a solution approach which combines Lagrangian relaxation and column generation in order to solve the integrated airline fleeting and crew pairing problem. In Zeighami et al. (2020) there has been used an alternating Lagrange Decomposition approach which passes information between a master problem and a sub problem to solve a crew assignment problem. A time-based decomposition approach is introduced in Glomb et al. (2020): There a rolling-horizon approach is applied, which decomposes the problem in time intervals and solve them one after another while considering information of subsequent time-steps.

In this paper a decomposition approach is used in order to solve the proposed model. It is similar to Benders Decomposition, based on the procedure stated in Laporte and Louveaux (1993): The master problem is chosen to be a binary program in order to be able to use lower bound functions to map the costs of the sub problem into the master problem. The approach can also be considered as an implementation of the Logic Based Benders Decomposition (LBBD) as described in Hooker and Ottosson (2003). LBBD determines functions mapping from the domain of master variables to the real numbers for several solutions of the master problem. The functions are supposed to be under estimators for the costs of the sub problem emerging when the master variables are fixed to the argument of the function. It is claimed that the approach is working better the better the under estimators reflect the true objective value of the sub problem. The reason for choosing this approach is discussed in more detail in Section 4.1.

However, all model variants mentioned so far have in common that they combine the classical long-term planning problems. They hardly take into account airline operations and approximated operational structures. Instead of modeling operational aspects as ground times or trajectories precisely, ground times and flying times are incorporated in an aggregated way, often as constant mean values or expectations. Especially in the recovery process, i.e., the reaction to unforeseen disruptions, it is easier to shortly adapt aircraft assignment and ground schedules than crew assignments. The reasons are that it is required by law that the crew duties do not exceed certain bounds. It is allowed to reassign certain flights to crews, but it is possible to loose the crews' goodwill if this happens too often. The aircraft, on the other hand, do not care if they are rescheduled in the short term and very frequently. See also Stojković and Soumis (2001) for a more precise view on the constraints of crew rescheduling.

Hence it makes perfectly sense to consider the crew assignment as given and to optimize aircraft assignment and ground schedules jointly. Nevertheless, turnaround optimization is described in the literature separately from the aircraft assignment models. In Wu and Caves (2004) the effectiveness of on-ground buffer times was evaluated and proven. Evler et al. (2018) introduced a multiple turnaround dependency model to investigate

the time and resource dependencies on ground. This implies a dependency of the steering of ground processes at airline-specific center airports on the overall plan of the airport.

Hence, despite the practical relevance, to the best of our knowledge, turnaround interdependencies are not taken into account in current recovery models. We are currently aware of one paper that solves the Tail Assignment model while more accurately representing turnaround processes. Eltoukhy et al. (2019) model a robust version of the Tail Assignment model with maintenance planning and use the turnaround time reduction approach. This approach allows the stepwise reduction of turnaround times, but the whole turnaround process is considered aggregatedly. Furthermore the resource scheduling at airports is not modeled exactly. Hence, our work is an extension, since we model the resource flow exactly and present an algorithm which solves the emerging problems to global optimality.

We proceed with the description of this optimization model in Section 3. First we introduce the Aircraft Assignment model, then, based on this, the model for the ground process steering.

3. Modeling

In this section the main model which consists of Aircraft Assignment and Turnaround Handling is presented. Based on a given flight schedule, it is capable of assigning aircraft to flights and steering turnaround processes optimally, regarding process resource dependencies at airports. We develop the model in three steps. In Section 3.1, the Aircraft Assignment Model is introduced, which models the assignment of fleets and individual aircraft to flights. In Section 3.2, this model is extended by the Turnaround Process Scheduling, which models the individual process steps on each flight connection as a function of time and can thus model flight delays. In Section 3.3, the control of the resources required for the ground processes at airports is modeled. This establishes dependencies between turnarounds of different flight strings. In each part of the model construction we first discuss the necessary sets and parameters, introduce model variables and then explain the constraints.

3.1. Aircraft Assignment: Fleet/Tail Assignment.

The goal of the Aircraft Assignment is to assign aircraft of different aircraft types (so-called fleets) to the flights of a given schedule. The model is based on underlying graphs, which are called *connection networks*, as shown in Figure 1. These were firstly introduced in the context of Tail Assignment by Abara (1989). It is common to cluster individual aircraft with identical or very similar technical attributes to aircraft fleets. Nevertheless, it is necessary to know the individual flight strings of each single aircraft in order to identify the flight connections on which turnaround processes can take place. Hence, we avoid using a time-space network (presented in Hane et al. (1995)) because it does not necessarily determine one-to-one flight strings per aircraft if it is not set up for each individual aircraft. Furthermore, a connection-based approach is advantageous for the integration of connection-specific turnaround processes. We have chosen an acyclic formulation because the operational turnaround processes are planned on a daily basis and the overall model therefore does not cover a typical cyclic period of one week. The graph contains flight nodes representing all flights which can be served by the corresponding fleet and start nodes for each aircraft of this fleet. Every aircraft (start) node is connected to all possible flight nodes corresponding to a flight which can be the aircraft's first flight. All flight nodes are interconnected as long as they can be executed consecutively with respect to location and time. We give an overview of the input sets, parameters and model variables in Table 1.

L denotes the set of different flight legs and K the set of fleets / aircraft types. Since not every aircraft type can be assigned to each flight due to range restrictions or low seat capacity, the set L_k describes the set of flights that can be executed by $k \in K$. Conversely, K_l is defined as the set of fleets capable of executing $l \in L$. A_k^{con} is the set of allowed flight connections for fleet $k \in K$. Each individual aircraft $q \in Q_k$ in the model has its own start airport s_q. The set of start connections between aircraft start nodes q and flights is described by

Set	**Description**
S	Set of airports (stations)
L	Set of flight legs
K	Set of aircraft types (fleets)
Q_k	Set of individual aircraft of fleet k
L_k	Set of flight legs flyable by fleet k
K_l	Set of fleets capable to fly flight leg l
A_k^{con}	Set of flight connections of fleet k: $(i,j) \in A_k^{\text{con}} \Leftrightarrow i,j \in L_k, s_i^{arr} = s_j^{dep}, \overline{\epsilon}_i \leq \overline{\varsigma}_j$
A_k^{start}	Set of start connections of fleet k: $= (q,j)$ for all $q \in Q_k : j \in L_k : s_q = s_j^{dep}$
A_k	Union of connection arc sets: $A_k^{\text{start}} \cup A_k^{\text{con}}$
Parameter	**Description**
$c_{k,l} \geq 0$	Assignment costs of fleet k to flight leg l
$s_q \in S$	Start airport of aircraft q
$s_l^{\text{dep}}, s_l^{\text{arr}} \in S$	Departure, Arrival airport of flight leg l
$\overline{\varsigma}_l, \overline{\epsilon}_l \geq 0$	Scheduled time of departure, arrival of flight leg l
Variable	**Description**
$x_{k,(i,j)}$	Binary variable indicating if an aircraft of fleet k is using connection (i,j)
$y_{k,l}$	Binary variable indicating if flight leg l is assigned to fleet k

TABLE 1. Sets, parameters and variables for the Aircraft Assignment model part

A_k^{start}. The costs of the specific fleet-flight assignment of fleet $k \in K$ to flight leg $l \in L_k$ are denoted with $c_{k,l}$.

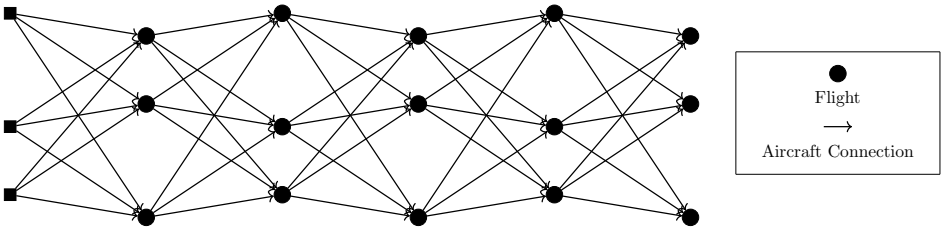

FIGURE 1. Connection network per fleet with flight nodes and ground edges

The parameters s_l^{dep} / s_l^{arr} denote the departure / arrival airport of flight leg l. The parameters $\overline{\varsigma}_l$ / $\overline{\epsilon}_l$ denote the scheduled departure / arrival time of flight leg l.

Binary variables $y_{k,l}$ become 1 if flight l is served by fleet k and 0 otherwise. Binary variables $x_{k,(i,j)}$ become 1 if an aircraft of the fleet k is assigned to a connection (i,j) and 0 otherwise. The Aircraft Assignment can thus be modeled as follows:

$$\text{(1a)} \qquad \min \quad \sum_{k \in K} \sum_{l \in L_k} c_{k,l} y_{k,l}$$

$$\text{(1b)} \qquad \text{s.t.} \quad \sum_{h:(h,l) \in A_k} x_{k,(h,l)} = y_{k,l} \qquad \forall l \in L_k, k \in K$$

$$\text{(1c)} \qquad \sum_{j:(l,j) \in A_k} x_{k,(l,j)} \leq y_{k,l} \qquad \forall l \in L_k, k \in K$$

$$\text{(1d)} \qquad \sum_{k \in K} y_{k,l} = 1 \qquad \forall l \in L$$

$$\text{(1e)} \qquad \sum_{l:(q,l) \in A_k^{\text{start}}} x_{k,(q,l)} \leq 1 \qquad \forall q \in Q_k, k \in K$$

$$\text{(1f)} \qquad x, y \quad \text{binary}$$

The objective function (1a) minimizes the assignment costs of fleet-flight combinations. Constraints (1b) and (1c) ensure the flow conservation on each flight, i.e., that every flight has exactly one preceding and at most one succeeding event. Preceding events can either be flights or starts, succeeding events only include flights. Constraints (1d) make sure that each flight leg is served by exactly one aircraft type. Constraint (1e) states that the flow out of aircraft nodes is at most 1, i.e., each aircraft is used at most once and has to start at the airport where it is at the beginning.

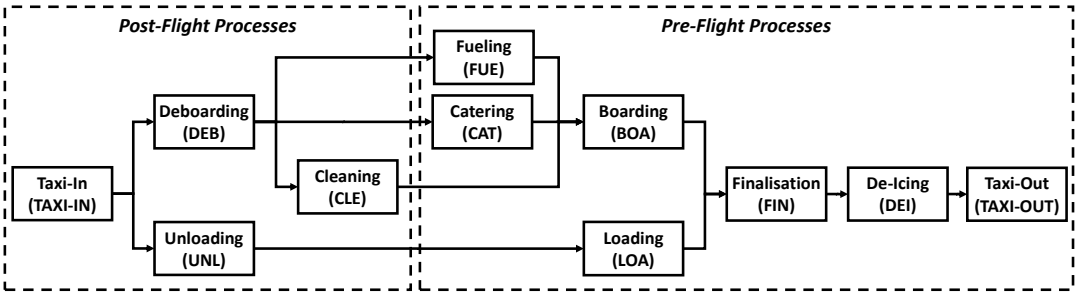

FIGURE 2. Schematic representation of the chronological sequence of turnaround process steps

Set	**Description**
S	Set of airports
P	Set of all turnaround steps $P^{\text{post}} \cup P^{\text{pre}}$
P^{post}	Set of post-flight turnaround steps (2): {TAXI-IN, DEB, UNL, CLE}
P^{pre}	Set of pre-flight turnaround steps (2): {FUE, CAT, BOA, LOA, FIN, DEI, TAXI-OUT}
$\tilde{P}$	$= \{(p_1, p_2) \mid p_1, p_2 \in P^{\text{post}}$ or $p_1, p_2 \in P^{\text{pre}}$ with (p_1, p_2) in Figure 2$\}$
$\overline{P}$	$= \{(p_1, p_2) \mid p_1 \in P^{\text{post}}, p_2 \in P^{\text{pre}}$ with (p_1, p_2) in Figure 2$\}$
Parameter	**Description**
$c^{\text{delay}}_{k,l} \geq 0$	Costs per delay minute on flight leg l when assigned to fleet k
$c^{\text{OPT}}_{k,l,p} \geq 0$	Option costs per turnaround process p of flight l when assigned to fleet k
$\overline{d_{k,l}} \geq 0$	Duration of flight l with fleet k
$\overline{d_{k,l,p}} \geq 0$	Default duration of process p of flight l when assigned to fleet k
$d^{\text{OPT}}_{k,l,p} \geq 0$	Possible duration reduction for process p of flight l when assigned to fleet k
$\overline{\varsigma}_l, \overline{\epsilon}_l \geq 0$	Scheduled time of departure, arrival of flight leg l
Variable from (1)	**Description**
$x_{k,(i,j)}$	Binary variable indicating if an aircraft of fleet k is using connection (i, j)
$y_{k,l}$	Determining fleet-flight assignments. Equals 1 if flight l is served by fleet k
Variable	**Description**
$\varsigma_{k,l}, \epsilon_{k,l}$	Take-off time, Landing time of flight l executed by fleet k
$\tau_{k,l}$	Delay of flight leg l executed by fleet k
$\varsigma_{k,l,p}, \delta_{k,l,p}, \epsilon_{k,l,p}$	Start time, duration, end time of turnaround process p of flight l executed by fleet k
$\psi_{k,l,p}$	Binary variable indicating if process p of flight l executed by fleet k is reduced

TABLE 2. Sets, parameters and variables for the turnaround schedule on each connection model

3.2. **Process Scheduling for isolated Turnaround Processes.** Each aircraft has to go through a variety of processes on ground between arrival and departure. This process sequence is known as the turnaround and summarized in Figure 2. It shows how the set of process steps is divided into pre-flight and post-flight processes and clarifies the notation. If a process has to be finished before another one can start, they are connected with an arc.

Turnaround planners have to make decisions on the characteristics of individual turnaround process steps, as well as the allocation of the necessary turnaround resources on the ground. Hence, the previously set up model must be linked to the ground processes of the respective assigned aircraft. At first we neglect the allocation of resources and deal with the timing of the turnaround processes. Therefore, we will first explain the additionally introduced sets, parameters and variables, which are listed in Table 2. As introduced before, let S be the set of airports. The process set P contains turnaround processes like boarding or catering. P can be partitioned into processes that take place before the flight (P^{pre}) and processes that take place after the flight (P^{post}). The pre-flight turnaround includes, for example, the boarding process, as this step directly depends on the number of passengers transported by the following flight. Tuples of two *successive* process steps are created in the sets $\tilde{P}$ and $\overline{P}$. $\overline{P}$ covers all tuples in which the first process is a post-flight process and the second process is a pre-flight process, whereas $\tilde{P}$ includes all other possible process tuples. The ground model captures two types of costs: $c^{\text{delay}}_{k,l}$ represents the cost of one delay minute while $c^{\text{OPT}}_{k,l,p}$ quantifies the costs of individual turnaround process accelerations for process p of flight l executed by fleet k. The time which is required to fly flight leg l with fleet k is $\overline{d_{k,l}}$. For all turnaround processes there exists a default time $\overline{d_{k,l,p}}$, which can be shortened by $d^{\text{OPT}}_{k,l,p}$ when accelerating / simplifying the process. Here, it is common

practice that processes such as taxiing, boarding or deboarding are speeded up through the choice of closer / better-equipped gates, giving the aircraft less taxiing time and allowing passengers to get off through both aircraft doors. Processes such as catering and cleaning can be reduced to a minimum, e.g., only standard catering is provided, only trash is collected but no carpet cleaning is done. The calculation of the delay is based on the scheduled time of arrival $\bar{\epsilon}_l$ per flight l. To represent these time-related aspects in the model, there exists a start time variable $\varsigma_{k,l}$ and a landing time variable $\epsilon_{k,l}$ for each flight l executed by fleet k. The turnaround processes per flight l and fleet k are characterized by a variable start time $\varsigma_{k,l,p}$, duration $\delta_{k,l,p}$ and end time $\epsilon_{k,l,p}$ as well. This duration can be controlled with the optional acceleration steered by the binary variable $\psi_{k,l,p}$. The deviation of the scheduled time of arrival $\bar{\epsilon}_l$ for flight l from the actual arrival time $\epsilon_{k,l}$ is recorded with the variable $\tau_{k,l}$. Taken together, the Aircraft Assignment Model (1) can be extended by the process scheduling on ground as described in Model 2

$$\text{(2a)} \quad \min \quad \text{(1a)} + \sum_{k\in K}\sum_{l\in L_k}\left(c^{\text{delay}}_{k,l}\tau_{k,l} + \sum_{p\in P} c^{\text{OPT}}_{k,l,p}\psi_{k,l,p}\right)$$

$$\text{s.t.} \quad \text{(1b)}-\text{(1f)}$$

$$\text{(2b)} \quad \varsigma_{k,l} + \overline{d_{k,l}}y_{k,l} = \epsilon_{k,l} \qquad \forall l \in L_k, k \in K$$

$$\text{(2c)} \quad \bar{\varsigma}_l y_{k,l} \leq \varsigma_{k,l} \qquad \forall l \in L_k, k \in K$$

$$\text{(2d)} \quad \epsilon_{k,l} \leq \bar{\epsilon}_l + \tau_{k,l} \qquad \forall l \in L_k, k \in K$$

$$\text{(2e)} \quad \epsilon_{k,l} = \varsigma_{k,l,\text{TAXI-IN}} \qquad \forall l \in L_k, k \in K$$

$$\text{(2f)} \quad \varsigma_{k,l,p} + \delta_{k,l,p} = \epsilon_{k,l,p} \qquad \forall l \in L_k, k \in K, p \in P$$

$$\text{(2g)} \quad \epsilon_{k,l,p_1} \leq \varsigma_{k,l,p_2} \qquad \forall l \in L_k, k \in K, (p_1,p_2) \in \overline{P}$$

$$\text{(2h)} \quad \epsilon_{k,i,p_1}x_{k,(i,j)} \leq \varsigma_{k,j,p_2}x_{k,(i,j)} \qquad \forall (i,j) \in A_k, k \in K, (p_1,p_2) \in \tilde{P}$$

$$\text{(2i)} \quad \epsilon_{k,l,\text{TAXI-OUT}} = \varsigma_{k,l} \qquad \forall l \in L_k, k \in K$$

$$\text{(2j)} \quad \delta_{k,l,p} = \overline{d_{k,l,p}}y_{k,l} - \overline{d^{\text{OPT}}_{k,l,p}}\psi_{k,l,p} \qquad \forall l \in L_k, k \in K, p \in P$$

$$\text{(2k)} \quad \varsigma, \tau, \delta, \epsilon \geq 0$$

$$\text{(2l)} \quad \psi \quad \text{binary}$$

The objective term minimizes the aircraft assignment costs and additionally the costs for flight arrival delays and turnaround options. Constraint (2b) makes sure that the end time of each flight l with fleet k equals the start time plus the flight duration. The start time of a flight is bounded from below by the scheduled time of departure in (2c). Constraint (2d) measures the delay in the case that the actual touchdown is planned to be later than the scheduled time of arrival. Constraint (2e) makes sure that the taxiing process begins directly after the touchdown. Constraint (2f) ensures that every process's start time plus its duration equals its end time. Constraint (2g) states that the end time of one process does not exceed the start time of the downstream process in the process tree, whenever both consecutive processes are part of the same process set (pre-flight / post-flight). Constraint (2h) states that the end time of one process is smaller or equal to the start time of the downstream process in the process tree, whenever both consecutive processes are part of different process sets (pre-flight / post-flight). Constraint (2i) sets the end time of the taxi-out process to the start time of the corresponding outbound flight. Constraint (2j) defines the actual duration of each process step depending on the fleet used to execute the respective flight and the options taken to reduce the individual process times.

3.3. **Feasible Resource Allocation at Hub-Airports.** Some turnaround processes require the utilization of scarce resources or personnel, as boarding / deboarding stairs or loading agents. In properly designed flight schedules the number of simultaneous turnaround processes usually does not exceed the resource availability. However, when disrupted schedules are re-optimized, this scenario can occur and it has to be decided, which

Set	**Description**
R	Set of turnaround resources: {BOA/DEB, LOA/UNL, CLE, FUE, CAT, DEICING}
$\tilde{S}$	Subset of S called hub-airports, where ground resources can be steered
$L_{k,(s,r)}$	Subset of L. Flight legs which can be executed by fleet k and depart or arrive at station s and need resource r for a turnaround process
$E_{(s,r)}$	Arcs in ground resource network at station s for resource type r
Parameter	**Description**
$T \geq 0$	Length of time horizon (a sufficiently large number)
$\overline{d}^{\text{trans}}_{(s,r)} \geq 0$	On-ground default resource transfer time
$B_{(s,r)} \geq 0$	Number of resources of type r at airport s
$p(r,s,l)\colon R \times S \times L \to P$	Process p, which execution depends on resource r for flight l on airport s
Variable from (1), (2)	**Description**
$y_{k,l}$	Determining fleet-flight assignments. Equals 1 if flight l is served by fleet k
$\varsigma_{k,l,p}, \epsilon_{k,l,p}$	Start time, end time of turnaround process p of flight l executed by fleet k
Variable	**Description**
$\varsigma_{k,l,(s,r)}, \epsilon_{k,l,(s,r)}$	Start time and end time of the presence of resource r at airport s for a process p of flight l executed with aircraft k
w_e	Binary variable indicating if one unit of resource flows along resource network edge e
θ_e	Variable expressing the time transferred from e^- to e^+ along edge $e \in E_{(s,r)}$

TABLE 3. Sets, parameters and variables for the turnaround resource ground model

processes are prioritized. Since it is not clear beforehand which processes will be simultaneous, we set up a Vehicle Routing Problem with Time Windows (*VRPTW*, see Solomon (1984) or Toth and Vigo (2014)) to route every resource through all turnaround processes to be served. The model is based on one resource network per resource and hub-airport, i.e., an airport where we consider detailed resource routing useful. Hubs are airline-specific center airports where airlines have sufficient ground resources or market power of their own to effectively influence the distribution of resources. Each resource network contains a start node $v^{start}_{(s,r)}$ for each process requiring this resource and one node $v_{k,l,(s,r)}$ for each flight l arriving or departing at the airport for each fleet k which can serve the flight. There is an arc from the start node to every other node and two nodes are connected if the process corresponding to the second node can be performed after the process corresponding to the first node.

The additional input sets, parameters and variables are summarized in Table 3. The set R contains all the resources required to perform turnaround processes. The set $\tilde{S}$ is the subset of airports where the detailed resource tracking model is set up. The sets $L_{k,(s,r)}$ represent the nodes of the ground network, i.e., the flight processes which may be necessary to perform at the airport with the help of resource r. Hence, the node set of the ground network is equivalent to $\{v^{start}_{(s,r)}\} \cup \bigcup_{k \in K} L_{k,(s,r)}$. The set $E_{(s,r)}$ contains the arcs of the ground network and connects all ground network nodes which can be served consecutively, including an arc from the start node to every other node and conversely. This arc set is considered to describe a transitive binary relation on the node set, i.e., if there is for pairwise different nodes v_1, v_2, v_3 an arc from node v_1 to v_2 and an arc from v_2 to v_3, then there is an arc from v_1 to v_3 as well. The parameter $\overline{d}^{\text{trans}}_{(s,r)}$ determines the transfer time necessary to move a resource from one ground network node to the next. The parameter $B_{(s,r)}$ is equal to the number of resources of type $r \in R$ available at airport $s \in \tilde{S}$. The mapping $p(\cdot)$ assigns the corresponding turnaround process to a triple of resource $r \in R$, $s \in S$, $l \in L$. This assignment is unique for our instances, but in case of usage of identical resources for different processes, the model can easily be adapted by making the nodes of the resource network and all corresponding model variables depend on the process as well. The resource routing optimization model contains additional start / end variables $\varsigma_{k,l,(s,r)}, \epsilon_{k,l,(s,r)}$, which describe how long resources are available to perform the corresponding processes. The binary w_e are equal to 1 if the start process of arc e is executed before the end node and 0 otherwise. The continuous positive variables θ_e equal the end time of the start process of arc e if the corresponding binary variable w_e is equal to 1 and 0 otherwise. The following model extends Model (2) by adding resource-tracking constraints at every hub-airport $s \in \tilde{S}$ for every ground resource $r \in R$. The time coupling of the different turnaround processes is done in Model (2). It thus represents the main model of this paper and includes all aspects which have to be

combined in order to consider Turnaround Handling while optimizing the Aircraft Assignment.

$$\text{(3a)} \quad \min \quad (2a) + 0$$

$$\text{s.t.} \quad (1b) - (1f), (2b) - (2l)$$

$$\text{(3b)} \quad \sum_{e \in \delta^-(v^{start}_{(s,r)})} w_e \leq B_{(s,r)}$$

$$\text{(3c)} \quad \sum_{e \in \delta^+(v_{k,l,(s,r)})} w_e = y_{k,l} \qquad \forall k \in K, l \in L_{k,(s,r)}, r \in R, s \in \tilde{S}$$

$$\text{(3d)} \quad \sum_{e \in \delta^-(v_{k,l,(s,r)})} w_e = y_{k,l} \qquad \forall k \in K, l \in L_{k,(s,r)}, r \in R, s \in \tilde{S}$$

$$\text{(3e)} \quad \sum_{e \in \delta^+(v_{k,l,(s,r)})} \theta_e + \overline{d}^{\text{trans}}_{(s,r)} y_{k,l} \leq y_{k,l} \varsigma_{k,l,(s,r)} \qquad \forall k \in K, l \in L_{k,(s,r)}, r \in R, s \in \tilde{S}$$

$$\text{(3f)} \quad \sum_{e \in \delta^-(v_{k,l,(s,r)})} \theta_e = y_{k,l} \epsilon_{k,l,(s,r)} \qquad \forall k \in K, l \in L_{k,(s,r)}, r \in R, s \in \tilde{S}$$

$$\text{(3g)} \quad \varsigma_{k,l,(s,r)} = \varsigma_{k,l,p(r,s,l)} \qquad \forall k \in K, l \in L_{k,(s,r)}, r \in R, s \in \tilde{S}$$

$$\text{(3h)} \quad \epsilon_{k,l,(s,r)} = \epsilon_{k,l,p(r,s,l)} \qquad \forall k \in K, l \in L_{k,(s,r)}, r \in R, s \in \tilde{S}$$

$$\text{(3i)} \quad \theta_e \leq T w_e \qquad \forall k \in K, e \in E_{(s,r)}, r \in R, s \in \tilde{S}$$

$$\text{(3j)} \quad \theta \geq 0$$

$$\text{(3k)} \quad w \quad \text{binary}$$

$$\text{(3l)} \quad \varsigma, \epsilon \quad \text{continuous}$$

Problem (3) contains all necessary elements in order to include accurate Turnaround Handling decisions into the Aircraft Assignment optimization and is thus our final model. The objective function (3a) of the model remains the same as in Model (2) as all relevant costs are already covered. Nevertheless, due to the additional resource allocation constraints for resources on ground, the objective value is influenced by further restricting the set of feasible solutions in terms of feasible turnaround timing decisions. Constraint (3b) limits the amount of resource r used to perform turnaround processes at station s. Constraints (3c) make sure that every node in the ground network has as many predecessor connections as the number of resources of type r to perform the turnaround process. In our model this is always equal to the corresponding y-variable, i.e., either 0 or 1. Constraints (3d) guarantee the same for succeeding connections. Constraints (3e) ensure that the start time of a turnaround process is at least the sum of time transmission going into the corresponding ground node plus the transfer time. Constraints (3f) make sure that the end time of a turnaround process equals the time transmission going out of the node. The process start and end times of Model (2) bound the times in Constraints (3g) and (3h) at which a resource belonging to the respective process must be available. Constraint (3i) guarantees that time is only transmitted along edges which are chosen.

The additional Constraints (3b)-(3l) exclusively influence the schedule of the executed turnarounds. They restrict the number of turnaround processes which can be executed simultaneously to the number of resources available for their execution. Nevertheless, all aircraft assignment combinations that are feasible under consideration of Model (1) can be extended to feasible solutions of Model (2), because the delay of flights is not bounded from above.

Although there is no immediate cost in routing resources on the ground, the problem is still complex: at an airport, the routing network is a complete graph, unless constraints on the maximum delay are imposed. Hence, the size of resource network grows quadratically with the number of turnaround processes.

Besides the already introduced capabilities of the model, it is also easy to integrate further features into the problem, like, e.g., Maintenance Planning or Crew Scheduling by introducing binary variables indicating the execution of maintenance or the selection of certain crew plan constellations. Turnaround Slot Time restrictions can also be incorporated by adding binary selection variables which indicate in which time slot the turnaround has to take place.

Our overall optimization model (3) combines complex problem structures. Using standard optimization software, it is difficult to solve even for small problem instances. This is demonstrated later in Section 6. Hence, in the next section we present a decomposition algorithm, which enables exact solving of the optimization model presented.

4. Solution Strategy

The integrated model for airline operational planning, stated in (3), cannot be solved efficiently with straightforward approaches, even for small instances. We locate the complicating factors at two points: The Aircraft Assignment part of the model (1) contains at least $\sum_{k\in K}|A_k|$ binary variables, one for each fleet-connection pair. This number grows quadratically in relation to the number of flights that have to be assigned. In addition, this assignment problem is combined with the additional routing constraints in (3), which corresponds to $|\tilde{S}||R|$ sub routing networks and thus contains $|E_{(s,r)}|$ binary variables for each sub network. The time dependencies between departure times and turnaround process times are modeled in (2) and are also controlled by binary variables. The structure of the SP is equivalent to a vehicle routing problem (VRP) with time windows (VRPTW), even if all master decisions are fixed. This problem class is known to be notoriously hard to solve, see for example Caceres-Cruz et al. (2014) for a good survey, or Salavati-Khoshghalb et al. (2019) who solved a VRP with stochastic demands for networks with up to 60 nodes and 9 scenarios or Gmira et al. (2021) who solved a time dependent VRPTW with up to 200 nodes with a heuristic approach or Fügenschuh (2006) who solved VRPs with coupled time windows with up to a few hundred nodes to optimality with an exact approach but also was not able to solve other networks with only 25 nodes to optimality. The fact that Model (3) has decision variables which determine which nodes the turnaround resource routing has to serve makes the problem even harder. Thus, we propose a decomposition based solution procedure which exploits the structure of the model to produce provably good solutions in acceptable time.

The remainder of this section is therefore concerned with the development of this decomposition approach, which we refer to as the island decomposition approach. In Section 4.1 the basic idea is explained. This involves splitting the overall Problem (3) into into a Master and Sub problem. In Section 4.2 we improve the basic decomposition approach by defining *islands*, which denote sets of flight connections. Section4.3 introduces different special types of islands and explains how they can be calculated and affect the decomposition approach. Section 4.4 shows how valid cuts can be derived based on given islands. All concepts are put together in Section 4.5, where the main algorithm, which is capable of solving Model (3), is proposed. The algorithm is proven to be an exact (not a heuristic) approach.

4.1. The basic decomposition idea.

An approach to solve Model (3) is proposed which has parallels to Benders' decomposition (Benders (1962)), but exploits model-specific structures. Model (3) is first decomposed into two problems, called **Master Problem (MP)** and **Sub Problem (SP)**. The master problem corresponds one-to-one with the presented Aircraft Assignment Problem (1). The objective function term is extended by a non-negative continuous variable σ, which will represent the costs arising in the SP:

$$\text{(MP)} \qquad \begin{aligned} \min \quad & \sum_{k\in K}\sum_{l\in L_k} c_{k,l}y_{k,l} + \sigma \\ \text{s.t.} \quad & \text{(1b)} - \text{(1f)},\ \sigma \geq 0 \end{aligned}$$

The sub problem depends on a solution of the master problem, i.e., (x^*, y^*, σ^*) feasible for problem (MP). The objective includes all turnaround related costs contained in 3a and is thus non-negative. The model consists of all constraints in Model (3) that are not covered in the master problem. It can thus be understood as a turnaround planning model based on the aircraft assignments in the MP:

$$\text{(SP)} \qquad \begin{aligned} \min \quad & \sum_{k\in K}\sum_{l\in L_k}\left(c^{\text{delay}}_{k,l}\tau_{k,l} + \sum_{p\in P} c^{\text{OPT}}_{k,l,p}\psi_{k,l,p}\right) \\ \text{s.t.} \quad & \text{(2b)} - \text{(2l)} \\ & \text{(3b)} - \text{(3l)} \end{aligned}$$

MP and SP do not divide the strongly structurally related problem parts. Splitting the Aircraft Assignment problem is not considered reasonable, since the cover constraint (1d) can lead to infeasibility if some of the flight legs are assigned to some aircraft without taking into account the remaining schedule. Hence, we added the procedure of assigning aircraft to flight legs completely in the MP. The remainder of the problem has been put in the SP. What we get is a decomposition approach with a Binary Program (BP) as MP and a Mixed Integer Linear Program (MILP) as SP.

We will briefly discuss on which concepts from the literature exist for this situation. The discussion is limited to different types of benders decomposition that can be found in the literature.

For the sake of completeness, the basic Benders Decomposition introduced in Benders (1962) works with sub problems, which do not include integer variables. This is crucial in order to use standard linear programming duality to derive valid cuts for the master problem. It works for integer sub problems, if the so-called integrality gap, i.e., the difference between the solution value of an integer program and the solution value of its continuous relaxation, is zero or low for SP for a large proportion of MP solutions.

In this case, the integer constraints in the sub problem are often neglected as in Cordeau et al. (2001) and Mercier and Soumis (2007). Thus, the feasibility / optimality cuts can be derived as in Benders (1962). After convergence of the method, an integer sub problem solution has to be computed based on the fixed solution of the master problem. Since the integrality gap of our SP is not low, this approach is ruled out, but will be considered as a benchmark approach in Section 5.

The approach could be improved by adding valid cutting planes to the SP in order to obtain a lower integrality gap. This procedure was tested in Sherali and Fraticelli (2002) using the reformulation-linearization technique to generate cutting planes. The number of constraints which have to be added to the SP to reduce the integrality gap significantly grows exponentially with the number of integer variables. Hence, this approach cannot be applied.

Instead of neglecting integrality constraints, the authors of Carøe and Tind (1998) make use of general duality theory of integer programs in order to generate valid cuts in the context of stochastic optimization. Nevertheless, the obtained dual sub problem has to be solved very often which we consider as difficult as the primal problem itself. As we will show in our computational experiments in Section 6, this is crucial, since even to solve the SP in isolation takes a lot of time. Hence, it is a necessary criterion for an efficient decomposition based solution approach to limit the number of sub problem solves as much as possible.

Instead of working with the difficult-to-get duals of integer programs, Codato and Fischetti (2004) established combinatorial benders cuts. These require a combinatorial master problem and a sub problem which is a feasibility problem. Whenever the sub problem is infeasible a cut is generated which excludes the current master problem solution from the set of feasible solutions. Since the SP is not a feasibility problem in our case, the approach is not directly applicable.

Instead, we want to follow the idea of logic-based benders approach, which is presented in Hooker and Ottosson (2003). The idea is to determine lower bound functions, depending on the master variables, for

objective values of the sub problem. These functions are used to generate cuts for the master problem, and under certain conditions it can be shown that the procedure terminates in a finite amount of time. In general, the method leaves a lot of space for interpretation and works for very general sub problem types, e.g. non-linear programs but also for integer programs. It remains to specify how the functions defining are generated for our optimization model, which is described in the following. To support computational tractability of the approach, we postulate the function generation procedure following some principles:

The functions should provide **non-trivial** lower bounds for as many as possible feasible master solutions at once, which are as tight as possible. To get a trivial lower bound function, one could take the indicator function of the current master solution, multiplied with the corresponding sub problem objective value.

The functions should be reasonably **easy to determine**. The full value function of the sub problem would be a candidate for such a lower bound function, but it is difficult to calculate and to store the value function of a complicated integer program.

The functions should be reasonably **easy to integrate** into the master problem, optimally without requiring to introduce new variables and only requiring linear constraints. The full value function of the sub problem is an example which violates the principle, since adding a value function of an integer program to the objective function of an MIP results in an MINLP.

A similar procedure is described in Laporte and Louveaux (1993), where the point of view is to use lower bound functions to map the cost of the sub problem into the master problem. Here, the requirement is explicitly that the master problem contains only binary variables, which is the case for our MP, even if it is extended by additional model aspects, like maintenance or crew planning, since the extensions can be realized using binary variables only.

Hence, the approach we developed is based on the use of lower bound functions. Since in the standard solution approach of Laporte and Louveaux (1993) a frequent solving of the sub problem required, we adjust this method application specific, so that only reduced forms of the SP have to be solved. We make use of the notation introduced in Table 4.

Whenever a variable (e.g. $y_{k,l}$) either in the MP or the SP comes with an asterisk (*, e.g. $y_{k,l}{}^*$), this means

Notation	**Description**
x^*, y^*, σ^*	Optimal solution of the master problem (MP)
$\psi^*, w^*, \varsigma^*, \epsilon^*, \delta^*, \tau^*, \theta^*$	Optimal solution of the sub problem (SP)
z^{MP}	Optimal objective value of the master problem (MP)
$z^{\mathrm{SP}}(x^*, y^*) = z^{\mathrm{SP}}(x^*) = z^{\mathrm{SP}}$	Optimal objective value of the sub problem (SP)

TABLE 4. Notation used in Section 4 for the island decomposition approach

the value of this variable in an optimal solution determined by some algorithm. The corresponding optimal objective values are denoted with z^{MP} for the MP and z^{SP} for the SP. Since the solution of the SP depends on the variable assignment of the MP it can also be written as $z^{\mathrm{SP}}(x^*, y^*)$ and since the assignment of x^* fully determines y^* simplified as $z^{\mathrm{SP}}(x^*)$.

A simplified method to calculate valid, but quite trivial lower bounds for the SP objective value depending on the MP variables is the following. The MP and the SP is solved iteratively, starting with the MP. The Aircraft Assignment plan in form of fixed flight connections is determined. As shown in the next section, all selected connections determine the right hand side of the SP in a way that many sub problem variables are implied to be 0, which simplifies the SP. Based on this, it is possible to calculate the solution of SP and pass information about the objective value z^{SP} to the MP in the form of additional constraints. Then the MP is reoptimized. This approach is illustrated in Figure 3.

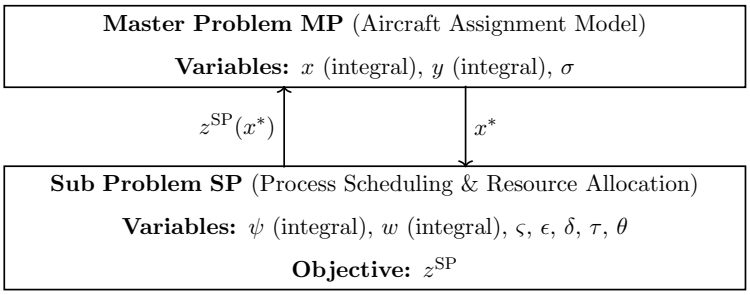

FIGURE 3. Scheme of iterative decomposition based on the variables used in the model

In order to conduct the approach, it is necessary to derive constraints for the MP based on the current value $z^{\mathrm{SP}}(x^*)$. Hence, the MP objective function (1a) is extended by a variable $\sigma \geq 0$, which is bounded from below by $z^{\mathrm{SP}}(x^*)$ (non-negative) in the case that x^* is selected in the MP.

Using the indicator function $\mathbb{1}_{\cdot}$ such a constraint can be formulated abstractly as follows:

$$z^{\mathrm{SP}}(x^*) \cdot \mathbb{1}_{x=x^*} \leq \sigma \tag{4}$$

To elaborate the term $\mathbb{1}_{x=x^*}$, one can make use of the special structure of the MP. Due to the flight cover constraints (1d) in combination with the flow constraints (1b) and (1c), it can be derived, that the set of positive connection variables determine the values of all other variables contained in the MP. Thus, the solution of the MP can be characterized by the set

$$\Omega := \{(k,i,j) | k \in K, (i,j) \in A_k, {x_{k,(i,j)}}^* = 1\}, \tag{5}$$

containing all flight connections that are selected in the current MP solution and Constraint (4) becomes

$$z^{\mathrm{SP}} \cdot \Big(\sum_{(k,i,j)\in\Omega} x_{k,(i,j)} - (|\Omega| - 1) \Big) \leq \sigma. \tag{6}$$

The proposed procedure trivially implements the determination of lower bound functions as described in Laporte and Louveaux (1993). In every iteration, directly after solving the SP, one additional cut (6) bounding σ from below is added to the MP and kept during the whole solution process, which terminates with an optimal solution of Model (3) if one MP solution is found the second time. This happens after a finite number of steps, since the pure-binary MP has a finite number of feasible solutions. A feasible solution of the model is determined in every iteration, provided that the entire model (3) is feasible.

However, since in each iteration only one additional MP solution enforces to a positive σ, whenever there exist many good solutions of MP which lead to some costs in SP, the procedure takes many iterations, which is computationally inefficient due to the complexity of our SP.

Hence, following the idea of Hooker and Ottosson (2003), we want to make use of the problem structure in order to be able to obtain cost information for the MP by solving smaller and relaxed versions of the SP, leading to more powerful lower bound functions able to estimate the value of the SP for many MP solution at once. This will be done essentially by identifying subsets of Ω that incur costs independently on the rest of the MP solution. We call these subsets *islands* and introduce them in the following Section 4.2.

4.2. **Islands and Island Costs.** The goal of this section is to improve the basic algorithm by identifying subsets of flight connection variables and corresponding delay and option variables in the SP. These subsets have the property that assigning all included flight connection variables to 1 forces costs in the SP, which are caused by the corresponding delay and option variables. We propose several methods to identify candidate subsets of master variables with the property that setting them to 1 in the MP causes a certain cost contribution in the SP in Section 4.3. Next, we define these subsets that we call *islands*.

Definition 4.1. *Islands and island costs. We call an arbitrary set $\mathscr{I}$ of fleet-connection tuples (k,i,j) corresponding to x-variables of* (MP) *an island. For an island $\mathscr{I}$, island costs are defined as*

$$
\begin{aligned}
z^{isl}_{\mathscr{I}} := \quad & \min \sum_{(k,i,j)\in\mathscr{I}} (c^{delay}_{k,j}\tau_{k,j} + \sum_{p\in P^{post}} c^{OPT}_{k,i,p}\psi_{k,i,p} + \sum_{p\in P^{pre}} c^{OPT}_{k,j,p}\psi_{k,j,p}) \\
\text{(IC)} \qquad & s.t. \quad \text{(1b)} - \text{(1f)}, \quad \text{(2b)} - \text{(2l)}, \quad \text{(3b)} - \text{(3l)} \\
& x_{k,(i,j)} = 1 \ \forall (k,i,j) \in \mathscr{I}, \\
& y_{k,l} = 1 \ \forall k \in K, l \in \{l' \in L_k : \exists \hat{l} \in L_k : (k,\hat{l},l') \in \mathscr{I} \vee (k,l',\hat{l}) \in \mathscr{I}\}.
\end{aligned}
$$

In words: The island costs are costs which at least emerge locally in SP if all connections in MP corresponding to $\mathscr{I}$ are fixed to 1*. We call an island $\mathscr{I}$ active for a solution x^* of MP, if for all $(k,i,j) \in \mathscr{I}$ $x^*_{k,(i,j)} = 1$.*

Example 4.2. *An example for an island and corresponding costs is one single connection: Consider flight leg* 1 *arriving at 10am and flight leg* 2 *departing (planned) at 10:30am with a regular turn time of 40 minutes and an option to reduce the turn time to* 30 *minutes at costs of \$*500 *and costs of \$*30 *per delay minute. This connection contributes to the SP objective value with at least the amount of \$*300 *(delay flight* 2 *by* 10*min). If any combined MP-SP-solution has costs of \$*300 *emerging on this connection, we know that this equals the island's costs.*

This example demonstrates how costs can be explicitly allocated to small substructures of the overall flight plan. Solving the introduced model (IC) frequently to determine island costs $z^{\text{isl}}_{\mathscr{I}}$ is intractable, because problem (IC) is nearly as complex as the integrated model (3). Instead, we never solve (IC) explicitly but aim for calculating tight lower bounds $z^{\text{isl, rel}}_{\mathscr{I}}$ of $z^{\text{isl}}_{\mathscr{I}}$ given an island $\mathscr{I}$. Knowing lower bounds for the island costs is sufficient for our framework, as will be shown later. Unfortunately it is not sufficient to solve SP once and derive island costs by

$$
z^{\text{SP}}_{\mathscr{I}} := \sum_{(k,i,j)\in\mathscr{I}} \big(c^{\text{delay}}_{k,j}\tau^*_{k,j} + \sum_{p\in P^{\text{post}}} c^{OPT}_{k,i,p}\psi^*_{k,i,p} + \sum_{p\in P^{\text{pre}}} c^{OPT}_{k,j,p}\psi^*_{k,j,p}\big), \tag{7}
$$

which denote the costs arising on island $\mathscr{I}$ in the solution of the SP. If this were the case, all relevant island costs could be determined by solving the sub problem once. We next observe that the island costs $z^{\text{isl}}_{\mathscr{I}}$ are a lower bound of the costs $z^{\text{SP}}_{\mathscr{I}}$ emerging on the island connections in a corresponding SP solution.

Observation 4.3. *Let x^* be a solution of Problem* (MP) *represented in the corresponding set Ω as defined in* (5)*. Given an island $\mathscr{I} \subseteq \Omega$, it holds that*

$$
z^{SP}_{\mathscr{I}} \geq z^{isl}_{\mathscr{I}}, \tag{8}
$$

where $z^{SP}_{\mathscr{I}}$ denotes the costs of $\mathscr{I}$ arising in SP (x^*) and $z^{isl}_{\mathscr{I}}$ denotes the solution value of (IC)*.*

The observation is correct, since the solution of the SP in the form of variable assignment is feasible for the island problem (IC) and additionally the value $z^{\text{SP}}_{\mathscr{I}}$ is the value of Problem (IC)'s objective function, evaluated at the solution point. Hence, it has to be at least as high as the optimal value of the problem. Furthermore, Lemma (4.4) gives insight about the correlation between the island costs of several islands and the costs incurred in the sub problem.

Lemma 4.4. *Let x^* be a solution of Problem* (MP) *and let $\mathscr{E}$ be a collection of pairwise disjoint Islands which are active for x^*. Then it holds that $z^{SP} \geq \sum_{\mathscr{I}\in\mathscr{E}} z^{isl}_{\mathscr{I}}$.*

In order to limit the size of this paper, the proof of Lemma 4.4 can be found in the appendix.

Next we show how easy-to-solve relaxations of Problem (IC) can be determined. Hence, an observation is presented which describes the solutions of Problem (IC), when the Aircraft Assignment model constraints are dropped. It serves as the theoretical substantiation of the following steps.

Lemma 4.5. *Let $\mathscr{P}$ be Problem* (IC) *for island $\mathscr{I}$ neglecting constraints* (1b)-(1f) *and let $\mathscr{F}$ be the set of all fleet-flight combinations which are not included in $\mathscr{I}$, namely*

$$\mathscr{F} := (k,l) \in \bigcup_{k' \in K} \{k'\} \times L_{k'} \setminus \{(k',l') : \exists j : (k',l',j) \in \mathscr{I} \vee \exists i : (k',i,l') \in \mathscr{I}\}.$$

Then an optimal solution for $\mathscr{P}$ exists, in which the variables $y_{k,l}$, $\varsigma_{k,l}, \epsilon_{k,l}, \tau_{k,l}$ and for all $p \in P$ the variables $\varsigma_{k,l,p}, \epsilon_{k,l,p}, \delta_{k,l,p}, \psi_{k,l,p}$, as well as w, θ corresponding to the fleet-flight combinations $(k,l) \in \mathscr{F}$, as well as all connection variables $x_{k,(i,j)}$ for all

$$(k,i,j) \in \mathscr{G} := \bigcup_{k' \in K} \{k'\} \times A_k \setminus \mathscr{I}$$

have value 0.

The quite technical and lengthy proof of Lemma 4.5 is in the appendix. The idea of the proof is that whenever any variable $y_{k,l}$ for $(k,l) \notin \mathscr{F}$ or $x_{k,(i,j)}$ for $(k,i,j) \notin \mathscr{G}$ is nonzero, we can just set it to zero, and adjust the remaining variables of Problem (IC) such that feasibility is restored, while the objective value does not get larger.

Lemma 4.5 states, that only x-variables contained in the island have to be considered. Therefore, the ground process constraints (2b)-(2l) have to be included only partially as well, containing all time (ς, ϵ), duration (δ), delay (τ) and option (ψ) variables associated with connection-fleet or flight-fleet combinations contained in $\mathscr{I}$. The turnaround resource constraints (3b)-(3l) are introduced for all hubs where a resource shortage can occur. In this case, all resource flow (w) and resource transmission (θ) variables are taken into account which are associated with connection-fleet and flight-fleet combinations belonging to the relevant Island $\mathscr{I}$.

It turns out, that reductions of the SP cannot only be derived due to solution of the MP, but also due to the structure of considered islands. For example, islands considering connections on just one flight string do not cause resource shortages since thus no parallel turnarounds are considered. Hence, all constraints and variables of constraints (3b)-(3l) can be neglected here as well. The structure of the resulting optimization models and thus the effort to find the optimal solution heavily depends on the connections / island under consideration. This is addressed in the following Section 4.3.

4.3. **Island Identification.** In this section we present approaches to find different types of islands. These types differ by the computational effort of identifying them and finding lower bounds $z_{\mathscr{I}}^{\text{isl, rel}}$ for their corresponding island costs $z_{\mathscr{I}}^{\text{isl}}$. In the following we will describe four different types of islands. For each type it is described how they can be identified and in which way the island cost Problem (IC) can be simplified in order to obtain the corresponding island costs.

Single Connection Island (SC). A large part of costs occuring in the SP in form of turnaround or delay costs are caused by single flight connections with too short periods of time between the arrival and departure of the connected flights to carry out all necessary turnaround processes. In this case $|\mathscr{A}| = 1$, which means that costs of optional turnaround quickening and delay costs of the connected flight are just weighed against each other and the most favorable combination is selected.

Since all islands are implicitly given by the connection structure of the problem instance and do not depend on a MP-solution, all (SC)-islands and the corresponding island costs can be calculated before optimizing the SP, or even the MP by determining the minimum between option and delay costs on every connection. This

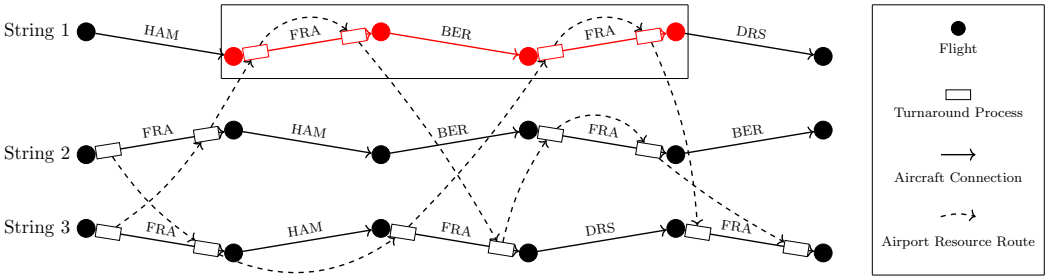

FIGURE 4. (SF)-island consisting of consecutive flight connections on one single flight string

is possible because the number of connections per fleet is in $\mathcal{O}(|K||L|^2)$. Real flight plans hardly get close to this bound. Hence, the procedure does not influence the runtime perceptibly.

Single Flight String Island (SF). Since costs are not only induced by single flight connections, flight sequences on individual flight strings have to be investigated as well. The idea is that all costs occurring on parts of one flight string are intrinsically caused by this flight sequence. An (SF)-island depends on chosen flight sequences and therefore depends on a feasible solution of the MP. It starts at a connection (i, j) executed by some aircraft of type $k \in K$ with positive costs or positive delay/delay costs on flight leg j. The end connection of a (SF)-island is a connection (h, l) with 0 delay on flight leg l and island costs on $\{(i, j), ..., (h, l)\}$ being equal to the island costs on $\{(i, j), ..., (h, l), (l, m)\}$ reduced by the (SC)-island costs on $\{(l, m)\}$. These conditions are checked along a flight string successively until the end connection is obtained and the island is determined. Algorithm 16 sketches how (SF)-islands are generated and the corresponding costs are calculated. Note that Algorithm 16 is only written informally and the algorithm written down using rigorously mathematical notation can be found in the appendix. The corresponding lower bound for the island costs are calculated by solving the following relaxation of Model (IC):

$$\text{(IC-SF)} \qquad \begin{aligned} z^{\text{isl, rel}}_{\mathscr{I}} \;:=\; & \min \sum_{(k,i,j)\in\mathscr{I}} \Big(c^{\text{delay}}_{k,j}\tau_{k,j} + \sum_{p\in P^{\text{post}}} c^{OPT}_{k,i,p}\psi_{k,i,p} + \sum_{p\in P^{\text{pre}}} c^{OPT}_{k,j,p}\psi_{k,j,p}\Big) \\ & \text{s.t.} \quad (2\text{b}) - (2\text{l}), x_{k,(i,j)} = 1 \;\forall (k,i,j) \in \mathscr{I}, x_{k,(i,j)} = 0 \;\forall (k,i,j) \notin \mathscr{I}. \end{aligned}$$

Problem (IC-SF) includes significantly fewer variables and constraints compared to Problem (IC), since the resource routing is neglected. Moreover, the problems can be optimized without having to calculate a solution of the SP beforehand. This is necessary because costs for (SF)-islands must be calculated solving Problem (IC-SF) whenever the solution of the MP implies new flight sequences. It can be divided into two components. First, the costs on a given flight string are weighed against each other from a certain connection as described to identify the relevant flight string part. Then, Problem (IC-SF) is solved to compute the cost associated with the island.

Multiple Flight Strings Island (MF). In order to determine candidates for (MF)-islands, we compare the delay and cost structure along flight strings in the SP with and without resource tracking constraints consisting of Model (3). The procedure is the following: it is iterated along flight strings (determined by the MP) for each aircraft. For each flight connection we compare costs caused by this connection (option and delay costs) and the delay on the outgoing leg of this connection in the (SF)-island calculation model (the model without resource constraints) and in the SP model (with resource constraints included).

We denote the corresponding delay and costs arising on one connection in the SP as the (SP)-delay and (SP)-costs, the delay and costs arising in the (SF)-islands as (SF)-delay and (SF)-costs. Here, four different cases can occur, which depend on the behaviour of the sub problem solution including resource routing constraints

Algorithm 1 Generate SF island

INPUT: Model (IC-SF), MP-solution with selected connection (k, i, j) with $k \in K, (i, j) \in A_k$
OUTPUT: $\mathscr{I}$, $z_{\mathscr{I}}^{\text{isl, rel}}$
procedure GEN_SF(IC-SF, (k, i, j), MP-solution):
    $\mathscr{I} \leftarrow \{(k, i, j)\}$ ▷ obtain island
    for connection c in ordered flight string including (i, j) **do**
        Add c to potential island $\mathscr{I}$
        Calculate island costs of $\mathscr{I}$ by solving Problem (IC-SF)
        if costs of island c as part of $\mathscr{I}$ == (SC)-island costs of c **then**
            delete c from $\mathscr{I}$
            break
        end if
    end for
    $z_{\mathscr{I}}^{\text{isl, rel}} \leftarrow$ Solve Problem (IC-SF) ▷ calculate island costs
    return $(\mathscr{I}, z_{\mathscr{I}}^{\text{isl, rel}})$
end procedure

Case	Setting	Description
1	(SP)-delay > (SF)-delay, (SP)-costs ≥ (SF)-costs	If the connection is the first connection on the current flight string section a deviation is caused on the airport of the connection because of resource shortages during some turnaround process on the connection. Hence it is necessary to look for resource bottlenecks immediately *before* the current connection.
2	(SP)-delay > (SF)-delay, (SP)-costs < (SF)-costs	In the SP-solution costs are omitted by permitting higher delays due to a later turnaround resource shortage on the flight string. Hence one has to look for resource bottlenecks *before* the current connection and before subsequent connections of the flight string at hub-airports.
3	(SP)-delay ≤ (SF)-delay, (SP)-costs > (SF)-costs	More costs to save time emerge in the SP-solution. On the observed connection or a later one of the flight string some process has to be finished earlier, so resource shortages occur on subsequent connections of the flight strings located at hub-airports *after or during* the connection's turnaround processes.
4	(SP)-delay ≤ (SF)-delay, (SP)-costs ≤ (SF)-costs	This behavior is caused by a Case 3 on a predecessor connection; it is proceeded with the predecessor connection.

TABLE 5. Case distinction for different settings which compare delay and costs on one connection in the SP and in Problem (IC-SF)

and are precisely described in Table 5. In the cases 1 and 2 resource bottlenecks have to be considered before the relevant connection and in case 3 and 4 these bottlenecks occur after the relevant connection.

Based on these insights we propose a heuristic procedure to determine *conflicting connections* which cause resource bottlenecks at hub-airports. If the bottleneck is expected before a certain connection (Case 1, Case 2), we go through the processes of the dedicated connection and look for predecessor processes with a tight time dependency in the current SP solution. There is a tight time dependency between two processes p_1, p_2 on flight legs l_1, l_2 served by fleet k_1, k_2 if they successively use the same resource r on airport s and

$$\epsilon^*_{k_1,l_1,(s,r)} + \overline{d}^{\text{trans}}_{(s,r)} \leq \varsigma^*_{k_2,l_2,(s,r)}$$

derived from constraints (3e) and (3f) holds with equality. This means that there is no time between the end of the first process and the start of the transfer as well as between the end of the transfer and the start of the second process. For flight legs l_1, l_2, we determine all flights passing the turnaround simultaneously via $\{i \in L_{s,r} : \epsilon^*_{k_i,i} \geq \epsilon^*_{k_1,l_1}, \varsigma^*_{k_i,i} \leq \varsigma^*_{k_2,l_2}\}$, i.e., exactly the flight legs that are prioritized over l in terms of resource allocation. All connections on hub-airports corresponding to the flights detected are considered as conflicting.

In Case 3 and 4 we look for successor processes with an almost tight time dependency, i.e., if l_1 is the current processes flight and l_2 is the successor processes flight, and k_1, k_2 are the currently assigned fleets,

$$\epsilon^*_{k_1,l_1,(s,r)} + \overline{d}^{\text{trans}}_{(s,r)} \leq \varsigma^*_{k_2,l_2,(s,r)}$$

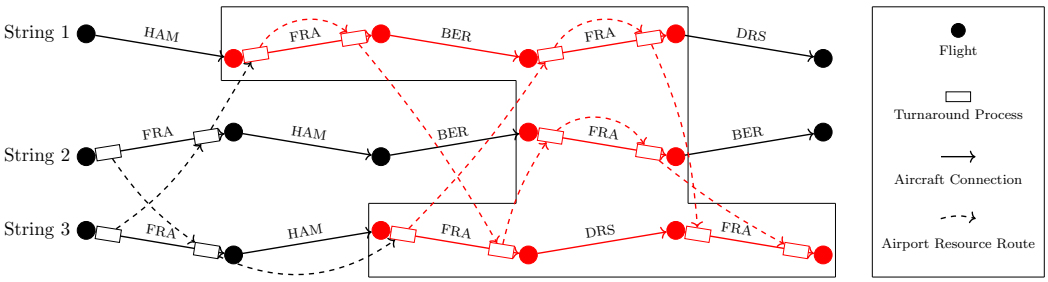

FIGURE 5. (MF)-island detected over flight strings which are connected by turnaround events taking place at the same hub-airport in similiar time windows

derived from constraints (3e) and (3f) holds with a slack of at most the amount of time difference of the current acceleration option compared to the next cheaper acceleration option. The determination of conflicting flight legs is equivalent to the procedure above.

Once all resource conflicts are detected, we unite (transitively) the (SF)-islands of connections which are in conflict with each other to (MF)-islands. The island costs are estimated by solving a reduced version of problem (IC), incorporating only connections which are contained in the island:

$$\text{(IC-MF)} \qquad \begin{aligned} z_{\mathscr{I}}^{\text{isl, rel}} \;:=\; & \min \sum_{(k,i,j)\in\mathscr{I}} (c_{k,j}^{\text{delay}}\tau_{k,j} + \sum_{p\in P^{\text{post}}} c_{k,i,p}^{OPT}\psi_{k,i,p} + \sum_{p\in P^{\text{pre}}} c_{k,j,p}^{OPT}\psi_{k,j,p}) \\ & \text{s.t.} \quad (2\text{b})-(2\text{l}),(3\text{b})-(3\text{l}), x_{k,(i,j)} = 1 \;\forall (k,i,j)\in\mathscr{I}, x_{k,(i,j)} = 0 \;\forall (k,i,j)\notin\mathscr{I}. \end{aligned}$$

The overall procedure to identify is sketched in Algorithm 20 informally and correctly detailed in the appendix. Given an (SF)-island, it searches for conflicts on each included connection. Therefore the pool of connections

Algorithm 2 Generate (MF)-island

```
INPUT: Model (IC-MF), MP Solution, SP Solution, active (SF)-Island Pool
OUTPUT: 𝒥, z^{isl, rel}_𝒥
procedure GEN_MF(IC-MF, MP Solution, SP Solution, active (SF)-Island Pool)
    Initialize island 𝒥 = ∅ and choose initial (SF)-island B to not be observed      ▷ obtain island
    while unobserved (SF)-islands exist do
        Choose arbitrary unobserved (SF)-island B
        for connection c in B do
            Add all (SF)-islands with conflicting connection to c (cases 1-4) to unobserved islands
        end for
        Add B to 𝒥 and tag B as observed
    end while
    𝒥 ← connections in all observed islands
    z^{isl, rel}_𝒥 ← Solve Problem (IC-MF)                                          ▷ calculate island costs
    return (𝒥, z^{isl, rel}_𝒥)
end procedure
```

which has to be checked for conflicts grows. When no connections are left to be checked, the algorithm obtains a (MF)-island by combining all (SF)-islands which include at least one conflicting connection. In the end, Problem (IC-MF) is solved to obtain the corresponding island costs.

Master Solution Island (MS). The largest possible islands are (MS)-islands, which consist of all connection variables selected in the current MP solution. They guarantee the overall algorithm to be exact. One (MS)-island is created in every decomposition iteration whenever the complete SP is optimized, since the island costs equal to the objective value $z^{\text{SP}}(x^*)$ of SP. The constraints added to the MP for this kind of island are

equivalent to the cuts (6) presented in Section 4.1. The usage of (MS)-islands ensures that all SP costs will be covered and that the overall procedure terminates.

Algorithm 3 Generate (MS)-island

1: **INPUT**: MP solution Ω (5), SP objective value z^{SP}
2: **OUTPUT**: $\mathscr{I}$, $z_{\mathscr{I}}^{\text{isl, rel}}$
3: **procedure** GEN_MS(Ω, z^{SP})
4: $\quad\mathscr{I} \leftarrow \Omega$ ▷ obtain island
5: $\quad z_{\mathscr{I}}^{\text{isl, rel}} \leftarrow z^{\text{SP}}$ ▷ calculate island costs
6: $\quad$**return** ($\mathscr{I}$, $z_{\mathscr{I}}^{\text{isl, rel}}$)
7: **end procedure**

4.4. **Combining Islands to generate Cuts.** In this Section it is explained how cuts can be derived from the given islands and the corresponding island costs, which bound σ in as a part of the MP-objective appropriately. Whenever a new island $\mathscr{I}$ is detected (no matter which type) and corresponding island costs $z_{\mathscr{I}}^{\text{isl, rel}}$ are calculated, the island is saved in the set $\mathscr{E}$, the island pool. Furthermore, a variable $\sigma_{\mathscr{I}}$ is added to the master problem, together with the additional constraint

$$\sigma_{\mathscr{I}} \geq z_{\mathscr{I}}^{\text{isl, rel}} \Big(\sum_{(i,j,k)\in\mathscr{I}} x_{k,(i,j)} + 1 - |\mathscr{I}| \Big). \tag{9}$$

For a given MP solution x^* we call an island $\mathscr{I} \in \mathscr{E}$ *active*, if $(k,i,j) \in \mathscr{I} \Rightarrow x^*_{k,(i,j)} = 1$. The variables $\sigma_{\mathscr{I}}$ are to interpret as "if an island $\mathscr{I}$ is active, then the variable $\sigma_{\mathscr{I}}$ takes the island costs as value, otherwise 0". They express a part of the sub problem costs caused by specific structures in the MP solution. The $\sigma_{\mathscr{I}}$-variables cannot be directly incorporated into the master problems objective function, since two active islands $\mathscr{I}$ and $\mathscr{J}$ do not necessarily have to be disjoint. Whenever this happens, the sum $\sigma_{\mathscr{I}} + \sigma_{\mathscr{J}}$ may overestimate the local SP costs if both $\mathscr{I}$ and $\mathscr{J}$ are active.

However, if $\mathscr{E}' \subseteq \mathscr{E}$ is a subset of disjoint and active islands, it holds that the sum of the respective island costs is at most the SP objective value:

$$z^{\text{SP}} \geq \sum_{\mathscr{I}\in\mathscr{E}'} z_{\mathscr{I}}^{\text{isl, rel}}\,. \tag{10}$$

One can exploit this property by bounding the variable σ incorporated in the objective of the MP from below. Whenever a new MP solution is calculated, a set packing problem is solved for the cut generation (CG), which determines a family of disjoint islands $\mathscr{E}^* \subseteq \mathscr{E}$, maximizing the RHS of Equation (10):

$$\begin{aligned} \max_{\mathscr{E}'\subseteq\mathscr{E}} \quad & \sum_{\mathscr{I}\in\mathscr{E}'} z_{\mathscr{I}}^{\text{isl, rel}} \\ \text{s.t.} \quad & \mathscr{I}\cap\mathscr{J} = \emptyset \quad \forall \mathscr{I},\mathscr{J}\in\mathscr{E}' : \mathscr{I}\neq\mathscr{J} \\ & \mathscr{I} \textit{ active for } x^* \quad \forall \mathscr{I}\in\mathscr{E}' \end{aligned} \tag{CG}$$

It is straight-forward to reformulate this problem into a binary linear program, which can be solved with standard software. The optimal value of Problem (CG) is compared to the current value σ^* of the master variable σ. In case the optimal value is greater than σ^*, the constraint

$$\sigma \geq \sum_{\mathscr{I}\in\mathscr{E}^*} \sigma_{\mathscr{I}} \tag{11}$$

is added to the MP, cutting off the current solution. The MP is solved again. In case σ is equal to the optimal value, additional islands will be generated.

Algorithm 4 *Island Decomposition* Algorithm

1: **INPUT**: Overall problem instance of Model (3)
2: **OUTPUT**: Optimal solution for the instance of Model (3)
3: **procedure** DECOMPOSITION(3)
4: Split problem (3) into (MP) and (SP)
5: Initialize float $z^{\text{SP}} \leftarrow \infty$, $\sigma^* \leftarrow 0$
6: Initialize (SC)-island pool $\mathscr{E}_{(\text{SC})} \leftarrow$ all SC Islands $\{(k,i,j)\}$, $x_{k,(i,j)}$ in (MP)
7: Initialize island pool $\mathscr{E} = \mathscr{E}_{(\text{SC})}$ and (SF)-island pool $\mathscr{E}_{(\text{SF})} = \{\}$
8: Solve MP, solve SP based on MP solution $\rightarrow$ feasible solution of the integrated Model (3)
9: **do** ▷ outer loop
10: **do** ▷ middle loop
11: **do** ▷ inner loop
12: $\Omega, \mathscr{F}, x^*, \sigma^* \leftarrow$ Solve MP
13: $\mathscr{E}^* \leftarrow$ Solve Model (CG) depending on x^*
14: Add Cut (11) depending on $\mathscr{E}^*$ to MP
15: **while** $\sigma^* < \sum_{\mathscr{J} \in \mathscr{E}^*} z^{\text{isl, rel}}_{\mathscr{J}}$
16: $\mathscr{E}^{new}_{(\text{SF})} \leftarrow \{\text{GEN\_SF(IC-SF, } (k,i,j), \mathscr{F}_k, \mathscr{E}_{(\text{SC})}\text{): } (k,i,j) \in \Omega\} \setminus \mathscr{E}$
17: Add $\mathscr{E}^{new}_{(\text{SF})}$ to $\mathscr{E}_{(\text{SF})}$ and $\mathscr{E}$
18: **while** $\mathscr{E}^{new}_{(\text{SF})} \neq \emptyset$
19: SP *, $z^{\text{SP}} \leftarrow$ Solve SP
20: $\mathscr{E}^{new}_{(\text{MF})} \leftarrow \{\text{GEN\_MF(IC-MF, } \Omega, \text{ SP }^*, \mathscr{E}_{(\text{SF})}, B\text{): } B \in \mathscr{E}_{(\text{SF})} \text{ with } B \subseteq \Omega\} \setminus \mathscr{E}$
21: $\mathscr{E}^{new}_{(\text{MS})} \leftarrow \{(\Omega, z^{\text{SP}})\} \setminus \mathscr{E}$ ▷ add (MS)-island
22: Add $\mathscr{E}^{new}_{(\text{MF})}$ and $\mathscr{E}^{new}_{(\text{MS})}$ to $\mathscr{E}$
23: **while** $\sigma^* \neq z^{\text{SP}}$
24: **return** Ω, SP *
25: **end procedure**

4.5. **The overall *Island Decomposition* algorithm.** In this section the overall Algorithm 4 is set up. It starts with splitting the complete problem in MP and SP in line 4 as described above. Subsequently, MP as well as SP are solved once, to determine a first feasible solution as reference in line 8. After this step all (SC)-islands are calculated and initialize the island pool $\mathscr{E}$. The decomposition approach takes place in the following lines running through three loops are passed through in a nested way:

Inner Loop (Lines 11-Line 15). The inner loop is executed most frequently. In each iteration the MP is solved and new cuts are added by solving Model (CG) depending on the MP-solution and the current island pool $\mathscr{E}^*$. The inner loop is exited as soon as the newly found cuts do not lead to higher turnaround costs in the MP, which are represented by σ^*.

Middle Loop (Lines 10-Line 18). After the inner loop possibly changed the current MP solution, the middle loop generates new (SF)-islands and adds them to the (SF)-island pool as well as to the overall island pool. The middle loop is exited, if there are no new (SF)-islands found in an iteration.

Outer Loop (Lines 9-Line 23). Solves the full SP. The obtained SP solution is used to calculate new (MF)-islands and (MS)-islands and the corresponding costs. The results are added to the overall island pool. The outer loop terminates when the value of σ^* equals the real observed SP costs, which equal z^{SP}.

When the outer loop is left, the optimal solution of Model (3) is calculated. The entire process is also graphically explained in Figure 6. The most important point about Algorithm 4 is that outer loop iterations

are reduced drastically, since as much cost information as possible can already be obtained by combining (SC)-islands, (SF)-islands and (MF)-islands.

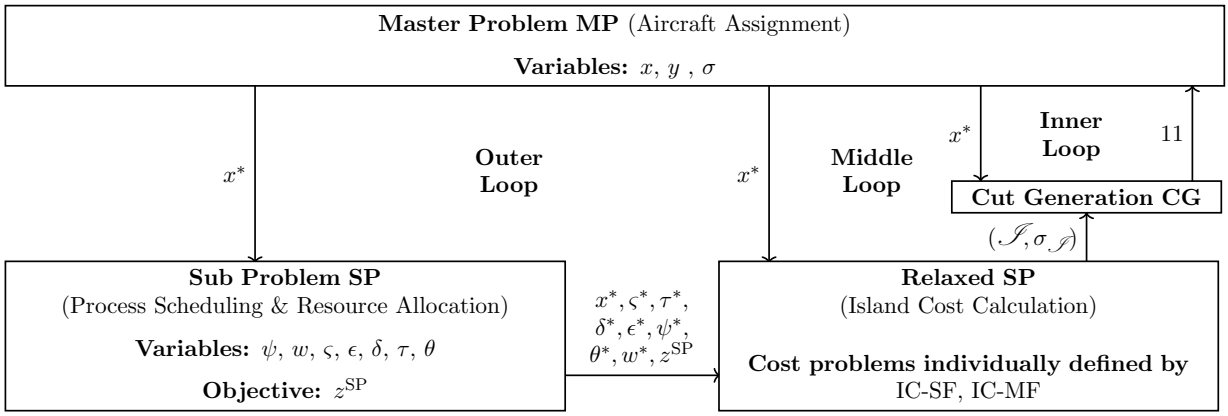

FIGURE 6. Scheme of the procedure for *Island Decomposition* using relaxed versions of the SP to calculate islands and island costs

We now prove that the proposed algorithm terminates after a finite number of steps and computes the optimal solution of the initial problem.

Theorem 4.6. *Algorithm 4 terminates after a finite number of steps and returns an optimal solution for the integrated tail assignment and ground process management problem* (3)*.*

Proof. The Aircraft Assignment master problem (MP) has a finite number of solutions. If the outer loop is run twice and the same master and sub solution are generated in each iteration, then the algorithm terminates, since in the first iteration the (MS)-island cut has been added to the MP. Due to Observation 4.3 it holds that in every other inner or middle loop iteration when this solution has been the current master solution, the SP cost bound has been at most the cost value of the (MS)-island. Hence, in the second outer loop iteration with the solution, the condition $\sigma^* = z^{\mathrm{SP}}$ is fulfilled and the algorithm terminates.

An analogous argument holds for the middle loop, which generates (SF)-islands. This procedure is uniquely determined by the current MP solution. Hence, when there are two iterations with the same master solution, the (SF)-islands corresponding to the island are already contained in the island pool $\mathscr{E}$. Therefore the middle loop is executed a finite number of times. The same argument holds for the inner loop regarding repetitive solutions of the MP. We have shown that the algorithm terminates after a finite number of steps with a feasible solution of the integrated problem.

In order to show its optimality, we consider x^* the solution of the master problem which is returned by the algorithm. We assume a solution x' of the master problem exists with $z^{\mathrm{MP}}(x') + z^{\mathrm{SP}}(x') < z^{\mathrm{MP}}(x^*) + z^{\mathrm{SP}}(x^*)$. Since the MP in line 12 has been solved to optimality, we can derive the existence of a constraint in the MP which enforces $\sigma \geq z^{\mathrm{SP}}(x')$. Since Observation 4.3 together with Observation 4.5 assure that the GENERATE_X_ISLANDS routines deliver lower bounds for the true island costs, and Lemma 4.4 assures that whenever this is the case, the inner loop only adds constraints to the MP which have the property that σ underestimates the value of z^{SP}, this is a contradiction.

Hence, the solution delivered by Algorithm 4 is optimal, which is exactly the required result. □

In the following section we prepare our computational study by introducing how we derive different instance sets using real-world flight schedules. In addition, we propose two benchmark algorithms which are used to evaluate the *Island Decomposition* approach.

5. Baseline Algorithms and Instance Sets

Since the model presented is a novel model, this chapter will focus on two important aspects: First, it will be shown how the instances for the subsequent computational study are generated. Afterwards we present two benchmark algorithms, which can also be used to solve the problem.

5.1. **Generating Instance Sets.** All of our instances are based on a real summer flight plan of a major German airline. However, this plan includes the flights of a whole week of all countries that are served. Our problem is more suitable for monitoring individual days in certain areas, since crucial turnaround decisions do not have to be planned for all days and only take place at hub-airports. In order to generate adequate instances, we rely on the common large base schedule, also containing information regarding the available fleets, from which individual sub-instances are derived. To avoid unrealistic instances, we first reduce the plan to a certain selected set of flights and airports, which must contain at least one hub.

We have reduced the turnaround sequence to four representative steps: De-Boarding, Catering, Cleaning and Boarding, containing two turnaround steps which require the same resource as well as sequential and parallel turnaround processes. Hence, the reduction of turnaround processes does not influence the structure or optimization behavior of the solution algorithms, but limits the computational effort.

We created different instances by setting the number of flights and aircraft within the random intervals given in Table 6. Feasible solutions utilize the whole aircraft fleet while covering all flights of the given schedule. The ratio between flights and aircraft and the structure of each schedule should lead to adequate, but relatively tight time windows between arrivals and departures of subsequently executed flight legs.

To generate this kind of instances, a reduced tail assignment problem is solved beforehand. The problem is defined on the base schedule. Its surrogate objective function minimizes the number of used aircraft, including a randomly given number of flight legs. The assignment costs for each fleet-flight pair were selected arbitrarily as well to be able to vary the instances. The solution yields all flights and aircraft which will be used in our instance from now on. Based on the selected connections of this reduced problem solution and the resulting ground times, turnaround processes were timed by splitting the overall ground time per connection. To provoke delays which result in interesting instances, the critical path of the turnaround process chain is designed not to fit entirely into the corresponding ground time interval on some connections. Additionally, a random number of resources per process is selected at the hub-airport. This number is chosen to provoke resource shortages at some points for the same reason. Thus it has to be drastically lower than the number of aircraft used in order to exclude situations where the resource limitation is not restrictive. All parameters and quantities listed in the model are added based on random intervals. All assignment decisions of this preliminary optimization procedure are discarded afterwards.

Instance Set	**I1**	**I2**
# Instances	50	4
# Flights	38-102	150-300
# Airports	3-8	13-24
# Aircraft	5-12	18-37
Ground interval	30-80 minutes	
Process duration factor	1.01 - 1.24	1.1-1.35
Resource factor	0.25 - 0.5	

Table 6. Characteristics of I1 (mean value study) and I2 (larger instances)

For our computational results we generated two instance sets I1 and I2. Instance set I1 has tightly timed process steps, which partly exceed the possible ground times and has very few resources on ground available as is usual in practice. It consists out of 50 indidual instances in order to perform a mean value study to analyze if our proposed approach delivers high-quality solutions regarding instance sizes of around 100 flights. Instance set I2 only consists of four instances. The smallest instance includes 150 flights and the instance size

is raised by steps of 50 flights, so that the biggest instance includes 300 flights daily. The examination of I2 is done instance by instance. The limits of computational tractability, depending on instance size and different solution procedures are to be analyzed. All characteristics per instance set are summarized in Table 6. For I1, instances were generated as a function of a random number of flights. The number of airports, aircraft were then adjusted as described to generate realistic schedule data, i.e., without too much time between two consecutive flights and just enough aircraft to execute all flights. For I2, a deterministic number of flights were used to define the instance, i.e., $\{150, 200, 250, 300\}$. The number of airports and aircraft is again based on the same procedure as in I1. In the calculated reference solution of the Tail Assignment problem only connections with ground times intervals between 30 and 80 minutes can be selected in order to construct tight flight schedules. Table 6 further characterizes both instances based on the so called *process duration factor* and the *resource factor*. For each of the instances, the time necessary to carry out turnaround processes is constructed such that the turnaround duration slightly exceeds the ground time between two flights in a reference solution. Hence, the duration of the the entire turnaround process is calculated by the product of the *process duration factor* and with the time spent in the reference solution on the ground. This value was chosen to replicate external factors on the basis of which delays occur. The process duration factor for I1 is set to values, which forces delay situations to occur, while the factors for I2 are even set to higher values in order to represent a larger variation of the instance settings. The *resource factor* affects the number of resources available on each ground at the hub-airport and represents the ratio between flight strings of an instance and the number of resources per type on ground.

5.2. **Benchmark algorithms.** In order to evaluate the solution quality of the Island decomposition algorithm, we present three further solution approaches that can be used to solve the problem.

We first tried to compare the solution characteristics of the decomposition approach with solving the overall model (3). Unfortunately, it turned out that the integrated solving of the problem spent a large amount of time on finding only a feasible solution to the problem in the first place.

Thus we concluded that it is beneficial to generate a feasible start solution by first solving Model (MP) first, fixing all the variables to the obtained value in the solution and solve (SP) afterwards. This procedure equals the start procedure in Line 8 of the *Island Decomposition* Algorithm 4. As discussed in the previous sections, this always leads to a feasible solution of the entire problem. This feasible solution is then used as a start solution for solving Problem (3) directly with a standard MIP-Solver. The procedure of improving this start solution is called *Benchmark Waterfall*.

We also want to compare our approach against an often used decomposition approach based on the use of standard Benders decomposition with sub problems neglecting integrality constraints. Many authors apply this standard version in order to calculate a probably good solution for the master problem using only the continuous relaxation of the sub problem. The behaviour of the solution calculated by the approach can be simulated by solving the integrated Model (3) while relaxing the integrality constraints of all variables included in the (SP). Nevertheless, the obtained solution of this problem is in general not feasible for the Problem (3), since it crucially depends on the integrality of the SP. Therefore, to obtain a comparable solution, we calculate the resulting integral solution by fixing the master variables of the entire Problem (3), reintroduce integrality constraints and calculate the SP solution. Since we want to compare the solution of the Benders approach, we optimized the problem with the relaxed SP for one hour and used additional 15 minutes in order to obtain the corresponding optimal integral solution. We only compare the resulting end solution with the other approaches, thus resulting an approach, which is not exact. We call this approach *Benchmark Benders*.

As a third benchmark algorithm, we want to compare a kind of classical lower-bound function method introduced in Laporte and Louveaux (1993), which our island decomposition approach is based on. For

this purpose, we again use our island decomposition approach, where all (SF)-islands and (MF)-islands are deactivated. The (MS)-islands lead to cuts in the MP, which can be considered as lower bound functions presented in the literature and thus bound the MP cost variable σ depending on the whole MP-solution from below. The main difference is that the optimality cuts in our MP do not have to be designed as a function of all binary variables, but depend only on the variables that take the value 1 in the current solution. This is due to the cover constraint (1d), which ensures that one flight is executed by exactly one aircraft and thus reduces the number of variables to be considered enormously. This approach is referred to as *Benchmark LBF* in the following.

6. Computational Results

The *Island Decomposition* approach is compared with the benchmark algorithms that are evaluated for two described instance sets. Our model is intended to adjust parts of existing plans on the day before operations, so that currently known ground conditions can be optimally planned. We first investigate with a mean value study of I1 whether the *Island Decomposition* can find high-quality solutions in an acceptable time. Furthermore, we check whether better solutions can be found faster than using the benchmark algorithms. We believe, that the decomposition approach be able to find tight lower bounds during the solution process due to the (SC)-islands, (SF)-islands and (MF)-islands.

In the further setting with instance set I2 we will investigate if the decomposition approach still gives good results if the the instance sizes are increased in terms of considered flights.

All algorithms are implemented in Python 3.7.1 and all MIPs are solved with Gurobi 9.1.1, see Gurobi Optimization (2020). The computations were executed on a 4-core machine with a Xeon E3-1240 v6 CPU (3.7GHz base frequency) and 32 GB RAM.

In the following we present the results of the computational study, containing comparison of runtimes, solution quality, performance profiles and dependencies of the solution behavior on instance characteristics.

6.1. Results for Instance Set I1.

In the following, we will analyze the results of the computations which deal with solving the instances from I1 using *Island Decomposition* and the benchmark algorithms listed in Section 5.

We first focus on a comparison between different solution quality indicator:

Classical gap: The classical optimality gap of an algorithm alg after t seconds is calculated using the following formula

$$\frac{\text{inc}(\text{alg}, t) - \text{lb}(\text{alg}, t)}{\text{inc}(\text{alg}, t)}, \tag{12}$$

whereby $\text{inc}(\text{alg}, t)$ denotes the best incumbent value found by algorithm alg after t seconds, and $\text{lb}(\text{alg}, t)$ corresponds to the best lower bound for the objective value found by algorithm alg after t seconds. For algorithm *Benchmark Waterfall* the gap is calculated by gurobi itself. The lower bound for *Benchmark Benders* corresponds to the best lower bound found for the problem without integrality constraints in the SP, while the incumbent equals the objective value of the integral end solution. Regarding *Island Decomposition* and *Benchmark LBF*, the lower bound corresponds to the current MP objective, while the incumbent equals the value of the best integer feasible MP-SP solution found.

Best bound gap: This gap uses the best found lower bound over all algorithms at the end of the calculations instead of the current bound. Thus the gap reflects the actual quality of the solution in terms of the incumbent more accurately. It is computed using the formula

$$\frac{\text{inc}(\text{alg}, t) - \text{lb}^*}{\text{inc}(\text{alg}, t)}, \tag{13}$$

whereby lb^* equals the best found lower bound over all algorithms after one hour of computations.

Best incumbent gap: Instead of fixing the lower bound, for the calculation of this gap the incumbent is fixed to the best one found over all algorithms and times steps. It can be calulated using following formula:

$$\frac{\text{inc}^* - \text{lb}(\text{alg}, t)}{\text{inc}^*}, \tag{14}$$

whereby inc^* denotes the best found incumbent over time and solution approach.

Figure 7 shows the mean value of the classical gaps of all instances in I1 over the entire solution process measured in seconds. The mean gap for the decomposition approach is lower than the mean gap for all benchmark approaches except for *Benchmark Benders* at all points in time. The curve for the *Island Decomposition* drops heavily at the beginning of the calculations, since all (SC)-islands estimate the costs right from the start. In the further course, the mean gap decreases less heavily. Any steps in the curve indicate that some instances could be solved completely to optimality. The *Island Decomposition* approach yields a mean gap of just over 3% after one hour, whereas *Benchmark LBF* takes a value close to 9%. The conventional solution approach *Benchmark Waterfall* yields a gap of about 8%. One can observe that the *Benchmark Benders* approach starts with a better initial solution than all other approaches and remains on the same level all the time. This is due to the fact, that the first feasible solution is computed using an MP-solution which considers a continuous relaxation of the SP. After one hour the SP with integrality constraints is solved fixing the obtained MP solution (which better fits the SP) and thus results one lower objective value of the entire problem (see Section 5.2). Even though *Benchmark Benders* has a smaller initial gap, is outperformed by the *Island Decomposition* approach, which has a worse initial solution.

In order to work out why the gap of Island Decomposition is the lowest, we also plot the best lower bound ratio in Figure 8 and the best incumbent ratio in Figure 9.

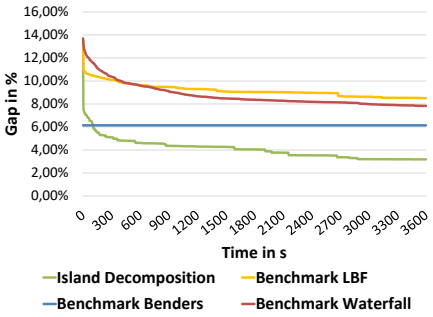

FIGURE 7. Mean classical gap

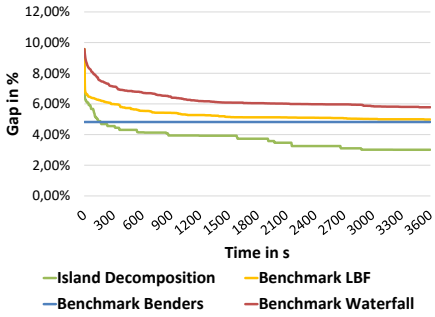

FIGURE 8. Best lower bound gap

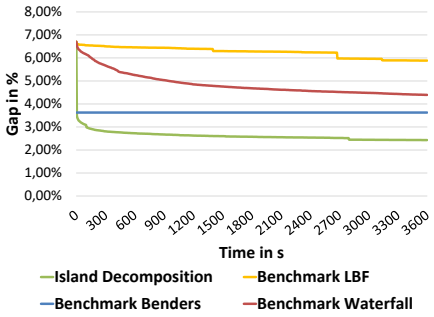

FIGURE 9. Best incumbent gap

Figure 8 shows the actual solution quality of the algorithms in terms of the calculated incumbent: It can be seen that the *Island Decomposition* produces the best solutions again on average from three minutes onwards and yields a mean lower bound gap of 3% after one hour. The *Benchmark Benders* comes second with around 5%, close to the result of *Benchmark LBF*. The latter approach comes with a smaller gap than *Benchmark Waterfall* with around 6%, which indicates, that the actual solutions obtained by *Benchmark LBF* yield better objective values, but fail to prove the distance to optimality due to poor lower bounds. This assessment is confirmed in Figure 9: Looking at the lower bound gap, one can see that the decomposition approach finds the tightest lower bounds (2.7%) very quickly, whereas *Benchmark LBF* yields a lower bound gap of 6.5%, which supports the fact that the (SC)-, (SF)- and (MF)-islands are crucial for the process of raising the σ values and thus the lower bound. The *Benchmark Waterfall* can continuously improve the lower bounds yielding a gap of 5% and *Benchmark Benders* is the second best approach with 4%.

We derive from Figure 8 and 9 that the *Island Decomposition* outperforms all benchmark algorithms in terms of their mean lower bounds as well as their actual incumbent values regarding I1. Exploiting the

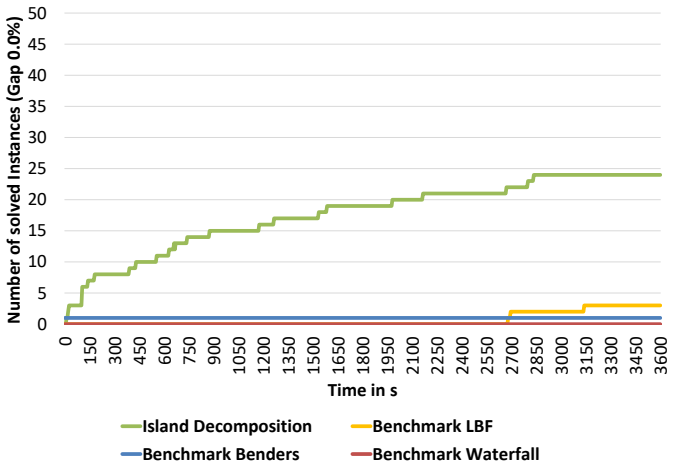

FIGURE 10. Solved Instances

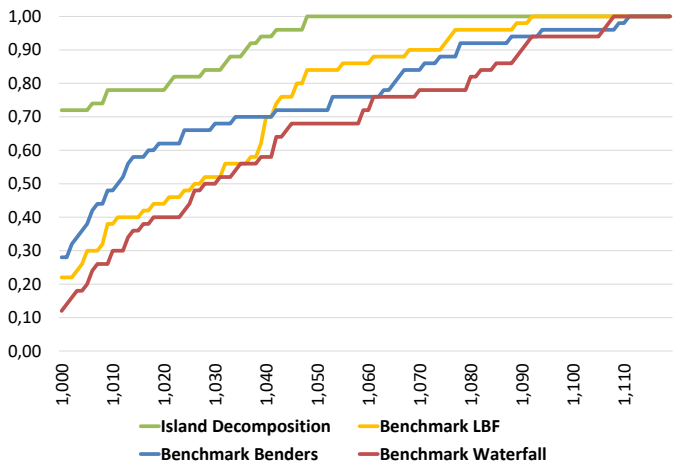

FIGURE 11. Performance Plot

problem structure, the decomposition approach can quickly approximate many costs of the SP in the MP, providing good lower bounds on the total cost. Moreover, based on the information MP solutions can be selected that reduce the total cost. One can also conclude, that the *Benchmark Benders* also performs well. We assume that the reason for that behavior ist, that the relaxation of integrality in the SP is still capable of approximating the real SP costs. Nevertheless, the approach is not exact can only prove the optimality in the case, that the SP solution of the relaxed SP is integral, which is not guaranteed.

All solution approaches except of *Benchmark Benders* are exact solution approaches which deliver the optimal solution in theory if enough time is available. Hence, we show how many of the instances could actually be solved to optimality within the given time. Figure 10 plots the number of instances solved to optimality over time for each solution approach.

It can be seen that the *Island Decomposition* approach was able to solve 24 of 50 instances during the whole time interval. On the other end *Benchmark Waterfall* was not able to solve a single instance to optimality. *Benchmark Benders* solves one instance to optimality, which is due to the fact, that the solution of the relaxed subproblem yields integral values for all contained binary and integral variables. The second best approach here is *Benchmark LBF* with three solved instances. It took a lot of time to prove their optimality (at around 45 minutes), since the lower bounds of the approach are not very tight. This comparison clearly shows that the strong benefit of our approach is the capability of proving optimality of the obtained results. In this category the *Island Decomposition* algorithm is clearly dominating the benchmark algorithms.

To give a better intuition on the gap distribution and to compare the algorithms more directly, we present a performance plot in Figure 11. It is based on the best incumbent value the algorithms found after 60 minutes of computation, which are compared to each other. Thereby, for each instance at the end of the calculation, its incumbent is compared per algorithm. This is divided by the best incumbent found. Thus, the algorithm that found the best solution receives a value of 1 for this instance and all those that had a higher incumbent value receive a value greater than 1. In Figure 11, for each instance on each x-axis value, the percentage of instances that have a value less than or equal to the x-axis value is plotted. Therefore the value at 1 represents the number of instances, for which the respective algorithm was able to find the best found solution. In the case that more than one algorithm was able to find the best possible incumbent, the values sum up to more than 100%. Figure 11 shows that the *Island Decomposition* approach provides the best incumbent value for 76% of all instances and finds a good incumbent solution which is within approximately 1-1.05% of the incumbent value the best of the three algorithms found for almost all instances. *Benchmark Benders* comes in second place for ratios under 1.04, since it provides many high quality solutions with a low classical gap. For higher values, *Benchmark LBF* performs better, since it is able to find most of its solutions with a gap in the

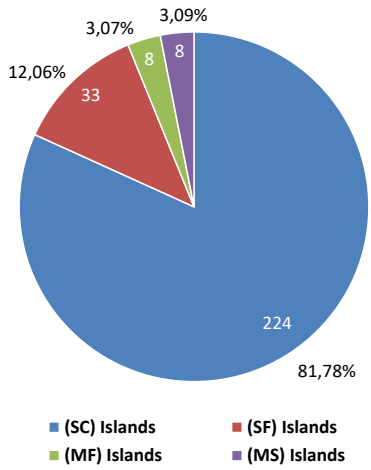

FIGURE 12. Mean proportions of different island types after 3600s

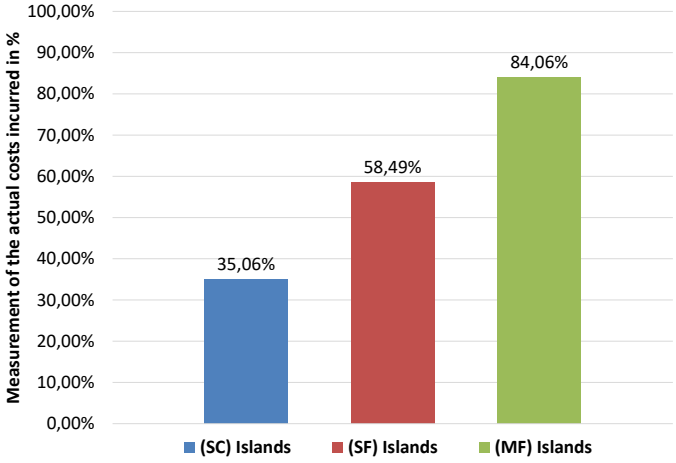

FIGURE 13. Percentage of costs in the SP that could be covered by each island type

middle field of the solutions, whereas *Benchmark Benders* finds more solutions with a higher gap. *Benchmark Waterfall* is dominated by all other algorithms for nearly all x-axis values. The worst gaps for a single solution found are produced by *Benchmark Benders*, which is why it is the last one to reach the value 100%.

We have now compared the results between the different benchmark approaches and now want to focus on how the *Island Decomposition* behaves during the solution process in order to explain the obtained values. Therefore the proportions of different island types which were generated during the solution process are shown in Figure 12.

Most of the generated islands are (SC)-islands (81.78%). On average 224 (SC)-islands were detected per instance, which means that on 224 aircraft-connection combinations costs are incurred independent of the other selected flight connections. As expected, The calculation per (SC)-island took only a fraction of a second, so that all of them (and their costs) were usually calculated in less than 20 seconds. Thus they are created at the beginning of each calculation, regardless of whether the connection to which the island corresponds has ever been assigned or not. The number of (SC)-islands predominates here, since islands are generated for all single connections at the beginning of the calculation regardless of the MP solutions found. All other island types, which are generated dynamically, just depend on relevant flight strings, which are selected in the MP solution. Therefore and because not all possible flight strings are useful regarding the objective function of the MP, the number of dynamically generated island types is significantly lower within one hour of computation.

The (SF)-islands are the dominant type of islands calculated dynamically by the algorithm which results in 33 islands on average and a proportion of 12.06%. The determination and cost calculation of a (SF)-island also took only less than 5 seconds on average. The proportion of (MF)-islands (3.07%) is similar to the one of (MS)-islands (3.09%), which corresponds to 8 islands on average for each type. The calculation of these island types took much longer and varies heavily depending on the given SP structure: The calculation of one (MS)-island took between 30 seconds and 10 minutes. Based on a given (MS)-island, the generation of a (MF)-island also took between 20 seconds and 6 minutes.

Even though the percentage of dynamically generated islands may seem small, Figure 13 illustrates that the effect of these islands is crucial to the decomposition approach: The plot shows what percentage of the actual costs generated in the SP could be covered on average by the respective island type in the mean. Therefore, after each iteration in which the whole SP was solved, we considered the current MP-solution: For each island type we calculated the ratio between the value, which σ takes regarding the island type in this MP-solution, and the SP objective value. This corresponds to a comparison of the actual SP costs arising and the subsequent considered SP-costs σ in the MP regarding the island cuts. One has to mention, that in the Figure 13,

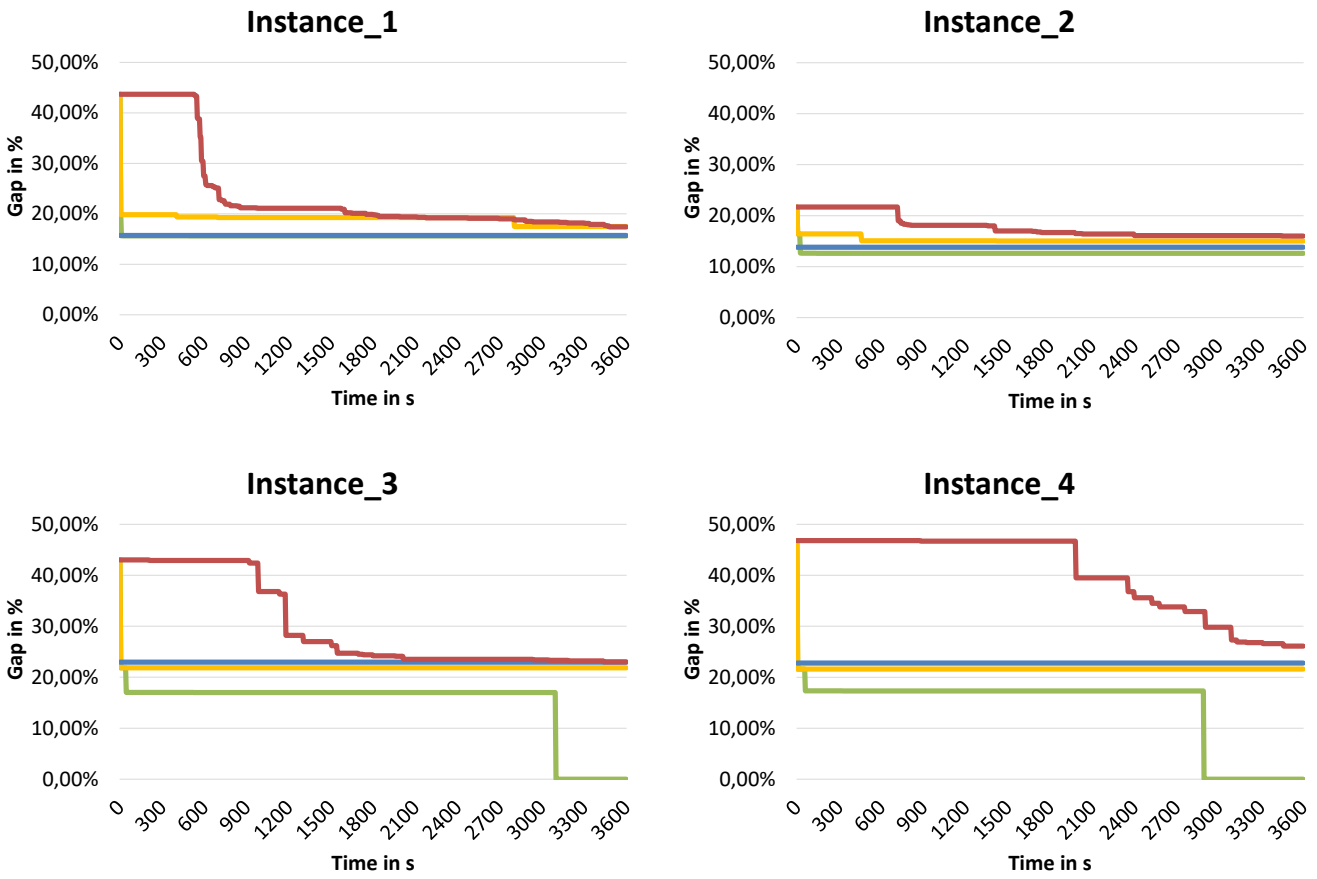

FIGURE 14. Classical lower bound gap per instance and solution approach.

the (SC)-island-value represents the value of σ, when only (SC)-islands are considered, the (SF)-island-value considers (SC)-islands and (SF)-islands and the (MF)-island-value takes into account all island types expect (MS)-islands. The (MS)-island-value is omitted since the (MS)-island corresponding to the MP-solution was already calculated and thus the value always equals 100%. We can see that the (SC)-islands cover around 35% of the costs, while additionally considering (SF)-islands yields a value of almost 60%. When the (MF)-islands are also used to generate the optimality cuts for the MP, it is possible to explain 84% percent of all arising costs in the SP. The remaining 16% can only be covered with the use of (MS)-islands, which is the reason why they are necessary in order to make the *Island Decomposition* Algorithm 4 an exact algorithm.

6.2. **Results for Instance Set I2.** Since the mean value study on instance set I1 proves the island decomposition approach to outperform comparable benchmark procedures for Problem (3), we now want to focus on testing the resilience of the algorithm in the case the considered instances are of larger size.

Figure 14 shows the course of the classical gaps for each of the four instances in I2. This figure is supported by Table 7, which shows the best found lower bound, objective and classical gap for *Island Decomposition* and all benchmark approaches. The values which are highlighted green represent the best values over all approaches. In the case there are two marked values of the same kind, this indicates, that the same solution was obtained. With respect to the classical gap, it can be seen, just as in I1, that benchmark decomposition yields the best results. This is not too obvious for instances 1 and 2, since the gap of the island decomposition approach is only slightly smaller than the gaps of the benchmark approaches. For instances 3 and 4, however, one can even see that the gap drops to 0% after about 50 minutes, which means that both instances can be solved to optimality by the *Island Decomposition* approach. Looking at the lower bounds in Table 7, it becomes clear that these good results are due to the superior calculation of the lower bounds.

Instance	#Flights	*Island Decomposition*			*Benchmark LBF*		
		LB	**OBJ**	**Gap**	**LB**	**OBJ**	**Gap**
1	150	1.516.493	1.797.111	15.61%	1.440.911	1.746.111	17.48%
2	200	1.970.007	2.254.093	12.60%	1.883.793	2.217.193	15.04%
3	250	3.039.360	3.039.360	0.00%	2.375.210	3.039.360	21.85%
4	300	3.764.333	3.764.333	0.00%	2.950.883	3.764.333	21.61%
Instance	**#Flights**	***Benchmark Waterfall***			***Benchmark Benders***		
		LB	**OBJ**	**Gap**	**LB**	**OBJ**	**Gap**
1	150	1.442.734	1.745.764	17.36%	1.441.537	1.710.146	15.71%
2	200	1.885.365	2.243.441	15.96%	1.884.128	2.185.747	13.80%
3	250	2.379.445	3.086.860	22.92%	2.374.946	3.082.960	22.97%
4	300	2.955.423	3.998.176	26.08%	2.950.894	3.821.890	22.79%

TABLE 7. Comparison of the best found lower bound, objective value and classical gap after one hour for *Benchmark Waterfall* and all benchmark approaches. The best values are highlighted in green.

Instance	#Flights	*Island Decomposition*				*Benchmark LBF*			
		(SC)	**(SF)**	**(MF)**	**(MS)**	**(SC)**	**(SF)**	**(MF)**	**(MS)**
1	150	1437	26	0	1	0	0	0	24
2	200	1955	34	0	1	0	0	0	5
3	250	3660	42	0	1	0	0	0	5
4	300	4449	44	0	1	0	0	0	3

TABLE 8. Number of generated islands per island type during the *Island Decomposition* and *Benchmark LBF*.

All benchmark approaches, on the other hand, deliver the same poor lower bounds, which is especially evident in instances 3 and 4. Here, the *Benchmark LBF* even manages to determine the optimal objective function value, but ends with a classical gap of more than 20 percent, since several subproblem runs are still necessary to prove the optimality. Looking at the objective value, *Benchmark Benders* delivers the very best results for instances 1 and 2. Here, too, the lower bounds are not sufficient to calculate lower gaps than the *Island Decomposition* approach.

Benchmark Waterfall does not deliver the best result in terms of any of the metrics for any instance and takes a long time to even push the gap to a comparable level to the other approaches. This becomes especially clear with the largest instance 4, where the end gap is over 3% higher than with all other methods.

In summary, the following can be said about gap development: *Benchmark Waterfall* takes the longest time to deliver comparable results at all. If the instance becomes too complex, the gap can no longer keep up with those of the other approaches. *Benchmark Benders* delivers very good results for smaller instances, which is due to the low objective value. However, *Benchmark Benders* is inferior to *Benchmark LBF* for larger instances, which indicates that the general decomposition structure of our approach is appropriate to obtain good results. *Benchmark LBF* quickly delivers low gaps (*Benchmark Benders* results refer to the final result), but fails to prove optimality. *Island Decomposition* is ahead for all instances, but can show its qualities especially for large instances.

It is interesting to observe, that the solved instances represent the biggest instances in terms of the number of included flights. This makes clear, that the efficiency of the tested approaches is highly depending on the individual instance structure. Whenever very few solutions of the MP have to be considered, since they dominate all other solutions, the *Benchmark LBF* delivers promising results, since only a small number of (MS)-islands will lead to the overall optimal solution. Whenever the situation is more complicated, additional good lower bounds obtained by adding (SC)-islands, (SF)-islands and (MF)-islands are necessary to prove optimality.

In order to verify the theses, a closer look is taken at the number of generated islands for *Island Decomposition* and *Benchmark LBF*. For a, we can see that only one (MS)-island was calculated for all instances. This corresponds to island calculated for the initial solution of the SP. Since no further (MS)-islands are calculated, no (MF)-islands are calculated. For (SC)-islands and (SF)-islands, we can observe an increase in quantity with increasing instance size. It is remarkable that only with these island types the *Benchmark LBF* is able to obtain the optimal objective value.

Benchmark LBF only uses (MS)-islands in order to improve the solution quality. It can be observed, that except for the first small instance, the number of (MS)-islands drops to be lower than five. Especially for instance 4, the SP seems to be hard to solve, since the SP is solved only three times during one hour of computations.

Summary. All in all, our computational section shows that the *Island Decomposition* can achieve very good results for the tested instances. The mean value study for I1, where very few turnaround resources are available, provides the proof of concept. Here, the *Island Decomposition* approach yields the best results, especially in realistic scenarios involving delay and resource bottlenecks. This is essentially based on the ability to calculate very tight lower bounds. As a result, far more instances could be solved to optimality compared to the benchmark procedures. The calculations for I2 also support the same conclusion, since optimal results can only be proven by the *Island Decomposition* approach. This indicates that the approach we chose to solve the problem was reasonable. Especially for the small instances in I2 it became clear that *Benchmark Benders* also provides very good objective values, but fails to prove their quality. This problem is of structural nature in the approach as long as the SP solutions are not random integers, which was not the case in any of our test runs. *Benchmark Waterfall* provided sufficient evidence that simply using a MIP solver to solve the problem yields the worst results in most of the cases.

7. Conclusion and Outlook

This paper has provided a new mathematical model to handle two operational tasks of airlines simultaneously: Aircraft Assignment and Turnaround Handling. Moreover we demonstrated that this model is not straight-forwardly solvable with standard approaches for reasonable problem sizes. Hence we developed a combinatorial decomposition approach, mathematically proved its capability of solving the model to optimality and presented an extensive computational study, which proves its effectiveness in practice.

Our future research will aim at integrating additional problem aspects to make the presented model more realistic and therefore attractive for practitioners.

Another research direction is to make the decomposition approach more efficient, for example by incorporating some of the smaller island types directly into the master model. We presume that this could lead to a large speed up while pertaining the property of the algorithm to be an exact one.

We believe that the introduced decomposition framework is not only applicable to the presented integrated tail assignment and ground process management model but can also be transferred to other mathematical problems. The condition is that these problems can be split into master and sub problem such that the prior is combinatorial and the latter highly depends on its integer nature.

Acknowledgements

This research has been conducted in the framework of the research project OPs-TIMAL, financed by the German Federal Ministry of Economic Affairs and Energy (BMWi).

References

Abara, J. (1989). Applying integer linear programming to the fleet assignment problem. *Interfaces*, 19(4):20–28.

Barnhart, C., Boland, N. L., Clarke, L. W., Johnson, E. L., Nemhauser, G. L., and Shenoi, R. G. (1998). Flight string models for aircraft fleeting and routing. *Transportation science*, 32(3):208–220.

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische mathematik*, 4(1):238–252.

Caceres-Cruz, J., Arias, P., Guimarans, D., Riera, D., and Juan, A. A. (2014). Rich vehicle routing problem: Survey. *ACM Computing Surveys (CSUR)*, 47(2):1–28.

Carøe, C. C. and Tind, J. (1998). L-shaped decomposition of two-stage stochastic programs with integer recourse. *Mathematical Programming*, 83(1):451–464.

Codato, G. and Fischetti, M. (2004). Combinatorial benders' cuts. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 178–195. Springer.

Cordeau, J.-F., Stojković, G., Soumis, F., and Desrosiers, J. (2001). Benders decomposition for simultaneous aircraft routing and crew scheduling. *Transportation science*, 35(4):375–388.

Desaulniers, G., Desrosiers, J., Dumas, Y., Solomon, M. M., and Soumis, F. (1997). Daily aircraft routing and scheduling. *Management Science*, 43(6):841–855.

Dunbar, M., Froyland, G., and Wu, C.-L. (2012). Robust airline schedule planning: Minimizing propagated delay in an integrated routing and crewing framework. *Transportation Science*, 46(2):204–216.

Eltoukhy, A. E., Wang, Z., Chan, F. T., Chung, S., Ma, H.-L., and Wang, X. (2019). Robust aircraft maintenance routing problem using a turn-around time reduction approach. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 50(12):4919–4932.

Evler, J., Asadi, E., Preis, H., and Fricke, H. (2018). Stochastic control of turnarounds at hub-airports. *Proceedings of the Eighth SESAR Innovation Days, Salzburg, Austria*, pages 3–7.

Fügenschuh, A. (2006). The vehicle routing problem with coupled time windows. *Central European Journal of Operations Research*, 14(2):157–176.

Glomb, L., Liers, F., and Roesel, F. (2020). A rolling-horizon approach for multi-period optimization.

Gmira, M., Gendreau, M., Lodi, A., and Potvin, J.-Y. (2021). Tabu search for the time-dependent vehicle routing problem with time windows on a road network. *European Journal of Operational Research*, 288(1):129–140.

Gopalan, R. and Talluri, K. T. (1998). The aircraft maintenance routing problem. *Operations Research*, 46(2):260–271.

Gurobi Optimization, L. (2020). Gurobi optimizer reference manual.

Hane, C. A., Barnhart, C., Johnson, E. L., Marsten, R. E., Nemhauser, G. L., and Sigismondi, G. (1995). The fleet assignment problem: Solving a large-scale integer program. *Mathematical Programming*, 70(1-3):211–232.

Hooker, J. N. and Ottosson, G. (2003). Logic-based benders decomposition. *Mathematical Programming*, 96(1):33–60.

Laporte, G. and Louveaux, F. V. (1993). The integer l-shaped method for stochastic integer programs with complete recourse. *Operations research letters*, 13(3):133–142.

Mercier, A., Cordeau, J.-F., and Soumis, F. (2005). A computational study of benders decomposition for the integrated aircraft routing and crew scheduling problem. *Computers & Operations Research*, 32(6):1451–1476.

Mercier, A. and Soumis, F. (2007). An integrated aircraft routing, crew scheduling and flight retiming model. *Computers & Operations Research*, 34(8):2251–2265.

Parmentier, A. and Meunier, F. (2020). Aircraft routing and crew pairing: updated algorithms at air france. *Omega*, 93:102073.

Salavati-Khoshghalb, M., Gendreau, M., Jabali, O., and Rei, W. (2019). An exact algorithm to solve the vehicle routing problem with stochastic demands under an optimal restocking policy. *European Journal of Operational Research*, 273(1):175–189.

Sandhu, R. and Klabjan, D. (2007). Integrated airline fleeting and crew-pairing decisions. *Operations Research*, 55(3):439–456.

Sarac, A., Batta, R., and Rump, C. M. (2006). A branch-and-price approach for operational aircraft maintenance routing. *European Journal of Operational Research*, 175(3):1850–1869.

Sherali, H. D., Bae, K.-H., and Haouari, M. (2013). A benders decomposition approach for an integrated airline schedule design and fleet assignment problem with flight retiming, schedule balance, and demand recapture. *Annals of Operations Research*, 210(1):213–244.

Sherali, H. D. and Fraticelli, B. M. (2002). A modification of benders' decomposition algorithm for discrete subproblems: An approach for stochastic programs with integer recourse. *Journal of Global Optimization*, 22(1):319–342.

Solomon, M. M. (1984). *Vehicle routing and scheduling with time window constraints: models and algorithms (heuristics)*. PhD thesis, University of Pennsylvania.

Stojković, M. and Soumis, F. (2001). An optimization model for the simultaneous operational flight and pilot scheduling problem. *Management Science*, 47(9):1290–1305.

Toth, P. and Vigo, D. (2014). *Vehicle routing: problems, methods, and applications*. SIAM.

Wu, C.-L. and Caves, R. E. (2004). Modelling and optimization of aircraft turnaround time at an airport. *Transportation Planning and Technology*, 27(1):47–66.

Yan, C., Barnhart, C., and Vaze, V. (2020). Choice-based airline schedule design and fleet assignment: A decomposition approach. *Available at SSRN 3513164*.

Zeighami, V., Saddoune, M., and Soumis, F. (2020). Alternating lagrangian decomposition for integrated airline crew scheduling problem. *European Journal of Operational Research*, 287(1):211–224.

8. Appendix

We want to present in this additional chapter an example which demonstrates the capability of the introduced mode, some proofs which have been omitted in the main manuscript and furthermore the algorithm details for the determination of the distinct island types. Since all algorithms are very technical and the content is explained in the main paper, the absence of the algorithms does not influence the understandability of the main manuscript.

8.1. **Model Capability.** In this section we will show which possibilities the presented modeling opens up and how it can be embedded in the application field of airline operations. The problem is not initially of strategic importance due to its size and can be applied for short-term planning. This can be useful one day before the day of operations, when one already expects dislocations at certain points of the schedule. This can be due to several reasons, like technical problems, temporary resource constraints at the hub-airport or propagated delays. The model capabilities will be discussed using a simple example, which is supported by the three Figures 15, 16 and 17. For the sake of simplification we assume that there exists only one turnaround process and there is only one resource at the hub-airport, which is needed in order to perform this process on each turnaround.

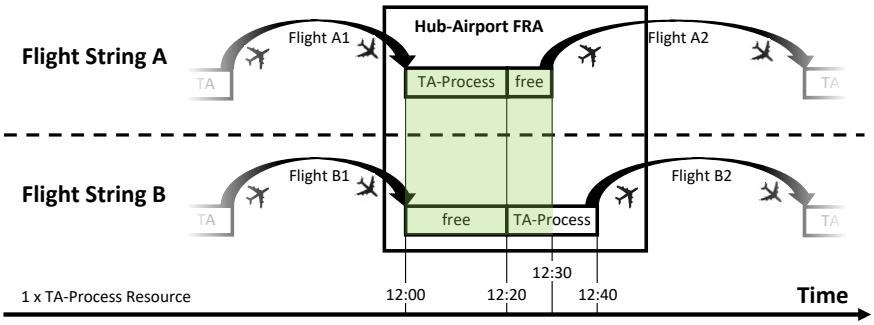

Figure 15. Initial situation of the example: All processes can be performed and no flight is delayed.

Figure 15 shows the initial situation: There exists two flight sequences (flight strings) A and B including two flights each. The two corresponding aircraft share a time-interval on ground from 12:00 to 12.30. Due to the subsequent departure times and the length of the TA-Processes, one resource on ground is sufficient to perform all tasks in time.

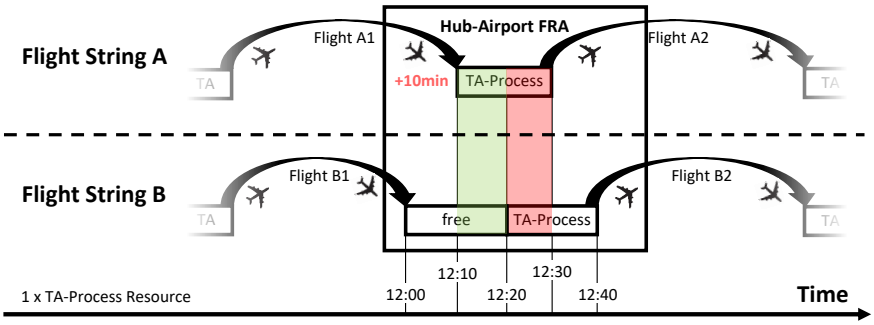

FIGURE 16. Initial disruption of the example: Flight A1 on Flight String A arrives 10 minutes later than planned. The subsequent turnaround process is delayed.

In Figure 16 the initial situation is disrupted by an arrival delay of 10 minutes regarding Flight A1. This leads to a total time of ground of exactly 20 minutes and forces the TA-Process to take place exactly during this time slot, since otherwise Flight A2 starts with delay. Hence, the TA-Process on Flight String B can not be executed during the given ground time, since the process can start earliest 10 minutes before the departure of Flight B2, but needs 20 minutes to be performed. The overlapping time drawn red in Figure 16 is caused by a resource bottleneck.

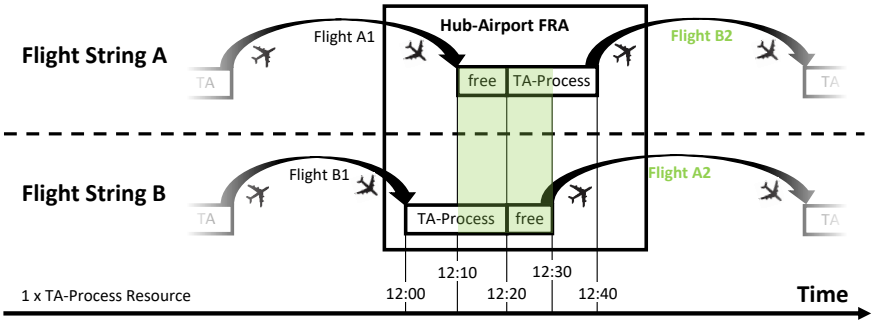

FIGURE 17. Solution which yields no delay propagation: Flight B2 is now executed on Flight String A. Hence, the TA-Process on Flight String A can take place later and makes it possible to execute the TA-Process on Flight String B in time.

The situation can not be solved without additional delay only considering turnaround measurements. Nevertheless, including the aircraft assignment yields a solution with no more delay on the subsequent flights, which is drawn in Figure 17. If Flight A2 and Flight B2 are swapped, which results in Flight B2 being a part of Flight String A and Flight A2 being part of Flight String B, the times on ground can be balanced in a way, that compensates the delay of Flight A1. The TA-Processes can be solved sequentially using only one resource neutralizing the resource bottleneck.

8.2. **Proof of Lemma 4.4.**

Proof. We prove first that the cardinality $|\{j \in L \mid \exists k \in K, i \in L, \mathscr{J} \in \mathscr{E} : (k,i,j) \in \mathscr{J}\}|$ is either 0 or 1. Assume it is not true, then there exist $(k,i,j) \neq (k',i',j)$ with $x^*_{k,(i,j)} = x^*_{k',(i',j)} = 1$. In case $k = k'$ this contradicts Constraint (1b), since the left side of the constraint evaluates to 2 and the right side is binary. In case $k \neq k'$, Constraint (1b) implies that $y^*_{k,j} = y^*_{k',j} = 1$. This contradicts Constraint (1d) since the left

side evaluates to $2 \neq 1$. Similar arguments apply to $|\{i \in L | \exists k \in K, j \in L, \mathscr{J} \in \mathscr{E} : (k,i,j) \in \mathscr{I}\}|$. Hence, rearranging the following sum is possible, since it is assured that no summand appears twice on the left side, whereby $\tau^{\mathscr{J}}, \psi^{\mathscr{J}}$ are part of the solutions of (IC) for every $\mathscr{J}$:

$$\begin{aligned}
\sum_{\mathscr{J}\in\mathscr{E}} z^{\text{isl}}_{\mathscr{J}} &= \sum_{\mathscr{J}\in\mathscr{E}} \sum_{(k,i,j)\in\mathscr{J}} (c^{\text{delay}}_{k,j} \tau^{\mathscr{J}}_{k,j} + \sum_{p\in P^{\text{post}}} c^{OPT}_{k,i,p} \psi^{\mathscr{J}}{}_{k,i,p} + \sum_{p\in P^{\text{pre}}} c^{OPT}_{k,j,p} \psi^{\mathscr{J}}{}_{k,j,p}) \\
&\overset{Obs.\ 4.3}{\leq} \sum_{\mathscr{J}\in\mathscr{E}} \sum_{(k,i,j)\in\mathscr{J}} (c^{\text{delay}}_{k,j} \tau^{*}_{k,j} + \sum_{p\in P^{\text{post}}} c^{OPT}_{k,i,p} \psi^{*}_{k,i,p} + \sum_{p\in P^{\text{pre}}} c^{OPT}_{k,j,p} \psi^{*}_{k,j,p}) \\
&\overset{\text{rearrange}}{\leq} \sum_{k\in K} \sum_{j\in L_k : \exists i \in L_k, \mathscr{J}\in\mathscr{E} : (k,i,j)\in\mathscr{J}} c^{\text{delay}}_{k,j} \tau^{*}_{k,j} \\
&+ \sum_{k\in K} \sum_{i\in L_k : \exists j \in L_k, \mathscr{J}\in\mathscr{E} : (k,i,j)\in\mathscr{J}} \sum_{p\in P^{\text{post}}} c^{OPT}_{k,i,p} \psi^{*}_{k,i,p} \\
&+ \sum_{k\in K} \sum_{j\in L_k : \exists i \in L_k, \mathscr{J}\in\mathscr{E} : (k,i,j)\in\mathscr{J}} \sum_{p\in P^{\text{pre}}} c^{OPT}_{k,j,p} \psi^{*}_{k,i,p} \\
&\overset{\text{add positive terms}}{\leq} \sum_{k\in K} \sum_{l\in L_k} c^{\text{delay}}_{k,l} \tau^{*}_{k,l} + \sum_{k\in K} \sum_{l\in L_k} \sum_{p\in P} c^{\text{OPT}}_{k,l,p} \psi^{*}_{k,l,p} = z^{\text{SP}}
\end{aligned}$$

This is exactly the statement of the lemma. □

8.3. **Proof of Lemma 4.5.**

Proof 1. *Assume we have a solution of* (3) *not fulfilling the statement. We set all variables* $y_{k,l} : (k,l) \in \mathscr{F}$ *to zero and all variables* $x_{k,(i,j)} : (k,i,j) \in \mathscr{G}$ *to zero. This operation reduces or preserves the objective value, since assignment costs are non-negative. We have to restore feasibility for Constraints* (2b) - (2l) *and* (3b)-(3l) *by adapting the other variables appropriately.*

First we can set for all $(k,l) \in \mathscr{F}$ *the variables* $\varsigma_{k,l}$ *to zero since Constraint* (2c) *reduces to* $0 \leq \varsigma_{k,l}$. *Furthermore, we can set* $\epsilon_{k,l}$ *to zero since Constraint* (2b) *reduces to* $\varsigma_{k,l} = \epsilon_{k,l}$. *Constraint* (2d) *then reduces to* $0 \leq \bar{\epsilon}_l + \tau_{k,l}$, *hence we can set* $\tau_{k,l}$ *to zero without getting infeasible, which leads to less or equal delay costs than before. Constraint* (2e) *reduces to* $0 = \varsigma_{k,l,TAXI\text{-}IN}$, *and this propagates due to Constraint* (2f) *over all post-flight turnaround process start and end time variables* $\varsigma_{k,l,p}$ *and* $\epsilon_{k,l,p}$, *if* $\delta_{k,l,p}$ *is equal to* 0. *The same logic applies due to Constraint* (2i) *which evaluates to* $\varsigma_{k,l,TAXI\text{-}OUT} = 0$ *and forces process start and end time variables of pre-flight turnaround processes to be zero due to Constraint* (2f), *with the same condition to* $\delta_{k,l,p}$. *We can guarantee this condition by setting* $\delta_{k,l,p}$ *to zero, while we have to restore validity of Constraint* (2j) *by setting the option variables* $\psi_{k,l,p}$ *to zero, which reduces or preserves the objective value. Since* $(k,l) \in \mathscr{F}$ *implies that* $(k,l,j) \in \mathscr{G}$ *or* $(k,i,l) \in \mathscr{G}$ *for incident edges, Constraint* (2h) *holds and* (2g) *reduces to* $0 = 0$ *since we have set* $\varsigma_{k,l,p} = \epsilon_{k,l,p} = 0$. *Next we show that we can restore feasibility for Constraints* (3b)-(3l). *Whenever we had that for* $(k,l) \in \mathscr{F}$ *that* $y_{k,l}$ *has been* 0 *before, the variables* w_e *and* θ_e *of incident arcs of the nodes corresponding to* (k,l) *in ground networks corresponding to arbitrary station-resource is* 0 *due to Constraints* (3c) *and* (3d), *together with Constraint* (3i). *In case we have changed the variable* $y_{k,l}$ *from* 1 *to zero, we can restore feasibility of the system by doing the following adaptations iteratively. Since* $y_{k,l}$ *has been* 1, *we have exactly one arc* e_1 *going to and exactly one arc* e_2 *coming from the node corresponding to* (k,l) *in every station-resource ground network for which* w_e *is equal to* 1 *due to Constraints* (3c) *and* (3d). *The arc* (e_1^-, e_2^+) *exists since we have defined the ground resource networks to be transitive. Hence we can set the variables* w_{e_1} *and* w_{e_2} *to zero and set the variable* $w_{(e_1^-, e_2^+)}$ *to one. Furthermore, we set the variable*

$w_{(e_1^-,e_2^+)}$ to θ_{e_1}, and afterwards set the variables θ_{e_1} and θ_{e_2} to 0. This guarantees directly that Constraints (3i) *for (e_1^-, e_2^+) and* (3f) *for e_1^- hold. Constraint* (3c) *holds for e_2^+ and Constraint* (3d) *holds for e_1^- since we have reduced one summand of the left hand side of each constraint by one and raised one summand by* 1*. In case e_1^- is the start node of the corresponding network, the same applies for Constraint* (3b)*. Furthermore, Constraint* (3e) *holds for e_2^+ since Constraints* (3e) *and* (3f) *imply that the right hand side of* (3e) *is larger or equal to the original value of $\theta_{e_2} + \overline{d}^{trans}_{(s,r)}$, which has been in turn due to Constraints* (3g), (3h) *and* (2f) *larger or equal than the original value of $\theta_{e_1} + \overline{d}^{trans}_{(s,r)}$, hence the right hand side of the constraint is in any case larger or equal to $\theta_{(e_1^-,e_2^+)} + \overline{d}^{trans}_{(s,r)}$ since this is equal to the original value of $\theta_{e_1} + \overline{d}^{trans}_{(s,r)}$. Finally, Constraints* (3g) *and* (3h) *evaluate to $0 \leq 0$ and $0 = 0$, respectively, such that we can set the variables $\varsigma_{k,l,(s,r)}$ and $\epsilon_{k,l,(s,r)}$ to zero, following Constraints* (3g) *and* (3h)*.*

To summarize, we've used an arbitrary optimal solution of Model (IC) *to construct a feasible solution of the model which fulfills the statement of the Lemma, since all variables which should be zero are zero. Since we have only used operations which reduce or preserve the objective value of the solution, this solution is optimal as well, and hence the Lemma is proven.*

8.4. **Algorithmic details.** The corresponding descriptions are contained in chapter 4.3 in the main paper, where the island types and their determination procedures are presented and explained.

Algorithm 5 Generate (SC)-island

1: **INPUT**: Model (IC-SC), (k, i, j) with $k \in K, (i, j) \in A_k$
2: **OUTPUT**: $\mathscr{I}$, $z^{\text{isl, rel}}_{\mathscr{I}}$
3: **procedure** GEN_SC((IC-SC), (k, i, j))
4: $\quad \mathscr{I} \leftarrow \{(k, i, j)\}$ ▷ obtain island
5: $\quad z^{\text{isl, rel}}_{\mathscr{I}} \leftarrow$ Solve Problem (IC-SC) ▷ calculate island costs
6: $\quad$ **return** $(\mathscr{I}, z^{\text{isl, rel}}_{\mathscr{I}})$
7: **end procedure**

Algorithm 6 Generate SF island

INPUT: Model (IC-SF), (k,i,j) with $k \in K, (i,j) \in A_k, \tau_{k,j} > 0$, flight-string $\mathscr{F}_k$ including connection (i,j), (SC)-island pool $\mathscr{E}_{(\mathrm{SC})}$ to determine $z^{\mathrm{isl,rel}}_{\{(k,l_1,l_2)\}} : (k,l_1,l_2) \in \mathscr{F}_k$.
OUTPUT: $\mathscr{I}$, $z^{\mathrm{isl,\ rel}}_{\mathscr{I}}$
procedure GEN_SF((IC-SF), (k,i,j), $\mathscr{F}_k, \mathscr{E}_{(\mathrm{SC})}$):
    $\mathscr{I} \leftarrow \{(k,i,j)\}$ ▷ obtain island
    for $(k,l_1,l_2) \in \mathscr{F}_k \setminus \{(k,i,j)\}$ **do**
        $\mathscr{I} \leftarrow \mathscr{I} \cup \{(k,l_1,l_2)\}$
        $(\tau^*, \psi^*) \leftarrow$ Solve Problem (IC-SF) for $\mathscr{I}$
        if $c^{\mathrm{delay}}_{k,l_2}\tau^*_{k,l_2} + \sum_{p\in P^{\mathrm{post}}} c^{OPT}_{k,l_1,p}\psi^*_{k,l_1,p} + \sum_{p\in P^{\mathrm{pre}}} c^{OPT}_{k,l_2,p}\psi^*_{k,l_2,p} = z^{\mathrm{isl,rel}}_{\{(k,l_1,l_2)\}}$ **then**
            $\mathscr{I} \leftarrow \mathscr{I} \setminus \{(k,l_1,l_2)\}$
            break
        end if
    end for
    $z^{\mathrm{isl,\ rel}}_{\mathscr{I}} \leftarrow$ Solve Problem (IC-SF) ▷ calculate island costs
    return $(\mathscr{I}, z^{\mathrm{isl,\ rel}}_{\mathscr{I}})$
end procedure

Algorithm 7 Generate (MF)-island

INPUT: Model (IC-MF), MP Solution Ω, SP Solution, active (SF)-Island Pool $\mathscr{E}_{(\mathrm{SF})}$, initial (SF)-island $B \in \mathscr{E}_{(\mathrm{SF})}$ with $B \subseteq \Omega$
OUTPUT: $\mathscr{I}$, $z^{\mathrm{isl,\ rel}}_{\mathscr{I}}$
procedure GEN_MF((IC-MF), Ω, SP*, $\mathscr{E}_{(\mathrm{SF})}$, B)
    $\mathscr{B} \leftarrow \emptyset$, $\mathscr{B}' \leftarrow \{B\}$ ▷ obtain island
    while $\mathscr{B}' \neq \emptyset$ **do**
        Choose arbitrary $B \in \mathscr{B}'$
        for $(k,i,j) \in B$ **do**
            for Arcs $(k',i',j') \in \Omega$ in conflict with (k,i,j) (Cases 1-4) **do**
                $\mathscr{B}' \leftarrow \mathscr{B}' \cup \{B'\} : B' \in \mathscr{E}_{(\mathrm{SF})}, (k',i',j') \in B'$ if $B' \notin \mathscr{B} \cup \mathscr{B}'$
            end for
        end for
        $\mathscr{B} \leftarrow \mathscr{B} \cup \{B\}$, $\mathscr{B}' \leftarrow \mathscr{B}' \setminus \{B\}$
    end while
    $\mathscr{I} \leftarrow \{(k,i,j) | \exists B \in \mathscr{B} : (k,i,j) \in B\}$
    $z^{\mathrm{isl,\ rel}}_{\mathscr{I}} \leftarrow$ Solve Problem (IC-MF) ▷ calculate island costs
    return $(\mathscr{I}, z^{\mathrm{isl,\ rel}}_{\mathscr{I}})$
end procedure

Algorithm 8 Generate (MS)-island

INPUT: MP solution Ω, SP objective value z^{SP}
OUTPUT: $\mathscr{I}$, $z^{\mathrm{isl,\ rel}}_{\mathscr{I}}$
procedure GEN_MS(Ω, z^{SP})
    $\mathscr{I} \leftarrow \Omega$ ▷ obtain island
    $z^{\mathrm{isl,\ rel}}_{\mathscr{I}} \leftarrow z^{\mathrm{SP}}$ ▷ calculate island costs
    return $(\mathscr{I}, z^{\mathrm{isl,\ rel}}_{\mathscr{I}})$
end procedure