OR Spectrum (2017) 39:65–93
DOI 10.1007/s00291-016-0443-1

REGULAR ARTICLE

Flight gate assignment and recovery strategies with stochastic arrival and departure times

Ulrich Dorndorf[1] · Florian Jaehn[2] · Erwin Pesch[3,4]

Received: 24 February 2015 / Accepted: 7 April 2016 / Published online: 22 April 2016

Abstract We consider the problem of assigning flights to airport gates. We examine the general case in which an aircraft serving a flight may be assigned to different gates for arrival, parking, and departure processing. The objectives can be divided into deterministic and stochastic goals. The former include maximization of the total assignment preference score, a minimal number of unassigned flights during overload periods, and minimization of the number of tows. A special focus lies on the stochastic objectives, which aim at minimizing the expected number of any kind of constraint violations, i.e. not respecting gate closures, violation of shadow restrictions (a situation in which gate assignments may cause blocking of neighboring gates) or of tow time restrictions and classical gate conflicts in which two aircraft are assigned to the same gate and are at the airport at the same time. We show that the minimization of expected gate conflicts can be modeled in a graph theoretical approach using the clique partitioning problem (CPP). We furthermore show that the classical (determin-

This work has been supported by the German Science Foundation (DFG) through the grant "Planung der Bodenabfertigung an Flughäfen" (PE 514/10-1).

✉ Florian Jaehn
florian.jaehn@wiwi.uni-augsburg.de

Ulrich Dorndorf
ulrich.dorndorf@inform-ac.com

Erwin Pesch
erwin.pesch@uni-siegen.de

1 INFORM GmbH, Pascalstr. 23, 52076 Aachen, Germany

2 Sustainable Operations and Logistics, University of Augsburg, Universitätsstr. 16, 86159 Augsburg, Germany

3 Institute of Information Systems, University of Siegen, Kohlbettstr. 15, 57068 Siegen, Germany

4 CASiM, HHL Leipzig, Jahnallee 59, 04109 Leipzig, Germany

istic) flight gate assignment problem, which can also be modeled using a CPP, can be integrated such that a simple though powerful model emerges, which no longer needs including a dummy gate, which is often used in practical gate assignment models. As constraint violations cannot fully be prevented, recovery strategies become necessary. We present a procedure for recovery planning that has proved its practical relevance at numerous airports. Finally, in an extensive numerical study we test our results on practical data, which contain a statistical analysis of flight arrival and departure times. The tests include a detailed comparison of current robustness measures and state-of-the-art approaches found in literature.

Keywords Gate assignment · Clique partitioning · Robustness

1 Introduction

Gate scheduling is concerned with finding an assignment of flights to terminal or ramp positions, called gates. It is a key activity in airport operations. With the increase of civil air traffic and the corresponding growth of airports in the past decades, the complexity of the task has increased significantly. At large international airports, several hundreds of flights must be handled per day. The task is further complicated by frequent changes of the underlying flight schedule on the day of operation, such as delays or aircraft changes.

Gates are scarce and expensive resources. Increasing the resource supply involves a time-consuming and costly redesigning of terminal buildings or the ramp and is usually not feasible in the short run. It is therefore of great economic importance for an airport or terminal operator to use the available gates in the best possible way.

The gate assignment also influences the quality of passenger service. Arriving flights sometimes have to wait on the ramp before traveling to their final position, because the assigned gate is still occupied by another flight. This situation is often caused by poor gate assignment or by failure to adapt an initial assignment to updates of the flight schedule. Changes of a gate schedule must take into account that gate assignments are published some time before the actual arrival or departure of a flight, for instance for planning purposes in other operational units, on passenger information displays and on boarding passes. The gate assignment also affects other ground services. A good assignment may reduce the number of aircraft tows required and may lead to reduced setup times for several ground service activities on the ramp as well as in the terminal.

The main input for gate assignment is a flight schedule with flight arrival and departure times and additional detailed flight information, including pairwise links between successive flights served by the same aircraft, the type of aircraft, the number of passengers, the cargo volume, and the origin or destination of a flight, which may further be classified as domestic or international. Furthermore, there is information about the distribution of the actual arrival and departure times as, e.g., the expected earliness or tardiness. The information in the flight schedule defines the time frame for processing a flight and the subset of gates to which it can or should be assigned, taking into account, e.g., aircraft-gate size compatibility, access to governmental inspection facilities for international flights, etc.

It is worth pointing out—from a practical point of view—one of the most important issues of gate scheduling: a gate schedule should be insensitive to small changes of input data; in other words, schedule flexibility is required. Obviously, the input data of any flight gate scheduling problem is subject to uncertainty and may change over time.

Input data uncertainty in gate scheduling may have a couple of reasons: (1) flight canceling or gate closure; (2) flight earliness or tardiness; (3) emergency flights; (4) severe weather conditions; (5) errors made by staff, and many others. For example, a tardy arrival of one aircraft may generate a chain of delayed arrivals for other aircraft which have been assigned to the same gate. In the worst case, this may lead to a "domino effect" and finally require a complete rescheduling, a situation which is highly undesirable.

We take a special focus on constraint violations, e.g., gate conflicts that appear if two flights have been assigned to the same gate and the aircraft are supposed to share this gate simultaneously, caused by schedule disruptions such as earliness or tardiness. Contrary to most earlier approaches, where the estimated arrival and departure times were the key input factors, we substitute them by the distributions of the arrival and departure times. The minimization of expected constraint violations will be our instrument for achieving assignment robustness in real planning situations. The flight gate assignment problem with the objective of minimizing gate conflicts can easily be modeled using a clique partitioning problem, which has been presented for deterministic flight gate assignments in Dorndorf et al. (2008). As a consequence, a fictitious dummy gate used in many models and at many airports can be omitted. As the actual arrival and departure times might lead to an infeasible assignment, we present a recovery strategy for the gate assignment problem. In this context, we differentiate between short-term recovery actions, caused by conflicts on short notice, and long-term recovery actions, which might even allow a beneficial but not enforced reassignment. Finally, we use real-life test data to evaluate how good our model and the according distribution reflect real-life situations. Furthermore, we compare current robustness measures and we show the effectiveness of our approach.

Mathematical models of assigning flights to gates are presented by Ding et al. (2004), Dorndorf (2002), and most recently by Kim et al. (2013) and Guépet et al. (2015). A detailed survey can be found in Dorndorf et al. (2007). However, robustness [see e.g., Mulvey et al. (1995) or, with an application to aviation, List et al. (2003)] has not been included in these approaches. An equal distribution of buffer times (gate idle times) when no aircraft is assigned to the gate is proposed by Hassounah and Steuart (1993) and Bolat (2000). Measuring robustness with fuzzy sets is described by Drexl and Nikulin (2006). An application at Amsterdam Schiphol is given by Diepen et al. (2012). A survey about robustness goals can be found in Nikulin (2006). There are very few publications considering situations similar to our application.

Dorndorf (2002) analyzes a deterministic case without introducing any robustness criteria. The objective contains the maximization of preference values, the minimization of dummy gate assignments, and the minimization of tows. The problem is modeled as multi-mode resource-constrained project scheduling problem with minimal and maximal time lags and the algorithmic details have been presented in Dorndorf et al. (2000). Gates are identified with the different modes of a project activity while every activity represents an aircraft either operated after arrival, or prepared for depar-

ture, or simply parking at its destination gate (mode). The model is solved using a layered branch-and-bound algorithm which is improved by constraint programming techniques and large neighborhood search. The binary branching either assigns a mode (gate) to an activity or forbids this assignment. Constraint propagation updates the variable domains. The computational intractability requires partitioning the set of flights into smaller clusters of flights competing for the same gates. While the branch-and-bound algorithm locally optimizes the gate assignment within the clusters (defining the layers of the algorithm), propagation globally adjusts the variable domains. Finally, a large neighborhood search improves bad assignments or removes potentially avoidable tows of aircraft.

Lim and Wang (2005) present a model for robust flight gate assignments in which the arrival and departure times are stochastic. Their model exclusively focuses on the robustness aspect, i.e. a distinction between arrival, parking and departure activities is not made and gate closures and shadow restrictions are not considered. The constraints only force that each flight is assigned to exactly one gate. There are no flight-to-gate preferences so that the expected number of gate conflicts is the only objective. The authors propose several estimation functions for the mean of the probability of a gate conflict of two aircraft. The proposed NP-hard model is solved using a hybrid metaheuristic which can be summarized as follows. A neighbor of a solution results from assigning exactly one aircraft to a different gate. In a first stage, a tabu search is applied using this neighborhood. In a second stage, the solution is improved through exchange cycles. In an exchange cycle, a variable number of flights is reassigned so that each of these flights is reassigned to a gate which was previously occupied by one of the other flights of the cycle. In other words, each of the reassigned flights "pushes" another flight to a different gate.

Stochastic flight delays are also recently considered by Castaing et al. (2016). In their optimization problem, they assign flights to gates in order to minimize the expected impact of gate blockage. In their network, gates flow through the system and all gates and aircraft are compatible with each other. In a second approach, potential fleet type–gate conflicts are recognized. Objective coefficients are approximated to represent the probability and severity of gate blockages as a function of gate turns, i.e., the number of departures and subsequent arrivals of aircraft at the same gates. There are the expected number and total time of gate blockages, the expected length of blockage for the outbound–inbound flight pair and weighted by the average number of connecting passengers on the inbound flight.

Neuman and Atkin (2013) model the gate allocation problem taking into consideration possible conflicts at taxiways around gates, i.e., they focus on an effective usage of gate space which also means avoiding the allocation of large gates. Ravizza et al. (2014) focus on the problem of aircraft ground movement from gates to a runway and vice versa. The idea is to reduce the environmental pollution by using waiting time as possible at the gate where engines are off rather than out on the taxiways.

Şeker and Noyan (2012) incorporate uncertainty to flight arrivals and departures and introduce robustness measures such as the number of conflicting flights and idle and buffer times into stochastic programming models. Tabu search is used to obtain reasonable schedules.

Kumar and Bierlaire (2014) consider the gate assignment at hub airports where arrival and departure of flights typically happen in waves. They impose penalties for non-assignments, aircraft tows and for not allocating the preferred gates. Additional objectives are maximization of passenger connection revenues, minimization of zone usage costs, and maximization of robustness of gate schedules. On the day of operation, a recovery of a schedule means minimizing schedule deviations in case of major disruptions. The model is implemented in OPL.

Yan and Tang (2007) present a framework for robust flight gate assignments which consists of three parts. Firstly, a stochastic flight delay gate assignment model is solved. Secondly, the results of the first part are used for a certain number of scenarios, to each of which a reassignment rule is applied. Based on the reassignment results, penalties (e.g., for the simultaneous assignment of two flights to the same gate) for the model in the first part are adjusted and the process is repeated. The gate assignment model used in the first part is very different from the problem considered here. In the model of Yan and Tang (2007), the arrival and departure time of each flight is to some extent part of the decision and neither tows nor tow time or shadow restrictions are included. Gate closures are only considered for a small number of different gate types. Besides the above-mentioned penalties, the objective includes the minimization of the total passenger waiting time but no gate preferences.

The remainder of the paper is organized as follows. A formal definition of the problem and related mathematical model are given in Sects. 2 and 3. A heuristic for solving the problem is proposed in Sect. 4. Section 5 proposes recovery strategies. Computational experiments and comparisons are presented in Sect. 6. Finally, a summary appears in Sect. 7.

2 Problem description

2.1 The flight gate assignment problem

The problem of finding a suitable gate assignment usually has to be addressed on three levels: during the preparation of seasonal flight schedules, the daily plans and on the day of operation, when gate schedules must be frequently altered to accommodate updates or disruptions in the flight schedule.

While at the airport, an aircraft goes through three stages of (1) arrival processing, (2) intermediate parking, and (3) departure processing. These stages are considered as separate (flight) activities that can potentially be assigned to different positions (gates) if necessary or advantageous. The aircraft may then have to travel between the assigned arrival, parking, and departure position. As this usually requires the use of tow tractors, we will refer to moving an aircraft between two positions as a tow. Thus, assigning an aircraft to gates is assigning the stages of processing, the arrival, parking and departure activity, to at most three gates. Obviously, towing requires some time and aircraft departure times must not be postponed due to towing. Therefore, such an operation is only possible if the ground time is sufficiently long. In case of a short ground time, it might be inevitable that all three activities have to be assigned to one gate. In this case, the activities are regarded as one flight and splitting them into three stages is omitted.

There are situations in which towing is inevitable, e.g., if an aircraft arrives as an international flight and has to be assigned to a gate with facilities for passport control but the aircraft departs as national flight and a gate without these facilities needs to be chosen. Especially for aircraft with a long ground time, it might be advantageous to tow this aircraft in order to free a popular gate or simply for providing more flexibility.

We are using the following notation for the input data:

- A set $N := \{1, 2, \ldots, n\}$ of n flight activities.
- A set $M := \{n + 1, n + 2, \ldots, n + m\}$ of m gates.
- The starting time S_i for each flight activity $i \in N$.
- The completion time C_i for each flight activity $i \in N$.
- The buffer time $t_{ij}(=t_{ji})$ between flight activities $i, j \in N$ calculated as $\max\{S_j - C_i; S_i - C_j\}$ and reduced by fixed gate setup times, which include, e.g., the attachment or removal of jet-bridges, pull or push-back of the aircraft. The buffer time represents the degree of flexibility of two activities that are assigned to the same gate. If $t_{ij} < 0$, both activities overlap, whereas a non-negative t_{ij} represents buffer (if i and j are an arrival and parking or a parking and departure of the same aircraft we define $t_{ij} = \infty$).
- A set of allowed gates $M(i) \subseteq M$ for each flight activity $i \in N$ where the assignment of the aircraft is feasible.
- The required tow time $\text{towtimes}_{i,j}$ for towing an aircraft from gate i to gate j. There are no tow times between two succeeding activities of an aircraft at the same gate ($\text{towtimes}_{i,i} := 0$).
- A set of shadow restrictions $S \subseteq N \times M \times N \times M$. The shadow restrictions restrict assigning two aircraft to neighboring gates because of wing tip proximity. Similarly, for an U-shaped terminal layout a big aircraft might cause blocking of a set of gates for all aircraft or those exceeding a certain size.
- A preference value $p_{ik} \geq 0$ for each flight activity $i \in N$ and each gate $k \in M$.
- The operator's utility function that allows us to compare different objectives by weighting each objective (to be introduced below) with a real number $\alpha_1, \alpha_2, \ldots$. This is necessary as different factors, namely preference values, the number of tows, and robustness influence the value of a solution.

If the arrival, parking and departure activity served by the same aircraft are not assigned to the same gate, one or two tows are needed. To ensure that the given arrival and departure times are maintained, the towing procedures must not exceed a certain time. The available time for towing is limited by the length of the parking activity, whose duration could completely be used for towing instead of parking. Therefore, we may consider the completion time of an aircraft's arrival (parking) activity as the starting (completion) time of towing the aircraft to its parking (departure) position. Thus, the duration of parking always includes the times for towing the aircraft to or from its parking position. Technically, it would also be possible to perform the towing operation sometime during parking and indeed, a rescheduling of the tow times may be performed when scheduling the tow tractors. This planning task is usually on a lower planning hierarchy level than the gate assignment. Thus, we stick to the assumption that towing is performed at the beginning or end of parking as the gain of flexibility seems to be limited compared to the operational effort induced by that.

Preference values and utility function are determined by airport or airline managers. The preference values are usually determined by the product of flight activity priorities and gate priorities. The flight activity priorities in turn are a classification concerning aircraft size, number of passengers, etc., whereas, the gate priorities are based on walking/taxiing distance, popularity scales, etc. Our experience at various airports showed that the utility functions significantly differ from airport to airport. For example, at some airports an extra tow is accepted even for a small increase in the flight gate preferences, while at others avoidable towing procedures are not used at all. It has to be noted that there are other possibilities to handle multiple objectives than transferring them into one objective (see e.g., Drexl and Nikulin 2006; Nikulin and Drexl 2010). However, for this application, where it is important to just implement one solution, it seems to be the most promising one.

The flight gate assignment problem can then be described as the problem of finding a mapping $f : N \to M$ which satisfies the following restrictions:

Gate closures: $f(i) \in M(i), \quad \forall i \in N$
Overlapping: $t_{ij} < 0 \Longrightarrow f(i) \neq f(j), \quad \forall i, j \in N$
Shadow restrictions: $(i,k,j,l) \in S \Longrightarrow f(i) \neq k \vee f(j) \neq l, \quad \forall i, j \in N, \forall k, l \in M$
Tow time restrictions: $S_{i''} - C_i \geq \text{towtimes}_{f(i),f(i')} + \text{towtimes}_{f(i'),f(i'')}$ $\forall i, i', i'' \in N$ of an aircraft's succeeding activities consisting of arrival i, parking i', and departure i''.

The tow time restrictions ensure that the total time for towing does not exceed the available time for parking.

In practical application it is often hard just finding a feasible solution. The given data sometimes even do not have a feasible solution. To overcome this problem, a dummy gate $n + m + 1$ with unrestricted capacity is introduced. Each flight activity can be assigned to the dummy gate. In a later planning stage flight activities assigned to the dummy gate have to be manually assigned to real gates. As this might counteract all previous optimization approaches, dummy gate assignments are highly undesirable. We will therefore show in the course of this paper, how the dummy gate, which is often used in practical applications, can be omitted. We define $p_{i(n+m+1)} := 0 \ \forall i \in N$.

The flight gate assignment problem is furthermore to optimize the linear combination of three objectives. Firstly, the preference values for the assignments are to be maximized. This ensures for instance priority for bigger aircraft, favors assignments close to a (previously determined) reference assignment, and diminishes dummy gate assignments. Secondly, the number of tows is to be minimized. Although tows are allowed or even necessary in order to make the assignment flexible, they produce costs and potentially avoidable traffic or congestion on the ramp or even aircraft damage. Thirdly, some kind of robustness measure should guarantee that the assignment is insensitive to small changes of the input data. This means that gate assignments are searched for, which can—at least to some degree—absorb flight tardiness or earliness without requiring a reassignment. An appropriate definition of this robustness measure is one of the main topics of this paper. For the moment let us assume that low buffer time (i.e. below a given parameter $t_{\max}$) between two flight activities assigned to the

same gate is punished, e.g., by minimizing the sum of $\max\{t_{\max} - t_{ij}, 0\}$ for all flight activities $i, j \in N$ assigned to the same gate.

In the literature, there are some approaches that solely focus on minimizing walking distances of passengers based on gate pairs or, highly connected with this, on minimizing the number of missed connecting flights (see Dorndorf et al. 2007). We do not consider this objective here, as it is not considered to be crucial at the airports from which we have data. In this context, it has to be noted that certain terminal layouts such as a star-shaped terminal in which transfer passengers must walk through the center, allow for considering walking distances through the gate preferences. Here, the gate preference increases the closer the gate is located to the center.

The three objectives considered here are weighted with positive number α_1 and negative numbers α_2, α_3, values reflecting the operator's utility mentioned above. The operator's utility function specifies the relevance of the different objectives, as e.g., an extra tow can be balanced if the preference values are raised by 100 or if the robustness goal is decreased by 30. As mentioned before, α_1, α_2, and α_3 commonly differ from airport to airport and can be regarded as part of the input data.

This flight gate assignment problem as it is modeled above appears at numerous international airports in Europe, North America, and Asia. It has been researched by Dorndorf (2002), Drexl and Nikulin (2006) and Dorndorf et al. (2008). The latter show that the problem can be modeled as a clique partitioning problem if there are no shadow restrictions and no tow time restrictions. We briefly summarize their ideas in the next subsection.

2.2 The clique partitioning problem

Given a complete, undirected, edge-weighted graph with $n + m$ vertices, any subset of the vertex set is called a clique. The clique partitioning problem (CPP) is to find a partition of the set of vertices into cliques so that the sum of all edge weights within all cliques is maximized. The problem can be described more formally using binary decision variables

$$x_{ik} = \begin{cases} 1 & \text{if vertices } i \text{ and } k \text{ belong to the same clique,} \\ 0 & \text{otherwise,} \end{cases} \qquad i, k \in \{1, 2, \ldots, n+m\}$$

for all edges $\{i, k\}$. If we denote the corresponding edge weights by w_{ik} the CPP can be described by the following model (see Grötschel and Wakabayashi 1989, 1990):

$$\begin{aligned} \max \quad & \sum_{1 \le i < k \le n+m} w_{ik} \cdot x_{ik} \\ \text{s.t.} \quad & x_{ij} + x_{jk} - x_{ik} \le 1 \quad \text{for } 1 \le i < j < k \le n+m \\ & x_{ij} - x_{jk} + x_{ik} \le 1 \quad \text{for } 1 \le i < j < k \le n+m \\ & -x_{ij} + x_{jk} + x_{ik} \le 1 \quad \text{for } 1 \le i < j < k \le n+m \\ & x_{ik} \in \{0, 1\} \quad \text{for } 1 \le i < k \le n+m \end{aligned} \tag{1}$$

The constraints guarantee that if vertices i and j belong to the same clique and vertices j and k do, then vertices i, j, k belong to the same clique.

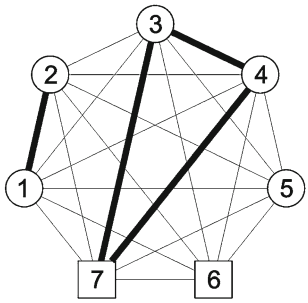

Fig. 1 A clique partition of a complete graph

Let us consider the n flight activities and the m gates as vertices of the complete graph. A partition of the set of vertices into cliques can then be seen as an assignment of flight activities to gates: if a flight activity vertex i is in the same clique as a gate vertex k, then we say flight activity i has to be assigned to gate k. If a flight activity vertex i is not in a clique that includes a gate vertex, then we assign flight activity i to the dummy gate. Obviously, any two gate vertices must not be in the same clique. Figure 1 shows a complete graph with seven vertices. Among them vertices 6 and 7 symbolize gates. In the clique partition $\{1, 2\}, \{3, 4, 7\}, \{5\}, \{6\}$ bold edges connect vertices being in the same clique. Flight activities 3 and 4 are assigned to gate 7 while flight activities 1, 2 and 5 are assigned to the dummy gate, as they did not obtain any particular gate assignment. The assignment would not change if vertices 1 and 2 each defined an own clique. This ambiguity has no influence on further procedures.

In case that the flight gate assignment problem described in the previous subsection has neither shadow nor tow time restrictions, we can find appropriate edge weights that allow us to use the clique partitioning problem for solving the gate assignment problem (see Dorndorf et al. 2008). Edges connecting two vertices that must not be in the same clique (each pair of gate vertices, a flight activity vertex i and all gates $k \notin M(i)$, two flight activity vertices $i, j \in N$ with $t_{ij} < 0$) are weighted $-\infty$. The weight of all other edges connecting flight activity vertices and gate vertices is their weighted preference value, i.e. $w_{ik} := \alpha_1 \cdot p_{ik} \forall i \in N, k \in M(i)$. Edges connecting vertices of succeeding flight activities belonging to the same aircraft, as there are arrival-parking or parking-departure, are weighted with α_2 in order to prevent tows. The remaining edges between flight activity vertices are weighted $\alpha_3 \cdot \max\{t_{\max} - t_{ij}, 0\}$.

The CPP is NP-complete unless the edge weights are all non-negative or all non-positive. Therefore, exact algorithms are usually not applicable for real-life instances usually consisting of far more than 100 vertices. However, there exist efficient heuristics for solving the CPP (see Dorndorf and Pesch 1994) that can be adjusted to assure the feasibility concerning shadow and tow time restrictions (see Dorndorf et al. 2008).

Although the clique partitioning problem reflects the flight gate assignment problem realistically, it suffers from two disadvantages. Firstly, there are still assignments to the dummy gate which is not possible in practice. This is undesirable as each dummy gate assignment has to be resolved manually no matter which flight delay scenario appears. Another drawback of the dummy gate is that once a flight activity is assigned to the dummy gate, the remaining flight activities are assigned without considering

that the one activity has to be introduced into the schedule later on. Secondly, the described robustness goal does not make any difference between flights with respect to their delay characteristics. It might be useful to have a longer time buffer after a flight which regularly is tardy while flights that are commonly on time might only need a smaller time buffer.

We will show that these two disadvantages can be eliminated considering starting and completion time probability distributions instead of deterministic values. However, non-deterministic activity times will somewhat change the problem and several modifications have to be introduced. It is not immediately evident whether all restrictions are satisfied. More precisely, the stochastic factor makes it impossible to generate flight gate assignments which guarantee feasibility for every scenario. This makes recovery strategies necessary.

3 Stochastic arrival and departure times

Let $i, j, k \in V$ be three activities and let $S_i, S_j, S_k \in \mathbb{N}$ denote the (deterministic) starting times while $C_i, C_j, C_k \in \mathbb{N}$ denote the completion times. If i, j and k are all served by the same aircraft and if i is the arrival activity, j is the parking and k is the departure activity, we can assume that $C_i = S_j$ and $C_j = S_k$.[1] From now on let all starting and completion times be random variables of a discrete distribution function. That is, for every S_i (C_i) there is a given function $X_{S_i} : \mathbb{N} \to \mathbb{R}$ $(X_{C_i} : \mathbb{N} \to \mathbb{R})$ where $P(X_{S_i} = x)$ describes the probability that the starting time of i is x. Note that these distribution functions need not to be independent. Especially X_{S_i} and X_{C_i} are usually strongly dependent. In practical applications these distribution functions can be estimated from historical data (see e.g., Richter 2005). We assume that in the planning stage, when the assignments of the flight activities to gates have to be determined, the exact starting and completion times are not known, but the distribution functions are.

Since most objectives and restrictions are only well-defined in the deterministic case, we have to find equivalent ones for the stochastic case. Accordingly, the edge weights for the clique partitioning problem have to be changed.

The assignments to the gates are still well-defined, and therefore the preference score objective does not need to be changed. Consequently, the number of tows is well-defined, and this objective can be retained as it is. However, without fixed times it is not possible to determine whether gate closure restrictions, overlapping restrictions, shadow restrictions, and the tow time restrictions are satisfied. Furthermore, the robustness goal cannot be achieved in the previous way. To redefine these restrictions and the objective, we determine the probability of an overlap:

Definition 3.1 For every pair i, j of flight activities, the probability that both activities appear to be at the airport at the same time is given by[2]

[1] We assume that the time for parking includes a possible towing procedure. That is, $C_j - S_j$ is the available time for towing and parking the aircraft.

[2] For convenience we do not consider gate setup times. They can easily be regarded when calculating the overlapping probabilities. In fact, in the computational tests gate setup times are included.

$$\mathrm{ol}(i, j) := 1 - P(X_{C_i} \leq X_{S_j}) - P(X_{C_j} \leq X_{S_i}).$$

Obviously, $\mathrm{ol}(i, j) = \mathrm{ol}(j, i)$. Note if i and j are served by the same aircraft, then $P(X_{C_i} \leq X_{S_j}) = 1$ or $P(X_{C_j} \leq X_{S_i}) = 1$ and therefore $\mathrm{ol}(i, j) = 0$. Since all X are discrete random variables, $\mathrm{ol}(i, j)$ can be determined by taking the sum of the random variables, and in case of independent random variables through convolution.

The overlapping restrictions are surely fulfilled if two flight activities i and j are only assigned to the same gate if $\mathrm{ol}(i, j) = 0$. However, since very big delays are unlikely but still possible, we have to assume $\mathrm{ol}(i, j) > 0$ for any pair of activities i and j not served by the same aircraft. In practical application it is therefore only possible to satisfy the overlapping restrictions with a high number of dummy gate assignments. This again leads to assignments that are not applicable, as mentioned before. It seems to be impossible to find an applicable assignment which guarantees feasibility for each possible scenario. In fact, gate assignments at big international airports most commonly turn out to be infeasible, especially if some flight has a delay of several hours. In such a case recovery strategies are needed. We may therefore allow our assignment to be infeasible in the former sense, but try to keep the probability of infeasibility low. That is, we drop the overlapping restrictions and dummy gate assignments, and add a new objective which minimizes the expected number of gate conflicts. In other words, we turn a hard constraint into a soft constraint. This new objective also replaces the previous robustness goal. We can easily integrate this new objective if we now weight an edge between flight activity vertices i and j with $-\mathrm{ol}(i, j)$ as shown in the following theorem:

Theorem 3.2 *Given a complete graph with n flight activity vertices and edge weights* $w_{ij} := -\mathrm{ol}(i, j)$. *Consider a clique partition of this graph in which* $x_{ij} = 1$ *denotes that i and j are assigned to the same gate and* $x_{ij} = 0$ *if not.*

The absolute value of the sum of the clique weights corresponds to the expected number of gate conflicts.

Proof Let A be a possible scenario, i.e. A defines for every activity i a fixed starting time S_i with $P(X_{S_i} = S_i) > 0$ and a fixed completion time C_i with $P(X_{C_i} = C_i) > 0$. Let $P(A)$ be the probability of scenario A.

Since all random variables X are discrete, there is only a countable set of scenarios with $P(A) > 0$. Let us therefore define the set of all scenarios:

$$\mathfrak{A} := \{A_1, A_2, \ldots\}$$

Since in a particular scenario A_k starting and completion times are given, it is easy to determine the number of gate conflicts in this scenario, which we will denote by $\mathrm{con}(A_k) := |\{(i, j) | x_{ij} = 1 \wedge i, j \text{ overlap in } A_k\}|$. The expected number of gate conflicts for the given clique partition can then be written as

$$\sum_{k \in \{1,2,\ldots\}} P(A_k) \cdot \mathrm{con}(A_k).$$

The edge weights can be rewritten as

$$w_{ij} = -\mathrm{ol}(i,j) = -\sum_{\substack{k\in\{1,2,\ldots\}\\ i,j \text{ overlap in } A_k}} P(A_k)$$

Then the absolute value of the sum of the clique weights is

$$\begin{aligned}\left|\sum_{1\le i<j\le n} x_{ij}\cdot w_{ij}\right| &= \sum_{1\le i<j\le n} x_{ij}\cdot \sum_{\substack{k\in\{1,2,\ldots\}\\ i,j \text{ overlap in } A_k}} P(A_k)\\ &= \sum_{1\le i<j\le n}\ \sum_{\substack{k\in\{1,2,\ldots\}\\ i,j \text{ overlap in } A_k}} x_{ij}\cdot P(A_k)\end{aligned}$$

which corresponds to the expected number of gate conflicts. □

The theorem shows that if the expected number of gate conflicts is the only objective, the CPP would deliver an assignment with the lowest gate conflict expectation. Thus, using $-\mathrm{ol}(i,j)$ as edge weight for edges connecting flight activity vertices allows us to introduce the expected gate conflict objective into our model. We may note that this decision criterion (and any other criterion) based on the expected value could lead to unfavorable solutions if the decision-maker is not risk neutral. For example, a solution with a gate conflict in every possible scenario might be considered as good as no gate conflict in 99 % of scenarios and 100 gate conflicts in 1 % of all scenarios. Yet, the benefit of this approach is that a dummy gate becomes superfluous, i.e., every flight activity is assigned to a real gate.

Analogously, we can introduce the gate closures into the new model with stochastic times. We have to distinguish whether a gate closure is temporary or outright. In the former case, again we drop this restriction and move it into the objective as follows. In the deterministic model, the edges connecting a flight activity vertex and a gate vertex were either weighted $-\infty$ (in case of a gate closure for this flight activity) or proportional to the preference value. If the gate closure is time dependent, say, the gate is closed until time t, we can easily determine the probability that flight activity i is infeasibly assigned to this gate by $P(X_{S_i} < t)$. This probability will be included to the edge weight, which already contains the flight/gate preference scores. If the gate closure is outright, we either keep the gate closure restriction as it is and strictly prohibit any according flight gate assignment using edge weight $-\infty$, or we choose α_2 (see Sect. 2.1) as edge weight implying one certain conflict if this edge is chosen. The question which weight should be preferred is user dependent and for our applications we decided to use $-\infty$.

Shadow restrictions are only relevant to pairs of overlapping flight activities assigned to specific gate pairs. Similarly to the overlapping restrictions we introduce an objective, which minimizes the expected number of violated shadow restrictions, i.e.

$$\min \sum_{(i,k,j,l)\in S} x_{ik} \cdot x_{jl} \cdot \mathrm{ol}(i,j).$$

Finally, the tow time restrictions can also be transferred to the objective. For a given assignment x of flight activities to gates and a parking activity i, the required tow time is known in advance and may be denoted by $tt(i,x)$. In case i is assigned in x to a different gate than its arrival and its departure activity, then $tt(i,x)$ includes the required time of two towing procedures. The probability that the duration of the parking activity exceeds this tow time can be determined by the sum of the random variables. The new tow time objective maximizes this probability for every parking activity and thus minimizes the expected number of violations against the tow time restrictions:

$$\max \sum_{\substack{i\in\{1,\ldots,n\}\\ i \text{ is parking}}} P\left(X_{C_i} - X_{S_i} \geq tt(i,x)\right)$$

Thus, we get a flight gate assignment problem only with restrictions concerning gate closures. The objectives consists of:

1. Maximization of flight/gate preference scores
2. Minimization of the number of tows
3. Minimization of the expected number of overlaps
4. Minimization of the expected number of gate closure violations
5. Minimization of the expected number of shadow restriction violations
6. Minimization of the expected number of tow time restriction violations.

These objectives will be weighted with real numbers $\alpha_1, \ldots, \alpha_6$. Note that the deterministic objectives 1 and 2, and the stochastic objectives 3 and 4 can be included into the CPP as described above. Furthermore, there are restrictions which prohibit two gate vertices to belong to the same clique. The overall objective for a given solution x is the sum of clique weights and the values of objectives 5 and 6. This implies that we do not have a classical clique partitioning problem anymore but a modified one (mCPP). As we no longer allow for assignments to a dummy gate, we have to add further restrictions to our model, because each flight activity vertex is supposed to be assigned to exactly one gate vertex. If vertices $\{1,2,\ldots,n\}$ represent flight activities and vertices $\{n+1,\ldots,n+m\}$ represent gates, we replace model (1) by

$$\begin{aligned}
\max \quad & \sum_{1\leq i<k\leq n+m} w_{ik} \cdot x_{ik} - \alpha_5 \cdot \sum_{(i,k,j,l)\in S} x_{ik} \cdot x_{jl} \cdot \mathrm{ol}(i,j) \\
& + \alpha_6 \cdot \sum_{\substack{i\in\{1,\ldots,n\}\\ i \text{ is parking}}} P\left(X_{C_i} - X_{S_i} \geq tt(i,x)\right) \\
\text{s.t.} \quad & x_{ij} + x_{jk} - x_{ik} \leq 1 \qquad \text{for } 1 \leq i < j < k \leq n+m \\
& x_{ij} - x_{jk} + x_{ik} \leq 1 \qquad \text{for } 1 \leq i < j < k \leq n+m \\
& -x_{ij} + x_{jk} + x_{ik} \leq 1 \qquad \text{for } 1 \leq i < j < k \leq n+m
\end{aligned}$$

$$x_{ik} \in \{0,1\} \qquad \text{for } 1 \le i < k \le n+m$$
$$\sum_{k=n+1}^{n+m} x_{ik} = 1 \quad \text{for } 1 \le i \le n \tag{2}$$

Note that just in order to get a maximization problem, we have changed the signs of the objective accordingly.

4 A modified Ejection Chain Algorithm

The clique partitioning problem is the focus of various research papers and exact as well as heuristic approaches have been presented. Among the heuristic algorithms, an Ejection Chain Algorithm seems to perform best (see Dorndorf and Pesch 1994; Jaehn and Pesch 2013). We modify this algorithm in order to make it applicable for the mCPP. The major changes compared to the Ejection Chain Algorithm found in the literature are threefold. Firstly, we implement a different initial solution, which tries to assign all activities of an aircraft to one gate. Secondly, neighboring solutions are not only evaluated concerning edge weights, but concerning their influence on the whole objective. Finally, we prohibit certain moves of vertices, which lead to infeasible solutions. This especially includes removing a gate representing vertex from its current cluster.

Finding an initial solution satisfying the restrictions of (2) is easily found by greedily assigning every flight activity to some gate as long as it fulfills the gate closure restrictions. However, to keep the number of tows low, connected flight activities (i.e. arrival-parking-departure) are together assigned to one gate. Obviously, this is only done if the preference values of the three activities are non-negative in order to prevent assignments to closed gates. The remaining activities are then greedily assigned to gates.

Given some solution x of the mCPP, this solution divides the set of vertices into clusters. The most simple procedures of our algorithm are one-step moves in which a solution x' is obtained from a solution x by moving only one activity vertex to another cluster. In the following, we describe the effects of such a one-step move on the six objective functions. The effect on the objectives 1 to 4, included in the mCPP, if vertex i is moved from its current cluster $C(i)$ to a cluster D can be calculated by

$$h_{1-4}(i, C(i), D) := \sum_{j \in D} w_{ij} - \sum_{j \in C(i) \setminus \{i\}} w_{ij}.$$

To describe the effect of such a move on the shadow objective 5, let us define that k is the gate vertex of i's current cluster $C(i)$ ($x_{ik} = 1$) and let k' be the gate vertex of i's new cluster D ($x'_{ik'} = 1$). Then we get

$$h_5(i, C(i), D) := \sum_{(j,l): x_{jl} = 1 \wedge (i,k,j,l) \in S} \mathrm{ol}(i,j) - \sum_{(j,l): x_{jl} = 1 \wedge (i,k',j,l) \in S} \mathrm{ol}(i,j).$$

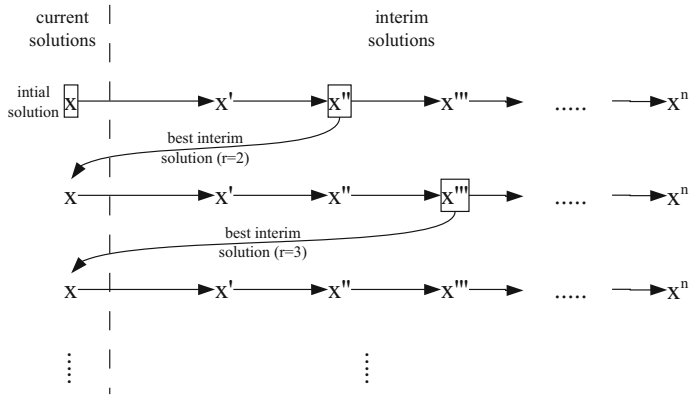

Fig. 2 Illustration of Algorithm 1

Finally, the effect on objective 6 is described by means of a parking activity park(i) of activity i. If i is a parking activity, then park$(i) = i$. If i is an arrival, park(i) denotes the successor of i and if i is a departure park(i) denotes the predecessor of i, both for the same aircraft. The required time for towing the aircraft corresponding to activity i in a given solution x has been denoted by $tt(i, x)$.

$$h_6(i, C(i), D) := P\left(X_{C_{\text{park}(i)}} - X_{S_{\text{park}(i)}} \geq tt(i, x')\right) - P\left(X_{C_{\text{park}(i)}} - X_{S_{\text{park}(i)}} \geq tt(i, x)\right)$$

The effect on the overall objective can then be summed up:

$$h(i, C(i), D) := h_{1-4}(i, C(i), D) + h_5(i, C(i), D) + h_6(i, C(i), D)$$

Starting from a feasible solution, our algorithm greedily searches for a sequence of at most n moves. Each move maximizes $h(i, C(i), D)$ under the restriction that a vertex must not be moved more than once in this sequence. Practically, this means that every flight activity is moved from its current gate to another one. However, to reach an improved feasible solution for the next iteration, only the subsequence is performed, which raises the objective at most. If there is no such subsequence, the algorithm will stop.

The procedure of the algorithm is also illustrated in Fig. 2. Each horizontal arrow denotes a one-step move.

Algorithm 1 **0. Initialization:** Create m clusters each containing a gate vertex $k \in M$. If possible, assign every triple i, i', i'' of arrival, parking and departure activities to a cluster which contains a gate vertex k^* with $p_{ik^*} + p_{i'k^*} + p_{i''k^*} \geq p_{ik} + p_{i'k} + p_{i''k} \forall k \in M(i) \cap M(i') \cap M(i'')$. Assign every remaining flight activity vertex i to a cluster which contains a gate vertex k^* with $p_{ik^*} \geq p_{ik} \forall k \in M(i)$. The obtained solution is the initial (incumbent) solution.

1. Finding a sequence of n moves: Copy the incumbent solution and obtain an interim solution. Denote all n flight activity vertices of the interim solution as non-tabu.

1.1 Finding a best move: Among all one-step moves of non-tabu vertices of the interim solution, find the one which leads to a feasible solution and which maximizes $h(i, C(i), D)$.

Perform this move to the interim solution, mark the moved vertex i^* as tabu and update all clusters.

Repeat 1.1 to the interim solution until all vertices are marked tabu or until no feasible solution is found.

Determine $r \in \{0, \ldots, n\}$ so that the interim's objective is maximized after performing the first r moves.

2. Improving the solution: If $r > 0$ perform the first r moves to the incumbent solution and go to step 1.

If $r = 0$ then stop and return the current (incumbent) solution.

Note that in 1.1 the one-step move is performed even if $h(i, C(i), D) < 0$, i.e. we allow for a temporary worsening of the objective function value. $h_{1-4}(i, C(i), D)$ can be determined for every i in time complexity $O((n + m) \cdot \log(n + m))$ as Dorndorf and Pesch (1994) show. For every i we need to check all other flight activities to obtain $h_5(i, C(i), D)$ under the assumption that all overlap probabilities have been calculated in advance. This leads to a time complexity of $O(n^2)$. Furthermore, if we calculate the distribution of the lengths of the parking activities in advance, we can determine $h_6(i, C(i), D)$ in linear time. Hence, the best one-step move can be found in $O(n^2)$ and as every iteration consists of a sequence of at most n moves the complexity of a whole iteration is $O(n^3)$ (if $n \geq m$).

5 Recovery strategies

Since we do not have exact information about arrival and departure times, gate conflicts are inevitable. As a matter of fact, in practice the assignments made in the planning stage often turn out to be infeasible, e.g., because of tardy flights. In order to cope with this situation, we present an online decision support system, which was developed in cooperation with the authors and which is used at several airports in North America, Europe, and Asia.

Assignments of flight activities to gates, made during the preparation of seasonal flight schedules, are repeatedly renewed using the previous assignment as a reference in order to stay close to the original plan. This renewal is caused by changes of the input data such as canceling of flights, rescheduling of flights, aircraft type changes leading to different gate closures, etc.

On the day of operation, the aforementioned procedure of renewing the assignment cannot be applied as before, due to two reasons. Firstly, the input data (including the arrival and departure times) are updated several times a minute. Methods for assigning flights to gates (in the planning stage) generally demand much more computational time if they consider the complete planning horizon of 1 day. Secondly, the input data might inevitably lead to infeasibility. In this case decision-makers usually change the

input data manually, e.g., by postponing flight arrival times. It simply means that the aircraft has to wait until a gate is available.

The outline of the online decision support system for the day of operation is as follows: the set N of flight activities is split into two disjoint subsets N_1 and N_2 using a time parameter t_{break} determined by the airport:

$$N_1 := \{i \in N | E(S_i) < t_{\text{break}}\}$$
$$N_2 := \{i \in N | E(S_i) \geq t_{\text{break}}\}$$

N_1 contains all flight activities already in process or with an expected starting time smaller than t_{break} while N_2 consists of all other flight activities. Many airports choose t_{break} to be a point of time when gate assignments are announced on passenger displays, which usually happens 1 or 2 h in advance. Parameter t_{break} is updated every 5 min. Recovery actions will be taken separately for N_1 and N_2. Let us firstly consider the short-term recovery actions on N_1. Note, an aircraft arriving at a gate blocks the gate at least until the activity's completion time and must not be reassigned during this period. Indeed, most aircraft that are considered to be in N_1 are already present at the airport. For the remaining aircraft in N_1, the time slot in which each aircraft lands is known. We may therefore assume that the starting and completion times of all flight activities in N_1 are deterministic.

5.1 Short-term recovery actions (N_1)

Disruptions in N_1 are very unlikely, but a reassignment of a flight activity from N_1 is much worse than a reassignment of a flight activity from N_2. A short-term reassignment might have a serious impact on punctuality and passenger service as passengers, baggage, and staff have to be redirected. The most important goal is to keep the current assignment as it is. Thus, flight activities in N_1 should only be reassigned if it is inevitable. Reasons for that can be a sudden gate breakdown or blocking of a gate by an aircraft facing problems during handling. Small changes in the input data (such as changes in the arrival or departure time) should therefore have no effect on the assignment unless the update results in an infeasible (current) assignment.

An operator is supposed to resolve conflicts either manually or by means of a decision support system. A manual resolution is only appropriate if the conflict is apparently easy to resolve, e.g., delaying the starting time of a flight activity by a couple of minutes until a blocked gate becomes available.

If a manual resolution of the conflict is not obvious, the operator can use the decision support system, which basically incorporates an algorithm using deterministic start and completion times and thus allowing dummy gate assignments, see Dorndorf et al. (2008). In order to have a small decision problem containing a substantially smaller number of variables, a feasible reassignment is calculated while flight activities from N_2 are neither included nor are any conflicts with them observed. The priority values of each flight activity to its currently assigned gate are considerably increased so that reassignments are kept to a minimum. The run times of this program must be kept in the range of a few seconds which is ensured by the small number of flight

activities considered. In a subsequent procedure the operator can slightly change the input data, namely the completion times of flight activities in N_1. The flight activities can be accelerated through allocating additional resources in order to locally limit reassignments or to avoid dummy gate assignments.

5.2 Long-term recovery actions (N_2)

Whenever t_{break} and N_2 are updated, all flight activities in N_2 are assigned using the previous solution as a reference assignment. That means that each flight activity of N_2 may be reassigned, but the preference values of each flight activity to its previously assigned gate are increased by a certain factor. This factor strongly depends on the requirements of the airport, especially on its policy of the time of announcing (preliminary) gate assignments. The current assignment of flight activities from N_1 is part of the input data in order to avoid conflicts between flight activities of both sets N_1 and N_2. The long-term recovery is achieved using Algorithm 1.

The whole recovery process of the online decision support system is illustrated in Fig. 3. This process is restarted periodically, e.g., every 5 min.

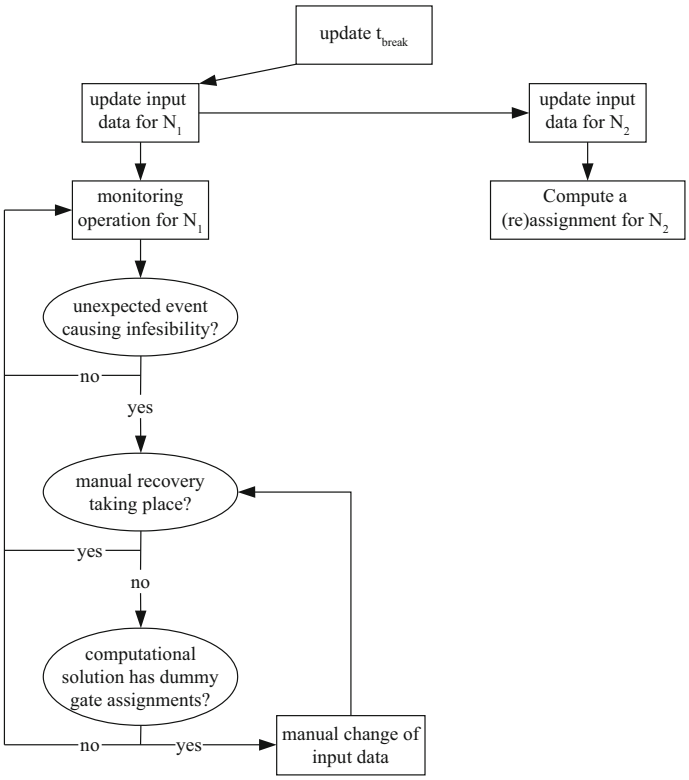

Fig. 3 The recovery process

6 Computational results

The flight gate assignment problem has been implemented using the presented model and algorithm. We used a large set of practical data that is described in more detail in Sect. 6.1. In order to get meaningful results, we compare our model to other models from literature and we compare different robustness measures. The comparative data are described in Sect. 6.2. Finally, the results of the flight gate assignment tests and the recovery tests are shown in Sects. 6.3 and 6.4.

6.1 Test data

We have used real-life data from a 14-day period at an international airport. The airport consists of 93 gates and the number of flight activities varies between 635 and 820 per day. The 14 test records are shown in Fig. 4 with each instance representing 1 day of operation. The set of flight activities is split into subsets of single activities and triple activities. Each triple activity consists of an arrival, a parking and a departure activity served by the same aircraft. Furthermore, Fig. 4 contains the total processing time of all flight activities and the maximum number of aircraft that are to be operated at the airport at the same time. The distribution of the flight activities during a day is shown for data set 1_08 in Fig. 5.

Although there are 93 gates and not more than 57 aircraft at the airport at one point of time, due to several reasons the airport is working to full capacity. Firstly, the gate closure restrictions prohibit an arbitrary assignment of flight activities to gates. On average, because of gate closures a flight activity can only be assigned to one out of 14 gates (not considering a dummy gate) which implies, e.g., for data set 1_08 that each gate can, on average, operate only 110 flight activities. For the same data set the

Data	n	single activities	triple activities	processing (hours)	simultaneous flights (peak)
1_08	735	114	207	730	45
1_09	801	90	237	756	45
1_10	783	93	230	726	45
1_11	806	86	240	765	46
1_12	797	89	236	728	45
1_13	800	92	236	761	44
1_14	666	54	204	987	57
2_03	723	132	197	1059	51
2_04	820	85	245	963	52
2_05	799	82	239	990	53
2_06	818	80	246	1015	55
2_07	815	80	245	977	54
2_08	818	77	247	1103	54
2_09	635	59	192	1056	57
Σ	10816	1213	3201	12616	

Fig. 4 The 14 records of the test set

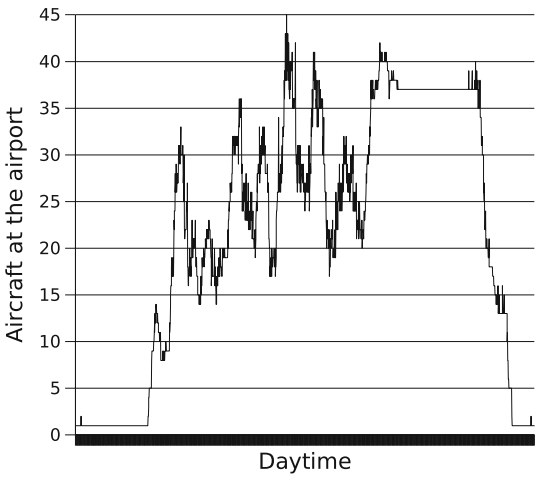

Fig. 5 Number of aircraft at the airport (Data set 1_08)

number of gates that can operate a certain percentage of flight activities (concerning gate closures) is listed in the following table. There is no gate that belongs to more than 30 % of the sets $M(i)$ of allowed gates for all activities $i \in N$.

Potential number of flight activity assignment	0–36 (≤5 %)	37–73 (≤10 %)	74–110 (≤15 %)	111–147 (≤20 %)	148–183 (≤25 %)	184–220 (≤30 %)	221–735 (>30 %)
Number of gates	14	12	16	25	19	7	0

Secondly, because of gate setup times (included in t_{ij} as mentioned above) a flight activity assigned to a certain gate blocks this gate for a longer time than the actual duration of this flight activity. Thus, this flight activity might block this gate for another flight activity even if their ground times do not overlap. Finally, shadow restrictions frequently cause a simultaneous blocking of more than one gate by a single activity.

For each activity only the scheduled starting and completion times are known. To receive their realistic distributions, we have used the statistical evaluations of Richter (2005). She realized that the starting time of an arrival activity is strongly dependent on the origin of the flight, the airline, and whether it is peak time or not. Other factors turned out to have no or only little influence on the starting time, e.g., day of the week, model of aircraft or number of passengers. Furthermore, Richter (2005) concluded that for this airport the distribution of the arrival times can be approximated best with a gamma distribution whose expected value is 2 min before the scheduled time, the median is 6 min before scheduled time, and the earliest possible arrival is about half an hour before the scheduled time. We have adopted this distribution for our model and discretized with 1 min intervals.

The gamma distribution is defined by two parameters $\alpha > 0$ and $\beta > 0$ and its density function γ is

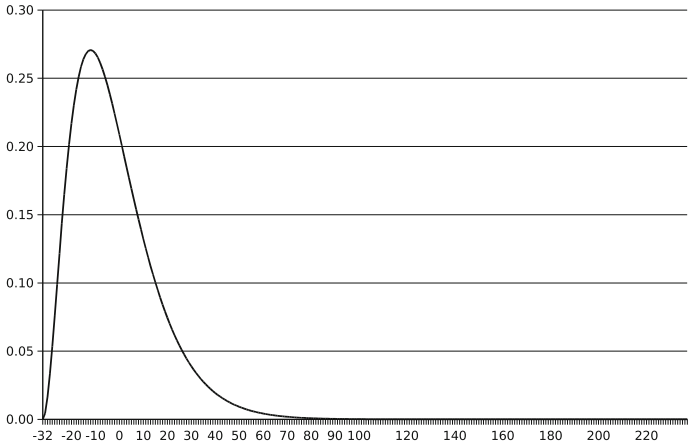

Fig. 6 Gamma distribution

$$\gamma(x, \alpha, \beta) := \begin{cases} \dfrac{1}{\beta^x \Gamma(\alpha)} x^{\alpha-1} \mathrm{e}^{-\frac{x}{\beta}} & x > 0 \\ 0 & x \leq 0 \end{cases}$$

with

$$\Gamma(\alpha) := \int_0^\infty t^{\alpha-1} \mathrm{e}^{-t} \, \mathrm{d}t.$$

The expected value of γ is $\alpha \cdot \beta$. The distribution will now be adjusted in order to use it for the arrival times. We discretize the distribution function so that $0 \leq x < 0.1$ stands for an early arrival of more than 31 up to 32 min, $0.1 \leq x < 0.2$ stands for an earliness of more than 30 and at most 31 min, etc. The deviation Δ from the scheduled time can then be described by $\Delta := 10 \cdot x - 32$. The probability of an arrival being, e.g., 20 min early ($\Delta = -20$) is calculated by

$$\int_{1.2}^{1.3} \gamma(x, \alpha, \beta) \, \mathrm{d}x.$$

Furthermore, we choose $\alpha = 3$ and $\beta = 1$, which leads to an expected value $x = 3$ and $\Delta = -2$ as required. The median is $x = 2.674$ and $\Delta = -5.26$. We accept that the median is slightly higher than required in order to have integral parameters α and β. The distribution $\gamma(\Delta, \alpha, \beta)$ is shown in Fig. 6.

This distribution implies that for example 60 % of all arrivals are early or on time and 0.49 % of the arrivals are delayed by more than an hour. For the computational tests we assume that the arrival times of two different aircraft are independent. The reasons for this assumption are twofold. Firstly, we simply have no information connected to the data at hand on which flights show dependencies. As mentioned before, any given information on dependencies can easily be considered when calculating the

overlapping probabilities. Secondly, most factors for a delay do not indicate a significant correlation between two aircraft, and if they do (such as weather), they are often so highly correlated that the effects on the system can be neglected. For example, a weather caused closing of the runway will generally delay all arriving aircraft by the same time span.

Next we have to determine the distributions for the completion times of arrival activities and the starting and completion times of parking and departure activities. Obviously, those are strongly dependent on the starting time of the arrival activity. For each arrival and departure activity there is a given fixed processing time. The completion of an arrival activity (which defines the starting time of the succeeding parking activity) is either on time (if the arrival is early or on time) or, in case of a delay, it is just the shifted distribution of the starting time. The starting time of a departure activity (which corresponds to the completion time of the preceding parking activity) will also be determined depending on the distribution of the arrival activity. If the arrival activity is early or if the tardiness does not exceed the length of the parking activity (which can be used as buffer time), then the departure activity will start on time. Otherwise the delay of the arrival activity will be propagated to the departure activity. In this case the starting and completion time of the parking activity will be identical. Again, the distribution of the completion time of the departure activity is the shifted distribution of the starting time. Given these distributions, the overlapping probabilities $\mathrm{ol}(i, j)$ for two flight activities i and j are then calculated by convolution of the according distributions (which are truncated for values above the planning horizon).

Finally, let us consider the weights α_i of the objectives. Although in the presented model the preference values were normalized so that dummy gate assignments have a preference value of zero, in our comparison we will use negative preference values for the dummy gate assignments for two reasons. Firstly, we explicitly want to point out the detriment of dummy gate assignments. Secondly, those are the values used at the airport and in former publications using these test data (see Dorndorf (2002); Dorndorf et al. (2008)). Thus, the preference value weight α_1 is split into $\alpha_1(I)$ and $\alpha_1(II)$. $\alpha_1(I)$ denotes the preference value weights for dummy gate assignments while $\alpha_1(II)$ describes the preference value weight for real gate assignments. The parameters are $\alpha_1(I) = 400, \alpha_1(II) = \alpha_2 = 1, \alpha_3 = \alpha_4 = \alpha_5 = \alpha_6 = 400$. That means that every dummy gate assignment is regarded as bad as an expected violation, say, e.g., a gate conflict or a shadow restriction violation. Note that in order to get a realistic model each dummy gate assignment and each value of the robustness objective is also weighted with a specific priority value of the flight activity.

6.2 Comparative data

We have implemented and tested our algorithm and compared the results to other relevant approaches found in literature and practice. In summary, there are comparative results obtained from six different approaches.

Layered branch-and-bound algorithm without robustness The system in operation at the airport solves the flight gate assignment problem as a resource-constrained project scheduling problem with generalized precedence constraints. The same restrictions

and the objective as proposed in Sect. 2.1 are used with the exception that robustness is not considered. The main idea of the solution procedure is a branch-and-bound algorithm presented in Dorndorf et al. (2000) that has been limited to a series of search layers in order to be applicable to data sets of realistic size, see Dorndorf (2002).

Layered branch-and-bound algorithm with robustness The previous approach has been extended so that the robustness measure proposed by Bolat (2000) is included. Bolat (2000) has tried to allocate the total amount of idle time equally between the periods of assignment of aircraft to gates. Although starting and completion times are deterministic the results are meaningful comparative data as they describe the state-of-the-art application.

Deterministic Ejection Chain Algorithm without robustness These results of an Ejection Chain Algorithm applied to the CPP without any robustness measure were published by Dorndorf et al. (2008).

Deterministic Ejection Chain Algorithm with robustness Here, the CPP was enhanced considering the robustness goal described on page 6 which punishes small buffer times.

Hybrid meta-heuristic We have re-implemented the approach of Lim and Wang (2005) and adjusted it so that it is applicable to the data at hand. As mentioned before, gate closures, shadow restrictions, flight gate preferences and tows are not considered in their approach. An integration of these features to the tabu search, which is the first phase of their algorithm, was easy to achieve. However, the application of exchange cycles to the model at hand is not possible in a straightforward way. The exchange cycles are determined using an improvement graph which denotes the change of the objective if one flight activity replaces another activity at a different gate. Contrary to their approach, the reassignment of a flight activity also influences flight activities on neighboring gates due to shadow and tow time restrictions. The proposed heuristic for finding good exchange cycles therefore had to be adjusted which might have reduced its efficiency.

Although Lim and Wang (2005) propose various estimation functions for the probabilities of gate conflicts, we used the gamma distribution mentioned in Sect. 6.1 in order to get a fair comparison. We used same parameters for the tabu search and the exchange cycles. Both were stopped after 1000 continuous iterations without improvement. This led to rather high computation times (almost 90 min on average) which is most likely due to the higher number of flight activities (they tested an instance with 285 flights).

Two-stage flight gate assignment framework Yan and Tang (2007) propose a framework which they applied at Taiwan Taoyuan International Airport (formerly CKS airport). Although they use a very different gate assignment model (see Sect. 1), we were able to apply the framework to the CPP as follows. The main part of this framework is a CPP in which the edge weights are adjusted during the solution process. We will call this adjustable CPP or simply aCPP. Furthermore, 30 scenarios with exact arrival and departure times are determined according to Sect. 6.1. Using these scenarios, the edge weights of edges connecting two flight activities (which must not be arrival, parking,

or departure of the same aircraft) in aCPP are determined. For each scenario in which these flight activities overlap, the according edge weight is decreased by a penalty p. The remaining edge weights were used as proposed in Sect. 2.2. Afterwards the following procedure is repeated 10 times.

1. Solve the aCPP with the Ejection Chain Algorithm.
2. For each scenario, use the result of step 1 as initial solution of a deterministic CPP (as shown in Sect. 2.2 without robustness) and apply the EC algorithm for a recovery solution.
3. Determine the overall objective which is the sum of the objectives obtained in step 2. If the overall objective is improved, denote the current solution of aCPP as best solution.
4. Adjust the edge weights of aCPP as follows. For each scenario determine the flight activities that had to be reassigned. For each of these flight activities decrease the preference value to its originally assigned gate by penalty p.

We tried various values for p and with respect to the number of 30 scenarios $p = 1$ turned out to deliver best results. In their work, Yan and Tang (2007) used 40 scenarios. However, they considered a test set with 172 flights. They report computation times lasting more than an hour. Thus, we only used 30 scenarios leading to computation times of almost 6 h on average.

All these data are compared in two tests to the new results obtained by our algorithm using stochastic starting and completion times. The first test considers the planning stage in which an assignment of flights to gates has to be determined before the day of operation. The second test deals with the recovery of the flight gate assignment. However, as described earlier, an online recovery planning is sensitive to the operator's decisions, and our comparative data do not provide any information on these decisions. Thus, let us assume that at the beginning of the day of operation, all starting and completion times become known. They have been obtained from simulation. In the second test, we investigate the number of required recovery actions necessary for a feasible assignment.

Note that some adjustments have to be made in order to use the described test data. The test data include some mandatory dummy gate assignments which are caused by input data that is not applicable, such as a mandatory tow (national arrival, international departure), but the allowed time for towing is not sufficient for any gate combination. In order to get 'fair' results, we force these flight activities to be exceptionally assigned to the dummy gate, even in the stochastic model.

6.3 Planning stage tests

At first we have compared results for the mCPP to those obtained by the aforementioned algorithms. Figure 7 contains for every algorithm the sum of the different (stochastic) objectives over all 14 test records , i.e.

- the number of dummy gate assignments $z_1(I)$,
- the flight/gate preference scores $z_1(II)$ without dummy gate assignments,
- the number of tows z_2,

Data	$z_1(I)$	$z_1(II)$	z_2	z_3	z_4	z_5	z_6	$g(f)$
Layered B&B without robustness	33	-2610.6	612	697.91	0	33.94	12.99	50852.2
Layered B&B with robustness	42	-2443.5	1003	230.80	0	96.33	31.49	18573.2
Deterministic EC without robustness	26	-2560.9	607	669.95	0	40.68	14.8	51160.5
Deterministic EC with robustness	25	-2472.7	849	118.61	0	101.34	23.27	19034.9
Hybrid Meta-Heuristic	7	-2411.6	1584	101.65	0	2.83	68.34	8736.5
Two Stage Framework	7	2448.2	1403	61.87	0	1.32	135.1	10665.6
Stochastic EC	7	-2407.3	1174	82.32	0	1.23	39.88	6554.7

Fig. 7 Flight gate assignment in the planning stage

- the expected number of overlaps z_3,
- the expected number of violations of gate closures z_4,
- the expected number of violations of shadow restrictions z_5,
- the expected number of violations of tow time restrictions z_6,
- the total objective score $g(f)$.

Let us first consider the layered branch-and-bound algorithm without robustness. If we add up the robustness objective values $z_3, \ldots, z_6$, we get on average 53 constraint violations per day.

This number was confirmed to be realistic for the considered airport, which makes us assume that the distribution function is reasonably chosen. It is obvious that the introduction of the robustness measure significantly improves the results of the layered branch-and-bound algorithm. The expected number of violations is less than half as big.

There are similar observations for the Ejection Chain Algorithm where the robustness measure (in this case the punishment of small buffer times) decreases the expected number of constraint violations drastically. With respect to the overall objective the Ejection Chain Algorithm scores in both cases worse than the layered branch-and-bound algorithm, with or without robustness. This is remarkable since the deterministic Ejection Chain Algorithm is better if no robustness objective is considered (cf. Dorndorf et al. 2008).

Notice, that both robustness measures focus on gate overlaps, and only their number decreases whereas, e.g., the expected number of shadow restrictions considerably increases. The observation is different for approaches considering stochastic flight delays. Their robustness objective decreases tremendously, in spite of the fact that all flight activities—except the mandatory dummy gate assignments—were assigned to real gates. This happens at the cost of a higher number of tows which also leads to an increase of the expected number of tow time violations. For the Stochastic EC, the expected number of violations per day drops below 9. The results show that the number of problems at airports caused by earliness and tardiness of aircraft can significantly be reduced even in comparison with deterministic results in which established robustness measures are used. For each record of the Stochastic EC the runtime on a Pentium M, 1.99 GHz under Windows XP, is on average 316 s and is always below 8 min.

6.4 Recovery stage tests

Now, after analyzing the planning stage we will investigate the case when all starting and completion times become known and potential violations of restrictions are visible

Data	$E(z_3)$	$E(z_4)$	$E(z_5)$	$E(z_6)$	z_3	z_4	z_5	z_6
1_08	3.5974	0	0.0092	3.8169	3.87	0	0.00	3.98
1_09	3.4410	0	0.0000	2.5561	3.52	0	0.00	2.50
1_10	2.6276	0	0.0052	2.4607	2.66	0	0.00	2.28
1_11	6.7959	0	0.0346	3.4528	6.67	0	0.00	3.20
1_12	7.2242	0	0.3809	2.2352	7.83	0	0.32	2.33
1_13	5.9038	0	0.0068	3.3589	5.77	0	0.02	3.35
1_14	0.7967	0	0.0002	2.1629	0.71	0	0.00	1.99
2_03	3.7559	0	0.0176	2.5602	3.89	0	0.04	2.64
2_04	6.9532	0	0.1672	2.9979	7.20	0	0.20	3.14
2_05	9.2989	0	0.2367	3.8564	9.32	0	0.24	3.94
2_06	12.6580	0	0.0395	3.7817	12.59	0	0.04	3.58
2_07	9.7530	0	0.1535	1.9039	9.69	0	0.14	2.12
2_08	8.4383	0	0.1461	3.3600	7.77	0	0.16	3.68
2_09	1.0768	0	0.0283	1.3719	1.23	0	0.00	1.44
Σ	82.3207	0	1.2258	39.8755	82.72	0	1.16	40.17

Fig. 8 Comparison of expected violations and real violations after 100 simulations

and have to be resolved. Again, we used the data set described in Fig. 4. For each test record we performed 100 simulations with respect to the known distribution functions of all starting and completion times and received 100 different time tables for every day. They are the basis for our recovery tests.

In a first step, we determine the real number of violations in each scenario. If 100 simulations are significant, the numbers of expected violations (last row of Fig. 7) and real violations have to be close. Figure 8 contains the expected number of violations $E(z_i)$. The average number of real violations per instance from the simulations are listed in the last four columns. We can see that the total number of real violations differs by less than 0.51 % only.

For each data set obtained through the 100 simulations we determine feasible assignments of flights to gates, i.e. no overlaps at a gate and no violations of gate closure restrictions, shadow restrictions, and tow time restrictions are allowed. Therefore, we solve the deterministic problem with respect to a reference assignment, i.e. the one from the planning stage, cf. Dorndorf et al. (2012). We take each of the seven assignments of Fig. 7 into account. Although the reference assignments are obtained using various algorithms we now solve the recovery problem exclusively using the Ejection Chain Algorithm. Since there might be mandatory infeasibilities, we allow assignments to a dummy gate again, which of course will be penalized in the objective.

The objective function to be minimized consists of dummy gate assignments, negated preference scores, number of tows, and the deviation from the reference assignment. The latter is weighted with the priority value for each flight activity, in case it is assigned to a different gate than the one in the reference assignment. Results are summarized in Fig. 9 where the columns correspond to

- the number of dummy gate assignments $z_1(I)$.
- the flight / gate preference scores $z_1(II)$.
- the number of tows z_2.

Data	$z_1(I)$	$z_1(II)$	z_2	Deviation	$g(f)$
Layered B&B without robustness	33.42	-2586.3	1011.7	897.1	21610.0
Layered B&B with robustness	36.48	-2445.1	1184.3	538.4	8974.1
Deterministic EC without robustness	32.84	-2539.6	996.2	879.3	21025.0
Deterministic EC with robustness	28.87	-2469.7	922.6	402.7	10246.9
Hybrid Meta-Heuristic	39.62	-2406.6	1538	297.0	7987.9
Two Stage Framework	33.21	-2448.7	1260	307.4	7071.6
Stochastic EC	34.07	-2404.3	1177.1	199.2	6803.8

Fig. 9 Flight gate assignment in the recovery stage

- the number of deviations from the reference assignment.
- the total objective score $g(f)$.

All calculations are repeated 100 times for each record, therefore Fig. 9 shows the mean values over 100 runs, which are finally summed up over all 14 days.

Obviously, each robustness measure effectively improves its non-robust counterpart. Especially the number of recovery actions needed, i.e. the number of deviations from the original plan is reduced tremendously. In general, we get similar results as in the planning stage. However, it is interesting to see that the non-robust Ejection Chain Algorithm is now slightly better than the non-robust layered branch-and-bound algorithm. It was just the opposite in the planning stage. Same holds true for the two-stage framework which is now considerably better than the hybrid meta-heuristic. It seems to pay off that the framework takes the different recovery scenarios already during the planning stage into account.

As predicted in the planning stage, the stochastic EC gets along with least recovery actions and has the best objective function value. However, the improvement compared to the other robustness measures is not nearly as significant as the improvement compared to non-robust results. This leads to the conclusion that any of the tested robustness measures certainly improve flight gate assignments while the consideration of stochastic arrival and departure times as done in the last three approaches is the best. Among these three methods, the stochastic EC turns out to be best. Note that the hybrid meta-heuristic and the two-stage framework were designed for slightly different problems. As their results are worse than the ones of the stochastic EC, this only shows the inferiority of these approaches for the considered airport. At other airports the validity needs to be further investigated.

7 Summary

We have presented more realistic extensions to the well-known flight gate assignment problem. Stochastic starting and completion times of flight activities are included and the problem is modeled as a clique partitioning problem. This leads to a measure of robustness where the expected number of gate conflicts is minimized. In our extensions we have minimized the expected number of violations of any kind of constraints, e.g., shadow restrictions. The model allows abdicating the dummy gate which is used in other models for many airports. Furthermore, an online decision support system has been presented proposing recovery actions for resolving constraint violations.

Finally, we tested this model on real-life test data and compared it to other approaches and different robustness measures. For the tests, statistical data on the arrival time distributions of flights were used. However, little is known about dependencies of these distributions. Acknowledging dependencies in these distributions could result in even more accurate solutions, which is therefore an interesting topic for future research.

References

Bolat A (2000) Procedures for providing robust gate assignments for arriving aircraft. Eur J Oper Res 120:63–80

Castaing J, Mukherjee I, Cohn A, Hurwitz L, Nguyen A, Müller JJ (2016) Reducing airport gate blockage in passenger aviation: models and analysis. Comput Oper Res 65:189–199

Diepen G, Akker J, Hoogeveen J, Smeltink J (2012) Finding a robust assignment of flights to gates at Amsterdam Airport Schiphol. J Sched 15(6):703–715

Ding H, Lim A, Rodrigues B, Zhu Y (2004) New heuristics for the overconstrained airport gate assignment problem. J Oper Res Soc 55:760–768

Dorndorf U (2002) Project scheduling with time windows: from theory to application. Physica, Heidelberg

Dorndorf U, Pesch E (1994) Fast clustering algorithms. ORSA J Comput 6:141–153

Dorndorf U, Pesch E, Phan-Huy T (2000) A time-oriented branch-and-bound algorithm for resource constrained project scheduling with generalised precedence constraints. Manag Sci 46:1365–1384

Dorndorf U, Drexl A, Nikulin Y, Pesch E (2007) Flight gate scheduling: state-of-the-art and recent developments. Omega 35:326–334

Dorndorf U, Jaehn F, Pesch E (2008) Modelling robust flight gate scheduling as a clique partitioning problem. Transp Sci 42:292–301

Dorndorf U, Jaehn F, Pesch E (2012) Flight gate scheduling with respect to a reference schedule. Ann Oper Res 194:177–187

Drexl A, Nikulin Y (2006) Fuzzy multicriteria flight gate assignment. Working paper no. 605, University of Kiel

Grötschel M, Wakabayashi Y (1989) A cutting plane algorithm for a clustering problem. Math Program B 45:52–96

Grötschel M, Wakabayashi Y (1990) Facets of the clique partitioning polytope. Math Program A 47:367–387

Guépet J, Acuna-Agost R, Briant O, Gayon J-P (2015) Exact and heuristic approaches to the airport stand allocation problem. Eur J Oper Res 246(2):597–608

Hassounah M, Steuart G (1993) Demand for aircraft gates. Transp Res Rec 1423:26–33

Jaehn F, Pesch E (2013) New bounds and constraint propagation techniques for the clique partitioning problem. Discrete Appl Math 161(13):2025–2037

Kim SH, Feron E, Clarke J-P (2013) Gate assignment to minimize passenger transit time and aircraft taxi time. J Guid Control Dyn 36(2):467–475

Kumar V, Bierlaire M (2014) Multi-objective airport gate assignment problem in planning and operations. J Adv Transp 48(7):902–926

Lim A, Wang F (2005) Robust airport gate assignment. In: ICTAI '05: proceedings of the 17th IEEE international conference on tools with artificial intelligence, Washington, DC, USA. IEEE Computer Society, pp 74–81

List GF, Wood B, Nozick LK, Turnquist MA, Jones DA, Kjeldgaard EA, Lawton CR (2003) Robust optimization for fleet planning under uncertainty. Transp Res Part E: Logist Transp Rev 39(3):209–227

Mulvey JM, Vanderbei RJ, Zenios SA (1995) Robust optimization of large-scale systems. Oper Res 43(2):264–281

Neuman UM, Atkin JA (2013) Airport gate assignment considering ground movement. Lect Notes Comput Sci 8197:184–198

Nikulin Y (2006) Robustness in combinatorial optimization and scheduling theory: an extended annotated bibliography. Working paper no. 606, University of Kiel

Nikulin Y, Drexl A (2010) Theoretical aspects of multicriteria flight gate scheduling: deterministic and fuzzy models. J Sched 13(3):261–280

Ravizza S, Atkin J, Burke E (2014) A more realistic approach for airport ground movement optimisation with stand holding. J Sched 17(5):507–520

Richter U (2005) Analyse und Simulation von FlugverspStungen zur robusten Schichtplanung von Abfertigungsdiensten auf FlughSfen. Master's thesis, Department of Mathematics, RWTH Aachen University

Şeker M, Noyan N (2012) Stochastic optimization models for the airport gate assignment problem. Transp Res Part E: Logist Transp Rev 48(2):438–459

Yan S, Tang C (2007) A heuristic approach for airport gate assignments for stochastic flight delays. Eur J Oper Res 180:547–567