Apron Layout Planning - Optimal Positioning of Aircraft Stands

Thomas Hagspihl[a,d], Rainer Kolisch[a,e], Pirmin Fontaine[b,f], and Sebastian Schiffels[c,g]

Abstract

Airlines and airports share a common interest in maximizing the share of air traffic that is processed at contact gates instead of remote parking positions. Handling aircraft at contact gates is more efficient as passengers and cargo do not have to be conveyed between the terminal building and parked aircraft by buses. More efficient operations allow for shorter connection times, and thus enhance the network performance of airlines. Furthermore, the employment of passenger boarding bridges at contact gates makes embarking and disembarking processes more comfortable for passengers, adding to an improved overall passenger perception. To that end, we optimize the layout of aircraft parking positions adjacent to an airport terminal with given shape and dimensions, pursuing two lexicographically ordered objectives. First, we minimize the number of aircraft that have to be diverted to remote parking positions, because positions adjacent to the terminal are not available. Second, we minimize the construction effort required for gate infrastructure. Collisions of parked aircraft must be prevented at all times, and we consider various traffic situations, as traffic volume and fleet mix are not constant in time. We introduce the Airport Gate Layout Problem and formulate it as a mixed-integer model, which considers both greenfield and brownfield scenarios. To solve the problem efficiently, we introduce a decomposition framework that exploits the structure of the problem and employ various acceleration techniques. Our approach reduces computation times substantially, allowing us to solve instances that are intractable for CPLEX. Based on a case study for Munich Airport, we demonstrate how airports can gain valuable insights from solving the problem.

Keywords: Transportation; Airport Design; Airport Operations; Integer Programming; Decomposition

[a]Technical University of Munich, TUM School of Management, Arcisstraße 21, 80333 Munich, Germany
[b]Catholic University of Eichstätt-Ingolstadt, Ingolstadt School of Management, Auf der Schanz 49, 85049 Ingolstadt, Germany
[c]University of Augsburg, Institute of Business Administration, Universitätsstr. 16, 86159 Augsburg, Germany
[d]Corresponding author; thomas.hagspihl@tum.de
[e]rainer.kolisch@tum.de
[f]pirmin.fontaine@ku.de
[g]sebastian.schiffels@uni-a.de

1 Introduction

At many airports, space on the apron is scarce and should therefore be used as efficiently as possible (see, e.g., Caves 1994). We consider the situation where an airport terminal is to be either constructed, extended, or refurbished and the arrangement of aircraft parking positions adjacent to the terminal building needs to be determined. Compared to remote parking positions, the use of parking positions adjacent to a terminal building is preferred by airlines, airports, and passengers as operations are safer, more efficient, and more comfortable. Passengers and their baggage do not need to be transported over longer distances and turnaround times are shorter. Consequently, the apron layout should be designed to minimize the number of aircraft that have to be handled at remote parking positions for space reasons.

When planning the layout, minimum safety distances between adjacent aircraft as well as aircraft and airport infrastructure must be adhered to in order to prevent collisions (International Civil Aviation Organization (ICAO) 2018, European Aviation Safety Agency (EASA) 2017). Furthermore, there is a wide variety of aircraft types, which differ considerably with regard to their dimensions, minimum safety distances, and requirements for parking position equipment. Hence, the equipment to be installed at a parking position as well as the distances to adjacent parking positions are based on the aircraft to be handled at the position, and the optimal overall layout depends on the expected fleet mix at the airport. Moreover, traffic volume and fleet mix are subject to considerable fluctuations at most airports, especially at large hub airports. Thus, airports are keen to identify the layout that minimizes the number of aircraft having to divert to remote parking positions across all expected traffic situations. On the other hand, the (re-)construction of a parking position adjacent to a terminal building is associated with high costs. For example, investment costs of 450.000€ are to be expected per passenger boarding bridge (see, e.g., Airport Improvement Magazine 2010, Travel PR News 2019). Consequently, among all alternatives that minimize the number of aircraft that have to be handled at remote parking positions, airports are particularly interested in determining the layout that minimizes the number of parking positions that need to be built.

Currently, the planning problem described is mostly approached manually, with CAD programs and simulation tools supporting the decision makers. However, while different layout proposals can be evaluated relative to each other using simulation, there is currently a lack of possibility to assess a proposal on its own, since the optimal layout is not known. Hence, the quality of the resulting layout strongly depends on the experience and skills of the planner. We address this gap by finding the best solution for a particular apron.

Layout planning problems in general have been considered from various perspectives in the literature. Most prominently, existing work on the facility layout problem addresses the general question of how individual facilities should be arranged within a given area, usually a factory floor, in order to minimize the overall transportation costs of goods. Recent surveys on this class of problems and its variations are provided by Drira, Pierreval, and Hajri-Gabouj (2007) and Anjos and Vieira (2017). More application-specific layout planning problems are considered by

Briskorn and Dienstknecht (2019) and Huang, Wong, and Tam (2011), who investigate the optimal positioning of tower cranes at construction sites, as well as Stephan, Weidinger, and Boysen (2021), who maximize the capacity of parking lots. However, due to the complex geometries of aircraft, the heterogeneity of different aircraft types, and the requirement that the layout be optimal with respect to the totality of different traffic situations, none these approaches can be applied to our problem.

Thus far, no studies have been published on optimizing the layout of parking positions at airport aprons. There are several studies investigating apron capacity, which either assume the layout is given or disregard it. For example, Mirkovic and Tosic (2014, 2016, and 2017) provide a mathematical framework to describe the capacity of a given apron, and Steuart (1974), Bandara and Wirasinghe (1989), Wirasinghe and Bandara (1990), Hassounah and Steuart (1993), and Narciso and Piera (2015) present approaches to calculate the number of aircraft parking positions a terminal should be equipped with.

A related problem on the operative level of decision making that considers airport gates is the Gate Assignment Problem, in which arriving aircraft must be assigned to aircraft stands. Typical objectives are to minimize the total passenger walking distance or to minimize the total aircraft taxi time. Daş, Gzara, and Stützle (2020) find that recent work tends to consider multiple objectives simultaneously. Extensive reviews on the gate assignment problem are provided by Dorndorf et al. (2007), Cheng, Ho, and Kwan (2012), Guépet et al. (2015), and Daş, Gzara, and Stützle (2020). Dorndorf, Jaehn, and Pesch (2008, 2012, and 2017) also consider the prevention of collisions between parked aircraft in their models. However, in contrast to the recurrent operative task to assign arriving aircraft to available gates, defining the layout represents a strategic problem.

In conclusion, no literature exists to date in which the layout of aircraft parking positions is optimized by means of mathematical programming, nor can other existing modeling approaches be directly applied to this problem. In the following, we introduce all aspects of the Airport Gate Layout Problem (AGLP) in detail and provide a mixed-integer formulation that can be applied to both greenfield and brownfield instances. We develop an exact procedure for solving the problem, in which we combine a decomposition approach with a bounding algorithm and a relaxation scheme. In a case study, we demonstrate the superiority of our solution procedure over CPLEX and show how the results of our work can support decision makers in practice.

The remainder of this paper is structured as follows: In Section 2, we provide a detailed problem description, and our modeling approach is described in Section 3. The resulting mathematical model and our solution methodology are provided in Sections 4 and 5, respectively. In Section 6, we present our computational experiments based on real world data, and we provide our conclusions in Section 7.

2 Problem Description

Planning problem We consider the situation where an airport terminal is to be constructed, extended, or refurbished and the aircraft parking positions adjacent to the terminal building must be planned. The position, shape, and dimensions of the terminal building and surrounding taxiways are given and define the areas where parking positions can be planned. Furthermore, a forecast for the traffic that needs to be accommodated at the apron is available. In this planning problem, we pursue two objectives, which are ordered lexicographically. The primary goal is to minimize the number of aircraft that cannot be parked in close proximity to the terminal building; the secondary goal is the minimization of the implementation costs of the layout. That is, if a new terminal or a terminal extension is in planning, the number of parking positions to be built should be minimized. If an existing terminal is to be refurbished, as few changes as necessary should be made to the existing layout.

Aircraft parking positions We will use the following terminology with respect to aircraft parking. The terms *gate* and *aircraft stand* refer to a subarea of the terminal apron utilized to accommodate an aircraft and are used synonymously. Aircraft stands in the immediate vicinity of the terminal building are called *contact stands*, while stands at a greater distance from the terminal are referred to as *remote stands*.

Gates need to be equipped to handle aircraft, with equipment specifications depending on the aircraft types to be accommodated at the gate. For example, a gate at which small aircraft are to be handled must be equipped with a single passenger boarding bridge. At gates where large aircraft are to be handled, however, a second (or even a third) passenger boarding bridge must be installed to guarantee appropriate passenger boarding and deboarding times. The equipment installed at a gate is *downward compatible*. In other words, if an aircraft of a certain size can be handled at a gate, other aircraft of the same size and all smaller aircraft can also be handled. Furthermore, gates can also be equipped to handle either one large aircraft or *two* small aircraft simultaneously. Such gates are referred to as *Multi Aircraft Ramping Stands* (MARS) (see NASEM 2010).

Due to the large number of infrastructure elements to be procured (for example, passenger boarding bridges, fuel, power and fresh air connections, ground markings, equipment inside the terminal), the construction of a new gate is expensive. In addition, since more or larger equipment is needed to handle larger aircraft, the investment costs per gate increase with the size of the aircraft to be handled. As a result, the number of gates to be built or rebuilt, especially those capable of handling large aircraft, should be minimized as long as this does not reduce the airport's ability to handle air traffic.

Aircraft handled at contact stands are commonly parked in the nose-in orientation, in which the noses of parked aircraft point toward the terminal facade. The exact parking positions of aircraft at gates are defined by *lead-in lines* and *stop lines*: When parked, the fuselage of the aircraft is aligned collinearly with the lead-in line and the nose wheel is at the stop line belonging to the aircraft type. Each lead-in line is assigned to exactly one gate, whereas each gate must have at

least one lead-in line. Lead-in lines can be placed at any angle to the terminal facade. Although a greater number of lines per gate increases flexibility in terms of parking aircraft at gates, as few lead-in lines as possible should be used to minimize complexity in daily operations[1]. Lead-in lines are typically linear and can therefore be defined by a starting point and an ending point. Figure 1 shows the ground layout of three gates at the airport of Nice, where gate 46 is a MARS position.

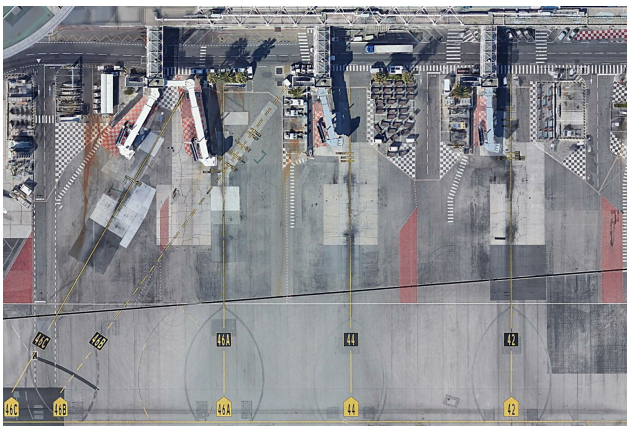


Figure 1: Gates 42, 44, and 46 at the airport of Nice (Source: Google Maps)

Aircraft classes To ensure collision-free operations on the apron, the geometries of the aircraft to be handled must be taken into account in the planning process. Because of the large number of aircraft types, planning is simplified by grouping aircraft into size classes. We will use the Aerodrome Reference Code (ARC) for that purpose, which is reproduced by ICAO (2018) and EASA (2017) and classifies aircraft according to wingspan, see Table 1[2].

Table 1: Classification of aircraft (ICAO 2018 and EASA 2017)

ARC letter	Wingspan [m]	Safety clearance [m]	Reduced safety clearance [m]
A	< 15	3	3
B	15 - 24	3	3
C	24 - 36	4.5	3
D	36 - 52	7.5	4.5
E	52 - 65	7.5	4.5
F	65 - 80	7.5	4.5

Aircraft parking restrictions There are few regulations regarding the design of aircraft stands. Most importantly, ICAO (2018) and EASA (2017) consistently define minimum safety clearances that must always be maintained between an aircraft and any other aircraft as well as airport structures. The minimum safety clearances depend on the ARC letter of the particular aircraft,

[1]MARS positions must have at least two lead-in lines to accommodate two small aircraft simultaneously.
[2]The same classification is introduced by FAA (2012) using a different notation.

and are reduced at aircraft stands equipped with a visual guidance docking system or other special cases apply. General and reduced safety clearances per ARC letter are provided in Table 1.

Parking an aircraft of a given class at a specific lead-in line may be prohibited for three reasons. First, certain areas of the apron may be inaccessible to aircraft of a certain class and above. Second, the location of a lead-in line, in combination with the layout of the surrounding infrastructure, might result in only aircraft up to a certain class being allowed to park on the line, as otherwise minimum safety clearances between aircraft and infrastructure would be violated or the aircraft would collide with the infrastructure. Third, the parking of an aircraft of a given class on a particular lead-in line could be temporarily prohibited if another aircraft of the same or a different class is being handled simultaneously at an adjacent position. The reason for this restriction is again to avoid violations of minimum safety clearances as well as collisions.

Air traffic characteristics at hub airports Traffic volumes and fleet mix are typically subject to significant fluctuations during the course of the day at hub airports. Traffic arrives and departs in waves, so airlines can offer their passengers comparatively short transfer times and a large number of transfer connections. Arrival and departure waves directly propagate to the situation on the apron, where the number and composition of the aircraft to be handled simultaneously can fluctuate considerably. To avoid recurring congestion, the apron should therefore be designed to handle as much of the expected peak-time traffic as possible at contact gates.

Moreover, the apron should not only be designed for current traffc peaks, but in anticipation of expected future traffic developments due to the high costs associated with gate infrastructure as well as the high operational effort and financial burden in case of future changes to the layout. For example, the proportion of very large ARC letter F aircraft is likely to decline in the medium term, as both Airbus and Boeing recently ceased production of their only ARC letter F aircraft, the A380 and 747-8, respectively, due to lack of demand. In contrast, the proportion of slightly smaller ARC letter E aircraft can be expected to trend upward due to their better fuel efficiency and greater operational flexibility.

3 Modeling Approach

Representation of air traffic We model air traffic on the apron analogous to Hagspihl et al. (2022); i.e., we represent air traffic as a collection of snapshots, each containing the number of aircraft per class that must be handled simultaneously at one particular point in time where traffic volume reaches a peak. We refer to these snapshots as *demand patterns*.

Demand patterns are extracted from a flight plan as follows. We classify aircraft based on their respective ARC letters, resulting in the set of aircraft classes $\mathcal{A} = \{1, \ldots, A\}$, where aircraft classes are sorted by ascending size. Using the flight schedule, we then determine the number of aircraft per class to be parked simultaneously over time. Let D_{at} denote the number of aircraft of class $a \in \mathcal{A}$ to be handled simultaneously at a given point in time $t \in \mathcal{T}$. We create a demand

pattern $(D_{1t}, D_{2t}, \ldots, D_{At})$ for time t if there is no other time $\hat{t}$ in the schedule that dominates t, i.e., at which $D_{at} \leq D_{a\hat{t}}\ \forall a \in \mathcal{A}$ and $\exists a \in \mathcal{A} : D_{at} < D_{a\hat{t}}$ holds. The set of all demand patterns resulting from that process is denoted as $\mathcal{K} = \{1, \ldots, K\} \subseteq \mathcal{T}$. In the following, D_{ak} represents the number of aircraft of class $a \in \mathcal{A}$ that are to be parked simultaneously for demand pattern $k \in \mathcal{K}$. Each value of D_{ak} is associated with a weighting factor $W_{ak} \in [0, 1]$, which indicates the relative importance of accommodating aircraft of class $a \in \mathcal{A}$ for demand pattern $k \in \mathcal{K}$. The value of W_{ak} depends on the size of the aircraft in class $a \in \mathcal{A}$ and the frequency of occurrence of demand pattern $k \in \mathcal{K}$. The larger an aircraft, the more passengers it can carry and the more important it is that it can be parked at a contact gate. Hence, larger aircraft classes are associated with higher values of W_{ak}. The more frequently a demand pattern is expected to occur, the more relevant it is for the planning process. Thus, demand patterns with a higher expected frequency of occurrence are associated with higher values of W_{ak} as well.

Aircraft parking positions Aircraft must be parked on lead-in lines, and each lead-in line used to park an aircraft must be assigned to a gate. We employ the given layout of the terminal building and the surrounding taxiways to derive sets of feasible positions for gates and lead-in lines in the following.

In the first step, we identify polygons as subareas of the apron inside which lead-in lines can be placed. The edges of these polygons are given either by airport infrastructure that must not be infringed upon by aircraft (for instance, the terminal facade or edges of the apron), or by taxiways. Second, in each polygon we define a set of *starting points* for lead-in lines; by starting point we mean the point of the lead-in line at which the nose of a parked aircraft is located. The starting points are placed at a constant distance $\Delta > 0$ from each other and at a constant safety distance from the terminal facade.

Third, we create lead-in lines by drawing straight lines from the starting points until they intersect with an edge of the surrounding polygon. These intersections define the ending points of the lead-in lines. We generate multiple lead-in lines from each starting point by varying the direction of the line. Therefore, parameter κ denotes the rotation angle between the lead-in lines that share the same starting point, $0 < \kappa < 360°$ and $360°$ should be an integer multiple of κ. Both Δ and κ are parameters of our approach that determine the planning granularity and must be specified up-front. Lead-in lines whose ending point is not on a taxiway cannot be reached by aircraft and are therefore discarded. All remaining lead-in lines resulting from this process are included in set $\mathcal{L} = \{1, \ldots, L\}$. Figure 2 shows an example of a terminal building with possible lead-in lines.

Each lead-in line used to park an aircraft has to be assigned to a gate. Thus, we define potential gate positions and introduce a parameter that indiciates whether a lead-in line can be assigned to a gate or not. We consider the set of lead-in line starting points as the set of possible gate positions. All potential gates are contained in set $\mathcal{G} = \{1, \ldots, G\}$. For each pair of lead-in line $l \in \mathcal{L}$ and gate $g \in \mathcal{G}$, binary parameter F_{lg} equals 1, if the direct path between lead-in line l and gate g is

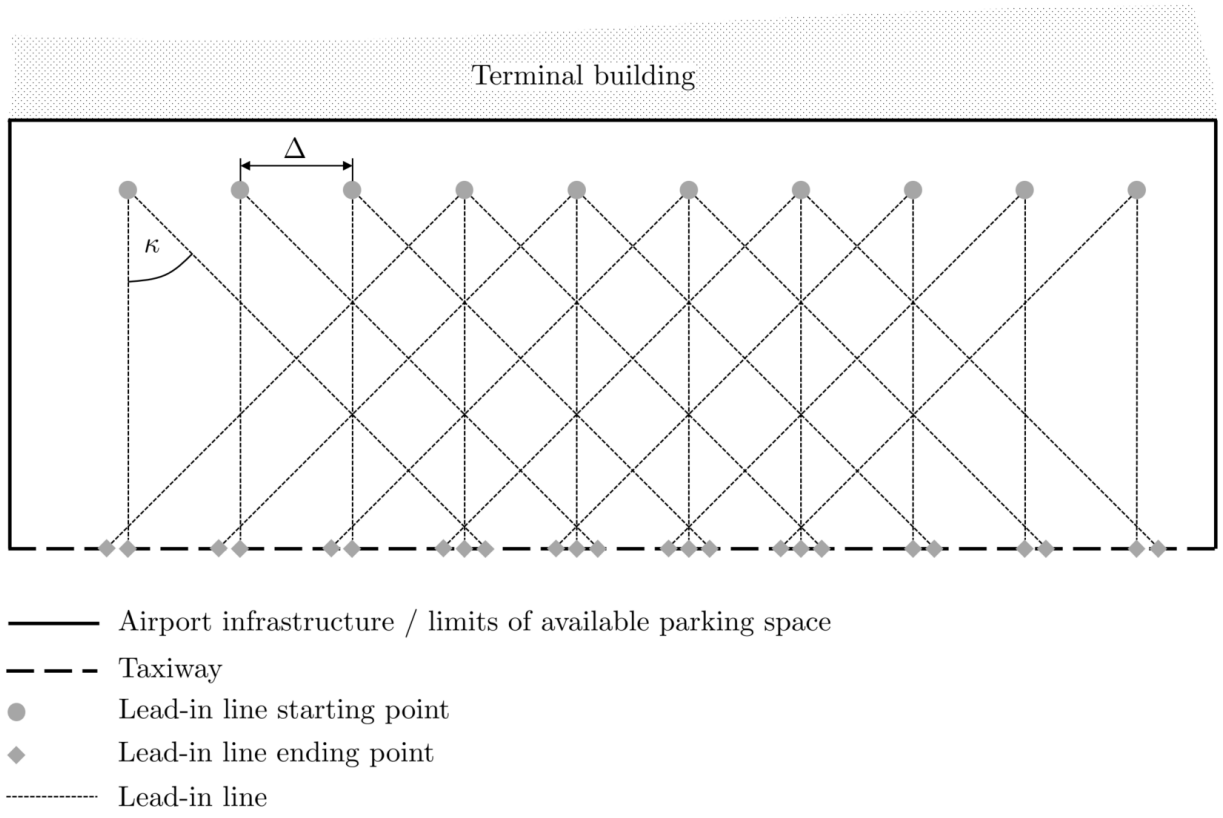


Figure 2: Example for generation of lead-in lines

unobstructed and smaller than a given threshold, and lead-in line l can therefore be assigned to gate g. Otherwise, F_{lg} equals 0.

Aircraft parking restrictions Parking an aircraft of a particular class on a particular lead-in line could be either permanently or temporarily prohibited to prevent collisions with airport infrastructure or other aircraft parked at adjacent positions. To identify all such infeasibilities, we apply a two-step procedure.

First, we compute a so-called *safety envelope* for each aircraft class. The safety envelope of an aircraft class $a \in \mathcal{A}$ is defined by the smallest possible polygon with the shape shown in Figure 3 inside which all aircraft of the class can be parked in a given orientation, including the minimum safety distances required according to Table 1. We compute the safety envelope for each aircraft class based on the geometry data of all aircraft types belonging to the class. The safety envelopes of all classes have the shape as shown in Figure 3 and differ only in the dimensions. Details are provided in Appendix B.

Second, for each aircraft class $a \in \mathcal{A}$ and lead-in line $l \in \mathcal{L}$, we use the safety envelopes to determine whether or not the aircraft can generally be parked on the line, and whether or not the aircraft can be parked on the line if another aircraft of class $b \in \mathcal{A}$ is parked on another lead-in line

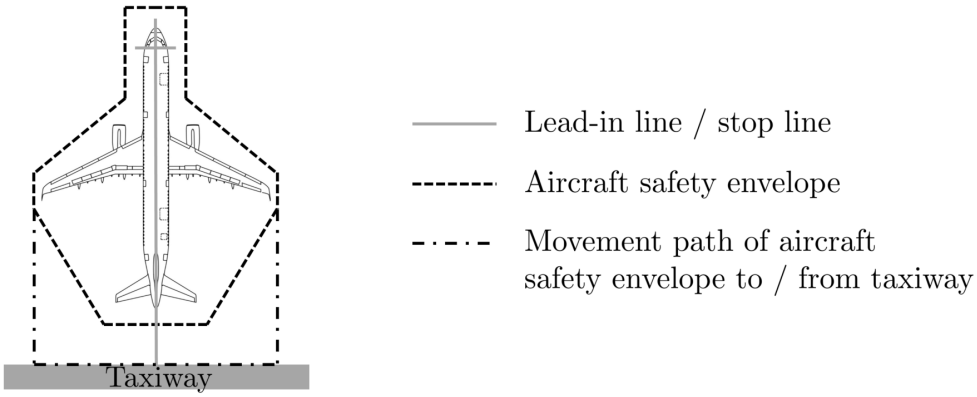


Figure 3: Shape of safety envelopes (Source of aircraft model: Airbus 2022)

$m \in \mathcal{L} \setminus \{l\}$ at the same time. If we place the safety envelope of an aircraft class on a lead-in line and the safety envelope then touches or intersects an edge of the surrounding polygon, i.e., collides with airport infrastructure, no aircraft of that class can be parked on the lead-in line. As a result, the binary parameter C_{al} equals 1 if aircraft of class $a \in \mathcal{A}$ can be accommodated on lead-in line $l \in \mathcal{L}$, and 0, otherwise. C_{al} also equals 0 if lead-in line l is located in an area of the apron that cannot be reached by class a aircraft. If $C_{al} = 0$ for a given lead-in line $l \in \mathcal{L}$ and all aircraft classes $a \in \mathcal{A}$, line l is discarded. Similarly, for all combinations of two lead-in lines $l, m \in \mathcal{L} : l \neq m$ and two aircraft classes $a, b \in \mathcal{A}$, we place the safety envelopes of classes a and b on lines l and m, respectively, and then examine whether the safety envelopes collide with each other (for details, we refer to Appendix C). Binary parameter E_{lmab} is 1 if aircraft of classes a and b cannot be handled simultaneously without collision at lead-in lines l and m, and 0, otherwise.

4 Model

In this section, we present a mixed-integer linear program for the AGLP. The notation is summarized in Table A.1 in Appendix A.

Decision variables First, the binary decision variable v_{ga} indicates whether or not gate $g \in \mathcal{G}$ is equipped to handle aircraft of class $a \in \mathcal{A}$. If $v_{ga} = 0$ for all aircraft classes $a \in \mathcal{A}$, gate $g \in \mathcal{G}$ cannot handle aircraft of any class and hence, is not built. Furthermore, from the downward compatibility of gate equipment it follows that if $v_{ga} = 1$, then $v_{g\hat{a}} = 1 \; \forall \hat{a} \in \mathcal{A} : \hat{a} < a$.

Second, the binary decision variable y_l equals 1 if lead-in line $l \in \mathcal{L}$ is used to park an aircraft for any demand pattern $k \in \mathcal{K}$, and 0, otherwise.

Third, the binary decision variable u_{gl} is 1 if lead-in line $l \in \mathcal{L}$ is assigned to gate $g \in \mathcal{G}$, and 0, otherwise.

Fourth, the binary decision variable x_{glak} equals 1 if for demand pattern $k \in \mathcal{K}$ an aircraft of class $a \in \mathcal{A}$ is parked at lead-in line $l \in \mathcal{L}$ and lead-in line l is assigned to gate $g \in \mathcal{G}$, and 0, otherwise.

Finally, the non-negative decision variable q_{ak} yields for each demand pattern $k \in \mathcal{K}$ the number of aircraft of class $a \in \mathcal{A}$ that cannot be accommodated at any of the lead-in lines in set $\mathcal{L}$, and hence, need to deviate to remote stands.

Model formulation We formulate the AGLP as follows:

$$\min \; z_1 = \sum_{a \in \mathcal{A}} \sum_{k \in \mathcal{K}} W_{ak} \cdot q_{ak} \tag{1a}$$

$$\min \; z_2 = \sum_{g \in \mathcal{G}} \sum_{a \in \mathcal{A}} v_{ga} \tag{1b}$$

subject to

$$\sum_{g \in \mathcal{G}} \sum_{l \in \mathcal{L}: C_{al}=1 \cap F_{lg}=1} x_{glak} + q_{ak} \geq D_{ak} \qquad \forall a \in \mathcal{A}; k \in \mathcal{K} \tag{1c}$$

$$\sum_{g \in \mathcal{G}: F_{lg}=1} \sum_{a \in \mathcal{A}: C_{al}=1} x_{glak} \leq y_l \qquad \forall l \in \mathcal{L}; k \in \mathcal{K} \tag{1d}$$

$$\sum_{g \in \mathcal{G}: F_{lg}=1} u_{gl} = y_l \qquad \forall l \in \mathcal{L} \tag{1e}$$

$$x_{glak} \leq u_{gl} \qquad \forall g \in \mathcal{G}; l \in \mathcal{L}; a \in \mathcal{A}; k \in \mathcal{K}: \; F_{lg} = 1; C_{al} = 1 \tag{1f}$$

$$\sum_{l \in \mathcal{L}: F_{lg}=1} \sum_{a \in \mathcal{A}: C_{al}=1} a \cdot x_{glak} \leq \sum_{a \in \mathcal{A}} v_{ga} \qquad \forall g \in \mathcal{G}; k \in \mathcal{K} \tag{1g}$$

$$v_{ga} \leq v_{gb} \qquad \forall g \in \mathcal{G}; a \in \{2, \ldots, A\}; \; b = a - 1 \tag{1h}$$

$$\sum_{l \in \mathcal{L}: F_{lg}=1} \left(\sum_{a \in \mathcal{A}^{\text{small}}: C_{al}=1} x_{glak} + \sum_{a \in \mathcal{A}^{\text{large}}: C_{al}=1} 2 \cdot x_{glak} \right) \leq 2 \qquad \forall g \in \mathcal{G}; k \in \mathcal{K} \tag{1i}$$

$$\sum_{g \in \mathcal{G}: F_{lg}=1} x_{glak} + \sum_{h \in \mathcal{G}: F_{mh}=1} x_{hmbk} \leq 1 \qquad \forall l, m \in \mathcal{L}; a, b \in \mathcal{A}; k \in \mathcal{K}: \; E_{lmab} = C_{al} = C_{bm} = 1; l < m \tag{1j}$$

$$q_{ak} \geq 0 \qquad \forall a \in \mathcal{A}; k \in \mathcal{K} \tag{1k}$$

$$v_{ga} \in \{0, 1\} \qquad \forall g \in \mathcal{G}; a \in \mathcal{A} \tag{1l}$$

$$x_{glak} \in \{0, 1\} \qquad \forall g \in \mathcal{G}; l \in \mathcal{L}; a \in \mathcal{A}; k \in \mathcal{K}: \; F_{lg} = 1; C_{al} = 1 \tag{1m}$$

$$y_l \in \{0, 1\} \qquad \forall l \in \mathcal{L} \tag{1n}$$

$$u_{gl} \in \{0, 1\} \qquad \forall g \in \mathcal{G}; l \in \mathcal{L}: F_{lg} = 1 \tag{1o}$$

The two lexicographically ordered objective functions are given in (1a) and (1b). Objective Function (1a) minimizes the number of aircraft that cannot be processed at any of the lead-in lines in set $\mathcal{L}$ over all demand patterns $k \in \mathcal{K}$, where the values of q_{ak} are weighted with weighting factors W_{ak}. Objective Function (1b) minimizes the number of gates to be built.

Demand constraints (1c) ensure for each aircraft class and demand pattern that either all aircraft are handled at contact gates or q_{ak} is increased accordingly. Constraints (1d)-(1h) determine the correct infrastructure decisions: Constraints (1d) make sure that the value of y_l equals 1 once lead-in line $l \in \mathcal{L}$ is used to park an aircraft. Constraints (1e) enforce that each lead-in line that is used to park an aircraft is assigned to exactly one gate. That is, while a lead-in line can be used to park an aircraft for more than one demand pattern, it has to be assigned to the same gate for all demand patterns. Constraints (1f) align the values of variables x_{glak} and u_{gl}, and Constraints (1g) ensure that gates are built and equipped for the aircraft classes they are supposed to handle. Note that when two small aircraft are handled simultaneously at one gate (MARS), the left side of the constraint takes both aircraft into account. For example, if two aircraft of class 1 are handled at a gate simultaneously, the left side of Constraints (1g) equals 2. Hence, the constraint necessitates that the gate be equipped for aircraft of class 2. This takes into account that gates at which two aircraft of a class are handled simultaneously require more equipment than gates at which only one aircraft of the same class is handled. Most obviously, two passenger boarding bridges need to be installed to handle two aircraft simultaneously. Constraints (1h) make sure that gate equipment is downward compatible, and Constraints (1i) enforce that a maximum of two aircraft (either two small aircraft or one large aircraft) can be handled at one gate simultaneously. For that purpose, we divide all aircraft classes from set $\mathcal{A}$ into two subsets $\mathcal{A}^{\text{small}}$ and $\mathcal{A}^{\text{large}}$, each containing the aircraft classes of which two and one aircraft, respectively, can be handled simultaneously at a gate. $\mathcal{A}^{\text{small}}$ contains ARC letter C aircraft and all smaller aircraft, while $\mathcal{A}^{\text{large}}$ contains all aircraft belonging to ARC letter D or larger. Safety constraints (1j) prevent minimum safety clearances from being violated or aircraft from colliding with each other. Finally, Constraints (1k)-(1o) define the domains of all decision variables.

Brownfield scenarios If an existing terminal is to be renovated or extended and existing parking positions can be relocated in the process, deviations from the existing gate layout should be kept to a minimum to minimize investment costs. More specifically, additional gates should only be built if they yield an improvement of z_1.

In a brownfield scenario, the set $\mathcal{G}$ is extended by the gates that already exist in reality. Additionally, we introduce the binary parameter H_{ga}, whose value is 1, if existing gate $g \in \mathcal{G}$ is equipped to handle an aircraft of class $a \in \mathcal{A}$, and 0, otherwise. To account for the downward compatibility of gates, $H_{g\hat{a}} = 1$ for all $\hat{a} \in \{0, \ldots, a-1\}$ if $H_{ga} = 1$. For all gates in $\mathcal{G}$ which do not yet exist in reality, H_{ga} equals 0 for all $a \in \mathcal{A}$.

Then, Model (1a)-(1o) is extended by the following constraints to ensure that existing gates are considered in the optimization process.

$$v_{ga} = H_{ga} \qquad \forall g \in \mathcal{G}; a \in \mathcal{A} : \sum_{\hat{a} \in \mathcal{A}} H_{g\hat{a}} > 0 \tag{1p}$$

Aggregating constraints For small distances Δ and rotation angles κ between adjacent lead-in lines, the number of Safety constraints (1j) becomes very large. We mitigate this problem by aggregating Constraints (1j) where possible. That is, omitting gates and demand patterns, we merge two Constraints (1j) $x_{la} + x_{mb} \leq 1$ and $x_{la} + x_{nc} \leq 1$ into one a new constraint $x_{la} + x_{mb} + x_{nc} \leq 1$, where $l, m, n \in \mathcal{L}$ and $a, b, c \in \mathcal{A}$. Note that the aggregated constraint is equivalent to the individual constraints only if the original Constraints (1j) contain a constraint $x_{mb} + x_{nc} \leq 1$. Otherwise, enforcing $E_{lnac} = 1$ would make the aggregated constraint more restrictive than the set of original constraints. We add further variables to the aggregated constraint, as long as for each pair of variables contained in the aggregated constraint there exists one original Constraint (1j) that contains both variables as well. With the remaining variables that could not be added to the aggregated constraint, we then try to construct an additional aggregated constraint employing the same procedure. Once the restriction represented by a constraint of type (1j) is reflected in one of the aggregated constraints, it is removed from the model. We create further aggregated constraints until all original Constraints (1j) are removed from the model. The algorithm itself is presented in Appendix D.

In addition to reducing the number of constraints needed, this formulation strengthens the LP relaxation of the problem, which reduces computation times considerably.

5 Solution Methodology

The results of our computational experiments in Section 6 will demonstrate that Model (1a)-(1p) quickly becomes intractable for decreasing values of Δ and κ. Therefore, we introduce a decomposition approach, which allows solving instances with high planning granularity within reasonable computation time to optimality. We will first give an overview in Section 5.1 before providing detailed insights into individual components in Sections 5.2 to 5.4. Finally, we will present the acceleration techniques that we use to support our approach in Section 5.5. The notation used in this section is summarized in Table A.2 in Appendix A.

5.1 Overview

We provide a summary of our solution approach as pseudo-code in Algorithm 1.

First, we decompose the apron into a set of areas $\mathcal{S} = \{1, \ldots, S\}$ that are independent with respect to Safety constraints (1j), intending to solve the AGLP for each area separately (line 1 in Algorithm 1, see Section 5.2); however, Demand constraints (1c) consider the apron as a whole. Thus, in order to solve the problem for each area independently, we must first decide for each demand pattern how many aircraft of which class(es) are to be assigned to which area.

Algorithm 1 Overview of our solution approach

```
1: Find set of areas 𝒮                                          ▷ See Section 5.2
2: Find set of demand decompositions 𝒞                          ▷ See Section 5.3
3: Determine initial value of LB_c for all c ∈ 𝒞                ▷ See Section 5.4
4: i ← 1
5: while | 𝒞 | > 1 do
6:    Select ĉ ∈ 𝒞 where LB_ĉ ≤ LB_c for all c ∈ 𝒞 \ {ĉ}
7:    Solve the subproblems associated with ĉ                   ▷ See Section 5.5
8:    if i = 1 then
9:       Determine UB based on the solution obtained
10:   else
11:      If possible, update UB and LB_c for all c ∈ 𝒞
12:   end if
13:   Eliminate all c ∈ 𝒞 where LB_c ≥ UB
14:   i ← i + 1
15: end while
```

Among the many possible ways to decompose demand patterns, we only consider those where (i) all aircraft assigned to an area can be parked at contact gates in the particular area and where (ii) the total number of aircraft (weighted by W_{ak}) that cannot be assigned to any area due to (i) is minimized. We call each of the resulting assignments a *demand decomposition*, and the set of demand decompositions is denoted by $\mathcal{C} = \{1, \ldots, C\}$ (line 2, see Section 5.3).

Let z_1^* and z_2^* denote the optimal values of Objective Functions (1a) and (1b), respectively. Similarly, let z_{1c}^* and z_{2c}^* be the optimal values of Objective Functions (1a) and (1b) when aircraft are assigned to areas according to demand decomposition $c \in \mathcal{C}$ ($z_{1c}^* \geq z_1^*$ and $z_{2c}^* \geq z_2^*$). As we will show in Section 5.3, ensuring (i) and (ii) is equivalent to minimizing Objective Function (1a), and thus all demand decompositions in $c \in \mathcal{C}$ lead to z_1^*, i.e., $z_{1c}^* = z_1^*$ $\forall c \in \mathcal{C}$. In order to solve the AGLP to optimality, we must identify a demand decomposition $c \in \mathcal{C}$ that also leads to z_2^*, i.e., where $z_{2c}^* = z_2^*$. We reduce the number of demand decompositions that need to be examined to find z_2^* and to prove optimality of z_2^* by means of a bounding algorithm (lines 5 to 15). First, we compute a lower bound LB_c on z_{2c}^* for each demand decomposition $c \in \mathcal{C}$ (line 3, see Section 5.4). Then, we select the demand decomposition with the lowest value of LB_c and determine z_{2c}^*. The result poses an upper bound UB on z_2^*. Thus, we can remove all demand decompositions from $\mathcal{C}$ for which $LB_c \geq UB$ holds (line 13). Furthermore, we will show in Section 5.4 that the solution may be used to update $LB_{c'}$ for other demand decompositions $c' \in \mathcal{C} \setminus \{c\}$, potentially leading to their removal from $\mathcal{C}$ as well. In the next iteration, we again select the demand decomposition $c \in \mathcal{C}$ with lowest value of LB_c. We repeat this process until no demand decomposition remains in $\mathcal{C}$. Then, the value of UB equals z_2^*.

To determine z_{2c}^* for one demand decomposition $c \in \mathcal{C}$, we solve Model (1b)-(1p) separately for each area $s \in \mathcal{S}$ (line 7, see Section 5.5). We call the resulting models the *subproblems* of

the AGLP, and we denote the optimal objective function value of the subproblem associated with demand decomposition $c \in \mathcal{C}$ and area $s \in \mathcal{S}$ with z^*_{2cs}. When the subproblems for all areas $s \in \mathcal{S}$ have been solved for demand decomposition $c \in \mathcal{C}$, we calculate z^*_{2c} as $\sum_{s \in \mathcal{S}} z^*_{2cs}$. We provide the mathematical formulation of a subproblem in Appendix E.

5.2 Decomposing the apron into independent areas

We partition the apron into independent areas $\mathcal{S}$ in an iterative process. A terminal facade can be described as a polygonal chain, consisting of a sequence of connected line segments. We initially assume that all gates and lead-in lines located before the same line segment are part of the same area, and that gates and lead-in lines located in front of different line segments belong to different areas. Let $\mathcal{G}_s$ and $\mathcal{L}_s$ be the resulting sets of lead-in lines and gates belonging to area $s \in \mathcal{S}$.

Definition 1 (**Independence of areas**)**.** Two areas $s, t \in \mathcal{S}, s \neq t$ are independent iff $E_{lmab} = 0 \; \forall l \in \mathcal{L}_s; m \in \mathcal{L}_t; a, b \in \mathcal{A}$.

If two areas $s, t \in \mathcal{S}, s \neq t$ are not independent according to Definition 1, we combine both areas into a new area u ($\mathcal{L}_u = \mathcal{L}_s \cup \mathcal{L}_t$, $\mathcal{G}_u = \mathcal{G}_s \cup \mathcal{G}_t$), add u to $\mathcal{S}$, remove s and t from $\mathcal{S}$, and check again whether all areas $s \in \mathcal{S}$ are mutually independent according to Definition 1. The process is finished when all areas in $\mathcal{S}$ are confirmed to be independent.

5.3 Determining the set of demand decompositions

We identify demand decompositions using a two-step procedure: First, we determine for each area $s \in \mathcal{S}$ how many aircraft of which classes *can* be parked simultaneously at lead-in lines $\mathcal{L}_s$.

Definition 2 (**Parking patterns**)**.** Let $r_a \in \mathbb{N}_0$ represent a number of class $a \in \mathcal{A}$ aircraft. We call $(r_1, r_2, \ldots, r_A)$ a feasible *parking pattern* for an area if all aircraft $r_1, r_2, \ldots, r_A$ can be parked at contact gates of the area simultaneously. Let $r_{ap} \in \mathbb{N}_0$ denote the number of aircraft of class $a \in \mathcal{A}$ that are parked simultaneously in a parking pattern p. We call $p = (r_{1p}, r_{2p}, \ldots, r_{Ap})$ an *efficient parking pattern* for an area if r_{ap} cannot be increased for any $a \in \mathcal{A}$ without leading to infeasibility with respect to Safety constraints (1j). The set of efficient parking patterns associated with area $s \in \mathcal{S}$ is denoted by $\mathcal{P}_s$.

Example 1. Let $\mathcal{A} = \{1, 2\}$, with $a = 1$ and $a = 2$ representing small and large aircraft, respectively. Assume that in a given area it is possible to park a maximum of four small aircraft when zero large aircraft are parked, and that a maximum of two large aircraft can be parked when zero small aircraft are parked. Furthermore, assume that two small aircraft can be parked when the number of large aircraft that are parked is one. Then, the set of efficient parking patterns for area $s \in \mathcal{S}$ is given as $\mathcal{P}_s = \{(4, 0), (2, 1), (0, 2)\}$.

Second, once $\mathcal{P}_s$ has been determined for all areas $s \in \mathcal{S}$, we create demand decompositions by selecting one parking pattern $p \in \mathcal{P}_s$ for each demand pattern $k \in \mathcal{K}$ and area $s \in \mathcal{S}$.

Definition 3 (**Demand decompositions**)**.** Let the function $p\left(c,s,k\right)$ return the parking pattern p that is selected from $\mathcal{P}_s$ for area $s \in \mathcal{S}$ and demand pattern $k \in \mathcal{K}$ in demand decomposition $c \in \mathcal{C}$. Then, demand decomposition $c \in \mathcal{C}$ is defined as

$$\begin{pmatrix} p\left(c,1,1\right) & p\left(c,1,2\right) & \dots & p\left(c,1,K\right) \\ p\left(c,2,1\right) & p\left(c,2,2\right) & \dots & p\left(c,2,K\right) \\ \vdots & \vdots & \vdots & \vdots \\ p\left(c,S,1\right) & p\left(c,S,2\right) & \dots & p\left(c,S,K\right) \end{pmatrix}.$$

Patterns are selected such that $\sum_{a\in\mathcal{A}}\sum_{k\in\mathcal{K}} W_{ak}\cdot\left(D_{ak} - \sum_{s\in\mathcal{S}} r_{ap(c,s,k)}\right)$ is minimized, which is equivalent to Objective Function 1a.

In the following, let $\mathcal{P}_{cs} = \left\{p\left(c,s,1\right), p\left(c,s,2\right), \dots, p\left(c,s,K\right)\right\}$ contain the parking patterns from the s-th row of demand decomposition $c \in \mathcal{C}$.

5.3.1 Identifying efficient parking patterns

To determine the set of efficient parking patterns $\mathcal{P}_s$ for area $s \in \mathcal{S}$, we employ a graph-based approach, which is inspired by existing work on identifying the set of pareto-efficient paths through a network (see, e.g., Martins 1984, Tung Tung and Lin Chew 1992) and dynamic programming.

We create a directed graph in which each feasible combination of lead-in line $l \in \mathcal{L}_s$ and aircraft class $a \in \mathcal{A}$ ($C_{al} = 1$) is represented by a node (l,a). Arcs between nodes represent feasible combinations of two parked aircraft, i.e., two nodes (l,a) and (m,b) are connected if $E_{lmab} = 0$. Arcs are directed from lead-in lines with lower indices to lead-in lines with higher indices. We add two dummy nodes to the network, one representing a single source and the other representing a single sink. For each non-dummy node, an in-arc is added from the source node and an out-arc is added to the sink node. Each path through this network corresponds to one feasible, yet not necessarily efficient parking pattern for the respective area. We construct paths incrementally by iterating over all nodes, starting at the source node. Each iteration consists of two steps: First, we compare all paths that lead into the particular node and prune paths if possible. Second, we extend all non-dominated paths leading into the node to all nodes, which can be reached from that node.

A path is pruned if it is dominated by another path (see Lemma 1) or it becomes apparent that the parking pattern associated with it is not needed given the demand patterns (see Lemma 2). Let $\sigma_{a\pi}^{(i)}$ denote the number of class $a \in \mathcal{A}$ aircraft parked in path π leading from the source node to node i.

Lemma 1 (**Path dominance**)**.** *A path π_1 dominates another path π_2 if $\sigma_{a\pi_1}^{(i)} \geq \sigma_{a\pi_2}^{(i)}$ $\forall a \in \mathcal{A}$ and $\sigma_{a\pi_1}^{(i)} > \sigma_{a\pi_2}^{(i)}$ $\exists a \in \mathcal{A}$.*

The proof directly follows from Definition 2. If paths $\pi_1, \pi_2, \dots, \pi_n$ are equivalent, i.e., $\sigma_{a\pi_1}^{(i)} = \sigma_{a\pi_2}^{(i)} = \dots = \sigma_{a\pi_n}^{(i)}$ $\forall a \in \mathcal{A}$, one path is selected arbitrarily and all other paths are pruned. □

Lemma 2 (**Paths and demand patterns**). *A path π can be pruned if $\sigma_{a\pi}^{(i)} > max_{k\in\mathcal{K}}\{D_{ak}\}$ $\exists a \in \mathcal{A}$ at a given node i.*

We consider situations where space is scarce, i.e., the optimal value of Objective Function (1a) is larger than 0 with respect to each demand pattern $k \in \mathcal{K}$. Thus, parking more class $a \in \mathcal{A}$ aircraft than is required for any of the demand patterns in $\mathcal{K}$ would translate to fewer aircraft of another class $b \in \mathcal{A}$ that can be parked and hence, would result in a worse outcome with respect to Objective Function (1a). □

Example 2. Assume there are $A = 2$ aircraft classes and demand patterns $(15, 0)$, $(13, 1)$, and $(10, 2)$. Then, a path π with $\sigma_{2\pi} = 3$ can be pruned, since no more than two aircraft of class 2 needs to be accommodated for any demand pattern.

Before we extend a path to node (m, b), we verify that parking a class b aircraft at lead-in line m does not lead to collisions with any other aircraft already considered in the path, i.e., that $E_{lmab} = 0$ for all nodes (l, a) already included in the path. While the definition of arcs ensures that this is always true for the node that was most recently added to the path, it may not be true for nodes added to the path earlier. Whenever we find that $E_{lmab} = 1$ for any node (l, a) already included in the path, we do not extend the path to node (m, b). We observe this phenomenon when the value of κ is small and/or when the terminal facade has a corner.

After the last iteration, i.e., when all paths have reached the sink node, none of the remaining paths is dominated by another path. Thus, each path that reaches the sink node is associated with a unique and efficient parking pattern for area s.

5.3.2 Creation of demand decompositions

A demand decomposition $c \in \mathcal{C}$ is created by selecting one parking pattern $p \in \mathcal{P}_s$ for each area $s \in \mathcal{S}$ and demand pattern $k \in \mathcal{K}$. Depending on the cardinalities of $\mathcal{P}_s$, $\mathcal{S}$, and $\mathcal{K}$, the number of possible demand decompositions may be very large. However, we are only interested in those demand decompositions that lead to z_1^*. Objective Function (1a) as well as demand decompositions can be decomposed with respect to demand patterns $k \in \mathcal{K}$. Let z_{1k}^* be the optimal value of Objective Function (1a) with respect to demand pattern $k \in \mathcal{K}$, i.e., $z_1^* = \sum_{k\in\mathcal{K}} z_{1k}^*$. In order to lead to z_1^*, a demand decomposition must lead to z_{1k}^* for each $k \in \mathcal{K}$; we use this by dividing the process to create demand decompositions into $K+1$ stages. In the k-th stage, we pretend that demand pattern k is the only demand pattern that exists and create the set of all possible demand decompositions, which we denote as $\mathcal{C}_k$. Following the matrix representation of demand decompositions as given in Definition 3, each demand decomposition $c \in \mathcal{C}_k$ is a column vector $(p(c,1,k), p(c,2,k), \ldots, p(c,S,k))^T$. Next, for each $c \in \mathcal{C}_k$ we calculate the associated value of Objective Function (1a) with respect to demand pattern k as $\sum_{a\in\mathcal{A}} \left(W_{ak} \cdot \max\left\{ \left(D_{ak} - \sum_{s\in\mathcal{S}} r_{ap(c,s,k)} \right), 0 \right\} \right)$. Then, we determine z_{1k}^* as the minimum of the results and remove all $c \in \mathcal{C}_k$ not leading to z_{1k}^*. Thus, in each stage $k \in \mathcal{K}$ we obtain the set of vectors that can be chosen for column k in demand decompositions $c \in \mathcal{C}$. In stage

$K+1$, $\mathcal{C}_k$ has been determined for all $k \in \mathcal{K}$ and we can create the set of demand decompositions $\mathcal{C}$. That is, one demand decomposition $c \in \mathcal{C}$ is created by selecting one $c \in \mathcal{C}_k$ for each $k \in \mathcal{K}$.

5.4 Computing lower bounds for demand decompositions

The value of the lower bound LB_c for a demand decomposition $c \in \mathcal{C}$ is derived from the minimum number of gates needed to handle all aircraft contained in demand decomposition c and the required equipment for these gates. Since each demand decomposition $c \in \mathcal{C}$ defines how many aircraft of which class are assigned to which area $s \in \mathcal{S}$ and the gates of the areas can be planned independently, we first compute area-specific partial bounds LB_{cs} and then compute LB_c as $LB_c = \sum_{s \in \mathcal{S}} LB_{cs}$.

This division into area-specific bounds LB_{cs} has two advantages: First, calculating LB_c as the sum of the area-specific bounds LB_{cs} leads to a tighter bound on the value of Objective Function (1b), because the assignment of aircraft to areas is incorporated in the values of LB_{cs}. Second, different demand decompositions $c_1, c_2 \in \mathcal{C}$ are often equivalent with respect to individual areas $s \in \mathcal{S}$, i.e., $\mathcal{P}_{c_1 s} = \mathcal{P}_{c_2 s}$. From this follows $LB_{c_1 s} = LB_{c_2 s}$. Thus, once the subproblem of demand decomposition c_1 is solved to optimality for area s, not only can $LB_{c_1 s}$ be updated, but also $LB_{c_2 s}$ and hence, LB_{c_2}. This potentially allows eliminating c_2 if $LB_{c_2} > UB$ after the update.

When computing LB_{cs} for a pattern configuration $c \in \mathcal{C}$ and an area $s \in \mathcal{S}$, we do not specify which of the gates from $\mathcal{G}_s$ are used but we only consider the minimum number of gates required per aircraft class. However, we take downward compatibility of gates as well as MARS mode into account. We provide our algorithm to compute LB_{cs} in Appendix F.

5.5 Acceleration techniques

We apply a number of problem-specific acceleration techniques to improve the performance of our approach.

Solution pool Subproblems for a given area $s \in \mathcal{S}$ are often identical for different demand decompositions, i.e., the same number of aircraft per class are assigned to a given area in different demand decompositions. Therefore, we make use of a solution pool, in which we store the solutions to all subproblems already solved and which we inspect each time before solving another subproblem.

Solving relaxed subproblems Each time we search for the optimal solution of a subproblem, we iteratively solve a relaxation of the problem and try to show that the solution satisfies all constraints that were relaxed. If the latter fails, we add the violated constraints to the relaxation and re-solve it in the next iteration. Otherwise, the solution is feasible for the original subproblem. Similar approaches exist in literature on the vehicle routing problem, where subtour elimination constraints are relaxed and added to the problem only when they are violated by the solution (see, e.g., Laporte, Desrochers, and Nobert 1984). The solutions to the relaxed problems are also added to the solution pool and reused if possible.

Consider the subproblem for a demand decomposition $c \in \mathcal{C}$ and an area $s \in \mathcal{S}$. In the relaxed subproblem, we first consider only one demand pattern $\hat{k}$, and all constraints of demand patterns $\mathcal{K} \setminus \left\{\hat{k}\right\}$ are removed. Once the optimal solution for the relaxed subproblem is found, we determine for each of the remaining demand patterns $k \in \mathcal{K} \setminus \left\{\hat{k}\right\}$ separately whether the solution is feasible or not, i.e., whether all aircraft contained in parking pattern $p\left(c, s, k\right)$ can be parked at contact gates in area s given the positions of and equipment installed at the gates in the solution. If we fail to show that the solution is feasible for one demand pattern $\tilde{k}$, the constraints associated with demand pattern $\tilde{k}$ are added to the relaxed problem and the next iteration is started by solving the relaxed problem again.

The behavior of the algorithm can be influenced by the sequence in which demand patterns are considered. We first sort the parking patterns $\mathcal{P}_{cs}$ by decreasing values of $r_{Ap(c,s,k)}$, i.e., by decreasing number of aircraft of the largest class. When two parking patterns have equal values of $r_{Ap(c,s,k)}$, we sort them by decreasing number of aircraft of the second largest aircraft class $A-1$, etc. The demand pattern we consider first in the subproblem is determined by the value of k of the first parking pattern in the resulting list, i.e., we select the demand pattern where the number of aircraft belonging to the largest class that occurs in $\mathcal{P}_{cs}$ is the highest. This ensures that already in the first iteration enough gates are equipped for the aircraft of that class.

Definition 4 (**Bottom-up and top-down strategies**)**.** We introduce two alternative strategies defining the sequence in which the remaining demand patterns are checked for feasibility.

- *Top-down*: We consider the demand patterns in the same order in which their associated parking patterns $p\left(c, s, k\right)$ appear in the sorted list.
- *Bottom-up*: We check the demand patterns in the opposite order. That is, we begin with the demand pattern whose associated parking pattern $p\left(c, s, k\right)$ accommodates the smallest number of large aircraft. Typically, this parking pattern has the highest total number of aircraft, which is the motivation for first examining the demand pattern associated with this parking pattern.

Example 3. Consider a situation where $K = 3$, $A = 2$, $p\left(c, s, 1\right) = (6, 0)$, $p\left(c, s, 2\right) = (4, 1)$, and $p\left(c, s, 3\right) = (2, 2)$ for a given demand decomposition $c \in \mathcal{C}$ and area $s \in \mathcal{S}$. We select demand pattern 3 with parking pattern $(2, 2)$ in the first iteration, which ensures that two gates for class 2 aircraft are built in the solution of the relaxed subproblem in the first iteration. The solution may also be feasible for the remainig demand patterns 1 and 2, as gates are downward compatible and the MARS mode can be used. To verify that the solution is indeed feasible for demand patterns 1 and 2 without violating any Safety constraints (1j), we need to perform a feasibility check for both demand patterns individually. If we apply the bottom-up strategy, we first check demand pattern 1 with parking pattern $(6, 0)$. In contrast, following the top-down strategy we begin with demand pattern 2 associated with parking pattern $(4, 1)$.

Checking for feasibility To determine if a solution of the relaxed subproblem is feasible for a demand pattern not considered in the relaxation, we use an adapted version of the network approach presented in Section 5.3. Let $\hat{v}_{ga}$ be the value of the variable v_{ga} in the current solution of the relaxed problem, and let k be the demand pattern for which the solution is checked. Before we create paths through the network, we delete all nodes representing combinations of lead-in lines and aircraft classes for which no feasible gate $g \in \mathcal{G}_s : F_{lg} = 1, \hat{v}_{ga} = 1$ exists. Again, we use an iterative process in which paths are constructed, checked for reasonableness, and pruned if possible. Beyond the pruning rules defined in Section 5.3, a path is truncated at node i if it meets any of the following conditions.

Lemma 3 (**Paths and parking patterns**). *A path π can be pruned if $\sigma_{a\pi}^{(i)} > r_{ap(c,s,k)}$ $\exists a \in \mathcal{A}$.*

If the number of class $a \in \mathcal{A}$ aircraft parked in path π leading from the source node to node i $\sigma_{a\pi}^{(i)}$ is greater than the number of class a aircraft contained in parking pattern $p\,(c, s, k)$, $\sigma_{a'\pi}^{(\text{sink})}$ must be smaller than $r_{a'p(c,s,k)}$ for another aircraft class $a' \in \mathcal{A} \setminus \{a\}$, because parking pattern $p\,(c, s, k)$ is efficient according to Definition 2. Hence, path π is not suitable to show that the solution is feasible for demand pattern k. □

In addition, we prune a path if it violates Constraints (E.1d), (E.1f), or (E.1h). That is, (i) each lead-in line that is used repeatedly for different demand patterns has to be assigned to the same gate for all demand patterns, (ii) aircraft can only be parked at lead-in lines which are assigned to gates equipped to handle aircraft of the respective class, and (iii) a gate can simultaneously accommodate at most one aircraft belonging to $\mathcal{A}^{\text{large}}$ or two aircraft belonging to $\mathcal{A}^{\text{small}}$. The heuristic procedure we employ to check compliance of a path with Constraints (E.1d), (E.1f), and (E.1h) is provided in Appendix H.

Monitoring the upper bound Each time we have solved a (relaxed) subproblem for a demand decomposition $c \in \mathcal{C}$, we check whether z_{2c}^* can still be smaller than the current value of UB given the solution(s) found so far. If not, the solution process for demand decomposition c can be stopped, and c can be removed from $\mathcal{C}$. As we have shown in Section 5.4, it may still be possible to update $LB_{\tilde{c}}$ for other demand decompositions $\tilde{c} \in \mathcal{C} \setminus \{c\}$ based on the solutions determined until abortion, which may also lead to their removal from $\mathcal{C}$.

Consistency of demand decompositions In the bounding algorithm, the set of subproblems to be solved next is found by selecting the demand decomposition from $\mathcal{C}$ that has the smallest LB_c value, ensuring that in each iteration the demand decomposition with the best potential to lead to a new best solution is explored. However, due to the large number of demand decompositions, we often observe several with the same LB_c values. We then sort the affected demand decompositions according to what we call the *consistency criterion*.

Definition 5 (**Consistency**). Let ζ_{cak} denote the number of aircraft belonging to classes $\{a, \ldots, A\}$ that are parked at all areas for demand pattern $k \in \mathcal{K}$ when aircraft are assigned to areas according

to demand decomposition $c \in \mathcal{C}$, i.e., $\zeta_{cak} = \sum_{s \in \mathcal{S}} \sum_{a' \in \{a,\dots,A\}} r_{a'p(c,s,k)}$. We call demand decomposition c *consistent* if there is no area $s \in \mathcal{S}$ for which $\sum_{a' \in \{a,\dots,A\}} r_{a'p(c,s,k_1)} < \sum_{a' \in \{a,\dots,A\}} r_{a'p(c,s,k_2)}$ holds for any aircraft class $a \in \mathcal{A}^{\text{large}}$ and two demand patterns $k_1, k_2 \in \mathcal{K} : k_1 \neq k_2$, given that $\zeta_{cak_1} \geq \zeta_{cak_2}$. A demand decomposition thus is consistent if large aircraft are repeatedly assigned to the same areas for different demand patterns, rather than being assigned to different areas. According to the consistency criterion, preference should be given to consistent demand decompositions over non-consistent demand decompositions with same LB_c value.

Example 4. Consider a greenfield situation with $A = 2$, where $a = 1$ and $a = 2$ represent classes of small and large aircraft, respectively. Assume $K = 2$, with the demand patterns given as $D_{ak} = (10, 1)$ and $(8, 2)$, and assume there are $S = 2$ areas. Furthermore, consider two distinct demand decompositions $c_1 = \begin{pmatrix} (3,1) & (3,1) \\ (6,0) & (4,1) \end{pmatrix}$ and $c_2 = \begin{pmatrix} (3,1) & (5,0) \\ (6,0) & (2,2) \end{pmatrix}$. According to Algorithm F.1, $LB_{c_1} = LB_{c_2}$. However, c_1 is a consistent demand decomposition, whereas c_2 is not, and hence, demand decomposition c_1 is considered first in the bounding algorithm following the consistency criterion.

We provide the algorithm we utilize to determine whether a demand decomposition is consistent or not in Appendix G.

6 Computational Experiments

In the following, we examine the performance of our approach and analyze the solutions of individual instances. In Section 6.1, we describe the instances we used in our experiments. Section 6.2 compares the performance of our approach to CPLEX solving Model (1a)-(1p) directly, and in Section 6.3, we investigate the impact of the acceleration techniques introduced in Section 5.5. Finally, we discuss the optimal solutions for individual instances in detail in Section 6.4. All experiments were performed on a computer equipped with an Intel Xeon E3-1225 v3 @3.20GHz processor and 12GB of working memory. The implementation was done in Java and CPLEX version 20.1.0 was used.

6.1 Instances

All instances are based on data from Munich Airport Terminal 1. With 47.9 million passengers in 2019, Munich Airport is the second largest airport in Germany (Munich Airport 2020). Before the crisis caused by the Corona virus in 2020, 101 airlines were active at Munich Airport, connecting the airport to 254 destinations in 75 countries. Transfer passengers accounted for 38% of total passenger traffic in 2019, so Munich Airport is considered a hub airport. The airport has two terminals, Terminal 1 being the smaller of the two providing about a third of the airport's total capacity. The terminal is currently being renovated and extended by a new section. Figure 4 shows the apron layout for Terminal 1, with the new extension highlighted in red.

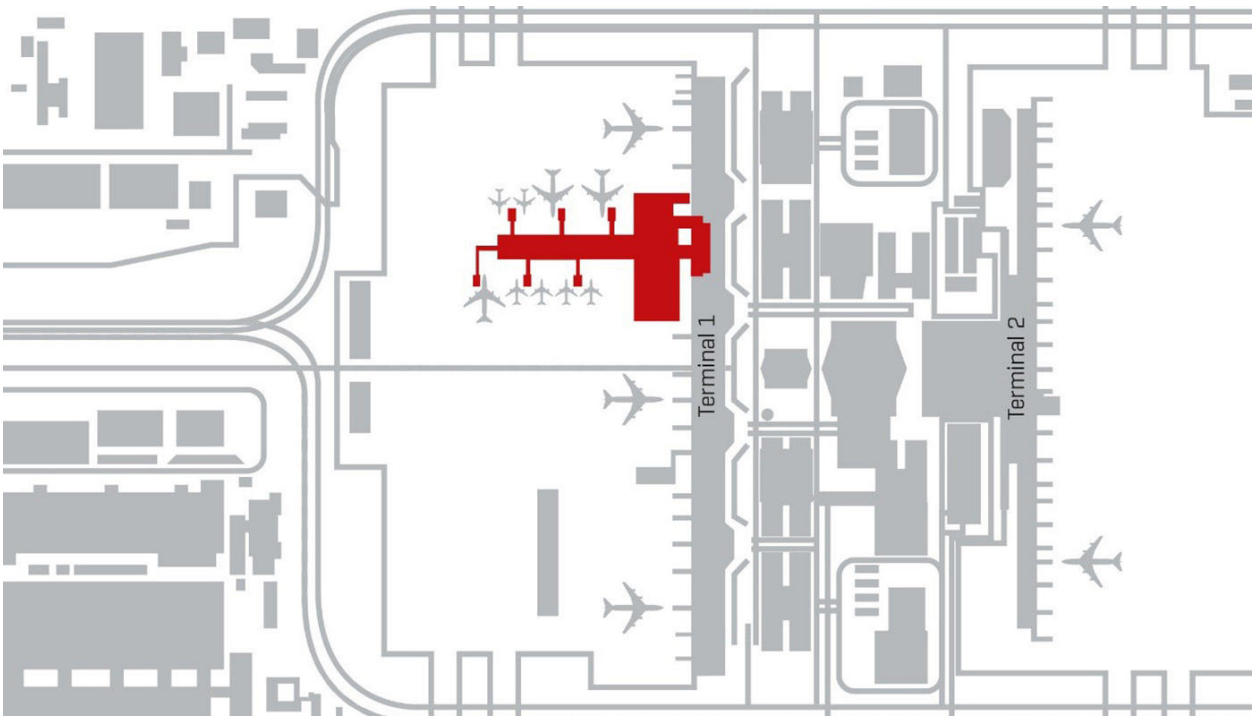


Figure 4: Terminal layout of Munich Airport and extension of Terminal 1 (Source: Munich Airport)

As part of the expansion and reconstruction activities, the positions of gates and lead-in lines have to be determined for the new section of the terminal, and changes may also be made in existing sections of Terminal 1. We consider three planning scenarios, resulting in three separate sets of instances:

- *Greenfield*: Existing gates and lead-in lines are not considered.
- *Soft brownfield*: Existing gates and lead-in lines south of the extension are considered. Existing gates cannot be changed, but additional lead-in lines and gates can be added to the layout of these sections.
- *True brownfield*: In sections south of the extension, the gate layout remains unchanged, i.e., only existing gates and lead-in lines can be used.

As demand patterns are defined for the terminal building as a whole, we solve the AGLP for the entire terminal, including the sections south of the extension regardless of the planning scenario. To determine the sets of lead-in lines $\mathcal{L}$ and gates $\mathcal{G}$ for all instances, as well as the values of the parameters C_{al}, E_{lmab}, F_{lg}, and H_{ga}, which determine the relationships among the gates and lead-in lines, we applied the procedure described in Section 3 and validated the results with planning experts from Munich Airport. We employed satellite images as well as floor plan drawings provided by Munich Airport to determine the relevant coordinates of the terminal building and surrounding taxiways. In all brownfield instances, 11 gates and 18 lead-in lines that already exist south of the extension were added to sets $\mathcal{G}$ and $\mathcal{L}$, respectively, based on their coordinates. Furthermore,

all other gates and lead-in lines south of the new extension were removed from sets $\mathcal{G}$ and $\mathcal{L}$ in the true brownfield instances. To investigate the computational performance of our approach, we created instances of different complexity for each planning scenario by varying the distance between adjacent starting points Δ as well as the rotation angle κ between the lead-in lines that share the same starting point, where $\Delta \in \{10\text{m}, 7.5\text{m}, 5\text{m}, 2.5\text{m}\}$ and $\kappa \in \{90°, 45°, 30°, 22.5°, 15°\}$. We ensured that 90° is an integer multiple of each value of κ, since we concluded from the exchange with Munich Airport that lead-in lines perpendicular to the terminal facade are the easiest to operate in practice. However, the use of non-perpendicular lead-in lines may still considerably improve z_1^*, as they may allow accommodation of additional aircraft in corners or at the ends of the terminal building. We visualize sets $\mathcal{G}$ and $\mathcal{L}$ for $\Delta = 5\text{m}$ and $\kappa = 22.5°$ in Appendix I for each planning scenario. In summary, we created instances for three scenarios, four values of Δ, and five values of κ, resulting in $3 \cdot 4 \cdot 5 = 60$ instances in total.

The smaller the values of Δ and κ, the more flexibly aircraft can be parked on the apron, and the better solutions we expect for Objective Function (1a). Better solutions for Objective Function (1a) might suggest worse solutions for Objective Function (1b), as accomodating more aircraft requires more gates. However, the smaller the values of Δ and κ, the more precisely gates can be placed, potentially reducing the number of gates needed to serve a given number of aircraft simultaneously.

In all instances, the set of aircraft classes $\mathcal{A}$ corresponds to the classification introduced in Table 1, and hence $\mathcal{A}^{\text{small}} = \{1, 2, 3\}$ and $\mathcal{A}^{\text{large}} = \{4, 5, 6\}$. Based on the interviews with experts from Munich Airport, we set C_{al} to 1 for all $a \in \mathcal{A}$ and $l \in \mathcal{L}$. Furthermore, F_{lg} equals 1 if the path between gate g and lead-in line l is unobstructed, and gate g is located at a maximum distance of 40.5 meters to the left of lead-in line l from the point of view of the parked aircraft. Finally, the values of E_{lmab} were determined as stated in Section 3.

Demand patterns $k \in \mathcal{K}$ and associated traffic volumes D_{ak} were obtained from the flightplan forecast of Munich Airport for the year 2030. We identified five demand patterns for Terminal 1, which are given in Table 2.

Table 2: Values of D_{ak} derived from the flightplan forecast for the year 2030

	k				
a	1	2	3	4	5
3	39	31	29	25	23
5	0	2	3	5	6
6	0	0	0	1	1

Aircraft of classes $\{1, 2, 4\}$ are rarely seen at Munich Airport and are therefore not included in Table 2.

Finally, the weights W_{ak} for Objective Function (1a) are calculated as follows: We assume that all demand patterns given in Table 2 have equal likelihood, and hence the values of W_{ak} are

independent of k. In contrast, larger aircraft are associated with larger weights. Thus, for each $a \in \mathcal{A}$ and $k \in \mathcal{K}$, we set W_{ak} to $0.2 \cdot a^2$.

6.2 Computational performance compared to CPLEX

To evaluate the performance of our approach, we applied it to all soft brownfield instances. We selected the soft brownfield scenario, because the sets $\mathcal{G}$ and $\mathcal{L}$ are the largest for given values of Δ and κ compared to the greenfield or true brownfield counterparts. We used all acceleration techniques introduced in Section 5.5 and applied the top-down strategy. As a benchmark, we solved Model (1a)-(1p) using CPLEX, with a time limit of 24 hours per objective function. In Table 3, we provide for each instance the resulting objective function values, optimality gaps, and runtimes for both CPLEX and our decomposition approach (DA). Furthermore, we provide relevant performance indicators for our approach, where C_{con} denotes the number of consistent demand decompositions of an instance; C_{exa} and SP_{exa} represent the number of demand decompositions and subproblems, respectively, examined in the bounding algorithm; and $\bar{K}_{SP}$ gives the average number of demand patterns that had to be included in the relaxed subproblems until either feasibility of the solution for the remaining demand patterns was proven or the optimization process was aborted.

Our results demonstrate that Model (1a)-(1p) cannot be solved to optimality by CPLEX for small values of Δ and κ within a reasonable amount of time. For five instances, z_1^* was not found within 24 hours. Furthermore, for two instances z_1^* was found but not proven to be optimal. Only for 11 of the 15 instances, for which z_1^* could be determined, could z_2^* also be found within the time limit. Again, for two of these 11 instances, z_2^* was found but not proven to be optimal. In summary, CPLEX could solve only 11 of 20 instances to optimality, thereby proving optimality for only nine instances. In particular, no instance with $\kappa = 15°$ could be solved to optimality, and for $\Delta \in \{5\text{m}, 2.5\text{m}\}$ only the instances with $\kappa = 90°$ could be solved to optimality with no remaining optimality gap.

In contrast, with the decomposition approach we can solve 18 of 20 instances to optimality within 24 hours, and we find a feasible solution for one additional instance with an optimality gap of 2.63%. Our approach finds z_1^* for all instances within less than three hours. Averaged over all instances where z_1^* was found and proven to be optimal by CPLEX within the time limit, our approach reduces the runtime per instance for Objective Function (1a) by 84.14%. As for Objective Function (1b), the decomposition approach reduces the average computation time per instance by 84.06% compared to CPLEX. Here, we only consider the instances for which both CPLEX and our algorithm found z_1^* and z_2^* and proved optimality within the time limit. Averaged over the same instances, our approach reduces the total computation time per instance, i.e., the time needed to solve both objective functions to optimality, by 82.70%.

The number of demand decompositions found varies between 24 and 2,750. In comparison, the number of demand decompositions examined in the bounding algorithm is significantly smaller. For ten instances, only two demand decompositions need to be examined in the bounding algorithm until it is proven that the optimal solution has been found. The maximum number of demand

Table 3: Computational results for soft brownfield instances

Instance					Objectives				Runtimes [min]				Info on DA				
					Objective (1a)		Objective (1b)		Objective (1a)		Objective (1b)						
Nr.	Δ [m]	κ [°]	G	L	CPLEX*	DA	CPLEX*	DA*	CPLEX	DA	CPLEX	DA	C	C_{con}	C_{exa}	SP_{exa}	$\bar{K}_{SP}$
1	10	90	136	119	61	61	34	34	0	0	0	0	2625	69	4	6	1.86
2		45		235	56	56	35	35	0	0	1	0	798	36	4	6	1.86
3		30		297	41	41	43 [12]	43	0	0	> 1440	0	630	20	2	4	2
4		22.5		418	36	36	37	37	7	0	36	1	416	48	4	6	1.5
5		15		613	31	31	–	39	139	2	> 1440	2	396	16	2	5	1.4
6	7.5	90	178	154	77	77	34	34	0	0	3	1	840	80	6	12	1.5
7		45		310	67	67	34	34	3	0	65	0	2750	112	2	7	1.14
8		30		392	50	50	36	36	3	0	14	1	80	18	4	5	1.57
9		22.5		556	40	40	40	40	504	1	12	1	198	24	2	4	1
10		15		815	41 [43]	41	–	36	> 1440	5	> 1440	7	384	48	2	4	1.25
11	5.0	90	260	222	56	56	35	35	0	0	2	0	315	18	2	4	1.5
12		45		454	50	50	–	39	82	1	> 1440	1	630	20	2	3	1.5
13		30		578	38	38	37 [1]	37	787	2	> 1440	4	980	36	8	8	1.2
14		22.5		823	29 [50]	29	–	37	> 1440	5	> 1440	189	1400	48	3	5	1.5
15		15		1210	61 [91]	29	–	38	> 1440	17	> 1440	334	770	30	2	4	1.75
16	2.5	90	507	428	50	50	38	38	2	1	138	2	216	42	4	7	1.43
17		45		891	47 [77]	38	–	41	> 1440	5	> 1440	511	24	6	2	4	2
18		30		1139	246 [99]	32	14	40	> 1440	14	15	313	504	63	6	8	1.38
19		22.5		1626	300 [100]	25	14	39 [3]	> 1440	25	32	> 1440	96	12	≥ 2	≥ 5	1.2
20		15		2400	360 [100]	20	14	≥ 38**	> 1440	172	11	> 1440	132	9	≥ 1	≥ 3	–
av.			270	684	87	43	–	37.5	581	12	652	212	709	37.75	3.2	5.5	1.48

* Values in parentheses indicate optimality gaps in percent

** Lower bound of first demand decomposition examined in the bounding algorithm

decompositions examined across all instances is eight, highlighting the efficiency of the bounding algorithm. The number of subproblems examined using CPLEX is also low, ranging between three and 12. The number of independent areas found is four for instances 6 and 7, and three for all other instances. However, we have no indication that the number of independent areas has an impact on computational performance.

The objective function values meet our expectations. For a given value of κ, z_1^* improves when Δ is reduced from 10m to 5m or from 5m to 2.5m. This is inevitable, because instances with $\Delta = 2.5$m include all lead-in lines and gates of instances with $\Delta = 5$m, and instances with $\Delta = 5$m include all lead-in lines and gates of instances with $\Delta = 10$m. Instances with $\Delta = 7.5$m do not fit into this scheme, which is why the z_1^* values of these instances can be and in fact are worse than those of the instances with $\Delta = 10$m. Similarly, smaller values of κ generally yield better solutions with respect to Objective Function (1a) for a given value of Δ. Regarding Objective Function (1b), we do not see any systematic impact of the values of Δ and κ on z_2^*. Nevertheless, the range of solutions with values between 34 and 43 indicates that the choice of Δ and κ exerts considerable influence on z_2^*.

6.3 Effectiveness of acceleration techniques

We introduced multiple acceleration techniques in Section 5.5. In the following, we examine the impact of each of these acceleration techniques on the performance of our approach. For this purpose, we solved all soft brownfield instances from Table 3 multiple times, each time removing or exchanging exactly one of the acceleration techniques.

For each configuration of our approach with respect to acceleration techniques, Table 4 depicts the number of instances (out of 20) for which a feasible solution was found within the time limit

(n_{fea}) and the number of instances that were solved to optimality within the time limit (n_{opt}). Furthermore, averaged over all instances counted in n_{opt}, Table 4 provides the percental runtime increase per instance (Δ^{+}_{time}) compared to the runtimes obtained in Table 3, where all acceleration techniques and the top-down strategy were used. As the acceleration techniques have no impact on the part of our approach dedicated to finding the best solution for Objective Function (1a), we concentrate on the computation times for Objective Function (1b). If not stated differently, the top-down strategy was applied to sort demand patterns.

Table 4: Performance results when individual acceleration techniques are deactivated

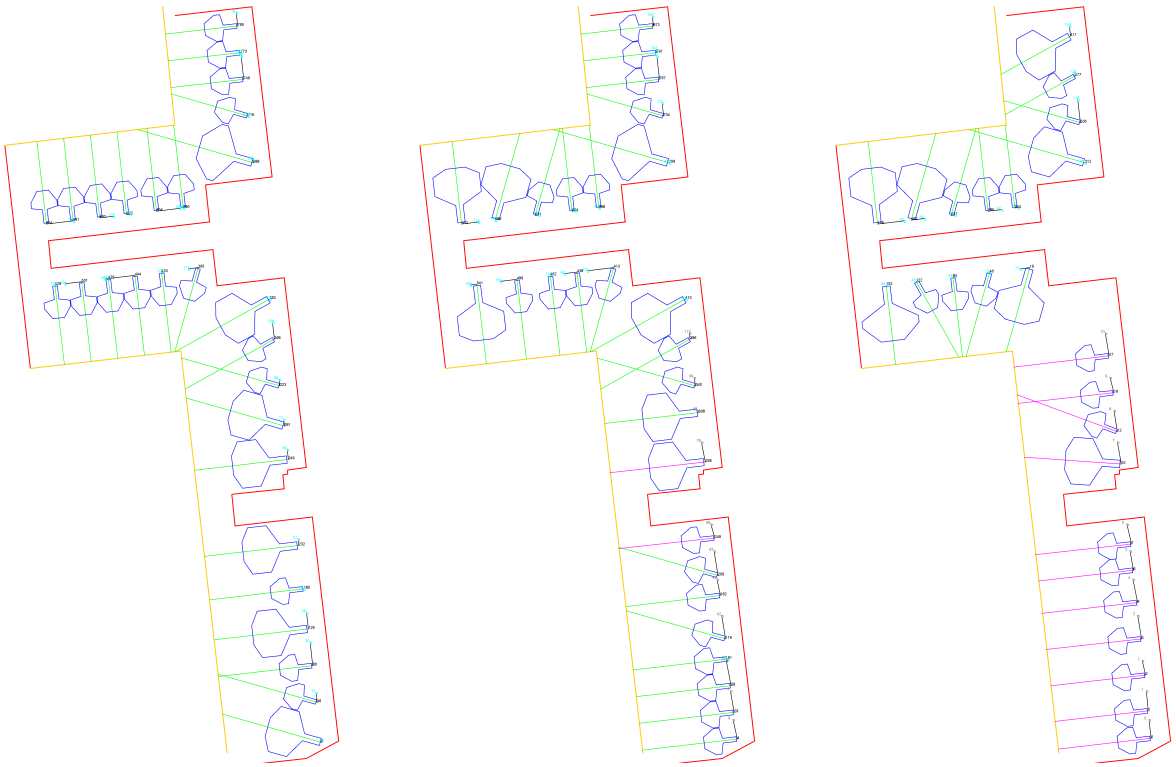
Description	n_{fea}	n_{opt}	Δ^{+}_{time} [%]
No solution pool	19	18	9.81
No relaxation of subproblems	15	14	859.21
Bottom-up strategy	19	18	58.17
No monitoring of UB	19	17	197.61
Consistency not considered	18	18	22.21

Removing any acceleration technique leads to a degradation of the performance of our algorithm. Most importantly, the average runtime per instance increases by more than 800% when subproblems are solved directly without relaxations, and fewer instances can be solved in that case. When the optimization process of the subproblems associated with demand decomposition $c \in \mathcal{C}$ is not terminated as soon as it becomes apparent that z^{*}_{2c} cannot be smaller than the current upper bound UB, we observe that the average runtime per instance increases by almost 200%. The remaining acceleration techniques also improve the performance of our algorithm, with average runtime reductions per instance and acceleration technique ranging between 10% and 60%.

6.4 Analysis of optimal layouts

We now address the resulting gate layout for Munich Airport Terminal 1. We have solved the greenfield, soft brownfield, and true brownfield instances for $\Delta = 5\text{m}$ and $\kappa = 22.5°$, and we compare the optimal layouts in the following. For each planning scenario, Figure 5 shows which gates and lead-in lines are used in the optimal solution, which lead-in lines are assigned to which gates, and the safety envelope of the aircraft parked at each lead-in line, indicating the class of each parked aircraft. The terminal facade and other airport structures are shown by red lines, and taxiways are illustrated by yellow lines. Existing gates are represented by gray circles in front of the terminal facade, whereas new gates are colored cyan. Similarly, existing lead-in lines are depicted in purple, while new lead-in lines are colored green. Black lines connect lead-in lines to the gates they are assigned to. Finally, aircraft safety envelopes are displayed as blue polygons. In order not to overload the figures, they show only lead-in lines and aircraft safety envelopes for demand pattern 5. Which lead-in lines are used for the remaining demand patterns is illustrated in Appendix J. Furthermore, for all planning scenarios, Table 5 provides the optimal values of Objective Functions

(1a) and (1b), the optimal values of Objective Function (1b) when already existing gates are not considered, and the number of gates used to park aircraft across all demand patterns.



(a) Greenfield (b) Soft brownfield (c) True brownfield

Figure 5: Optimal solutions for different planning scenarios, demand pattern 5

Table 5: Performance data of optimal layouts for different planning scenarios

Planning scenario	z_1^*	z_2^*	z_2^* without already existing gates	Number of gates used
Greenfield	28.8	36	36	27
Soft brownfield	28.8	37	23	28
True brownfield	55.8	35	21	25

Compared to the true brownfield case, sets $\mathcal{G}$ and $\mathcal{L}$ contain significantly more gates and lead-in lines in the greenfield and soft brownfield scenarios, which increases planning flexibility. Consequently, the number of aircraft that cannot be parked at contact gates weighted by aircraft class and demand pattern (z_1^*) is substantially lower in the greenfield and soft brownfield scenarios than in the true brownfield scenario. However, there is no difference in that value between the greenfield and soft brownfield scenarios, suggesting that the existing gates and lead-in lines that distinguish the two scenarios from each other do not lead to an increase in parking capacity in our case.

The construction effort (z_2^*) is similar in all scenarios when we assume that all gates need to be newly built, regardless of whether they already exist or not. In contrast, when we take into account

that already existing gates do not need to be newly built, the remaining construction effort is the highest in the greenfield scenario.

Our results indicate that the substantial decrease in the number of aircraft that cannot be parked at contact gates weighted by aircraft class and demand pattern in the soft brownfield scenario compared to the true brownfield scenario does not lead to an equally significant increase in construction effort for gates. Instead, the increase in construction effort is comparatively small, also when taking the already existing gates into account. Thus, by allowing changes to the existing layout in the southern part of the terminal, a significant improvement in parking capacity can be achieved, while the actual changes to the existing gate layout remain relatively small.

While the number of gates used across all demand patterns is lower in the greenfield scenario than in the soft brownfield case, the actual construction effort is higher in the greenfield scenario. This is consistent with our previous finding that in the soft brownfield scenario, although individual changes are made to the existing layout, the existing gates are utilized well. In contrast, few existing lead-in lines are used in the soft brownfield scenario, because we do not penalize the use of new lead-in lines in our model.

Figure 5 shows that mainly orthogonal lead-in lines are used in the optimal layouts regardless of the scenario, without explicitely incentivizing the use of orthogonal lead-in lines in our model. Non-orthogonal lead-in lines are only used in three cases: (i) At corners of the terminal building where no orthogonal lead-in lines can be used, (ii) when specific geometric situations favor the use of non-orthogonal lead-in lines, and (iii) when there is slack space that does not suffice to park another aircraft, even if all lead-in lines were orthogonal. However, in situations (i) and (ii), the use of non-orthogonal lead-in lines enables parking more aircraft simultaneously, so non-orthogonal lead-in lines should always be considered in the planning. As an example for (i), consider the junction of the new extension to the existing terminal building, where large aircraft are always parked at non-orthogonal lead-in lines in the corners to fill the corner space as efficiently as possible. Regarding (ii), two class 3 aircraft are parked at non-orthogonal lead-in lines at the southern facade of the new extension building in the true brownfield scenario. Given the parking positions of the larger aircraft to the left and right of this spot, only two class 3 aircraft could be parked in this section instead of three if only perpendicular lead-in lines were used. An example for (iii) can be found at the northern end of the terminal in the true brownfield scenario, where the two last aircraft could also be parked at orthogonal lead-in lines.

Finally, we find that the optimal layouts in Figure 5 and Appendix J are quite distinct from each other: Depending on the scenario, large aircraft are parked in different areas of the terminal and few lead-in lines can be found, which are used in all scenarios for a given demand pattern. Nevertheless, the solutions for the greenfield and soft brownfield scenarios are very similar in terms of our objective functions, which leads us to suspect that several solutions exist per scenario that are equivalent with respect to our objective functions. In that case, it might be useful to identify all these equivalent solutions in order to be able to provide the planner with several alternatives.

7 Conclusion

We introduced the Airport Gate Layout Problem (AGLP) and presented a mixed-integer model formulation for the problem that can be applied to both greenfield and brownfield instances. To solve the problem efficiently, we presented a decomposition framework, which features a custom bounding algorithm as well as problem-specific acceleration techniques. In our computational experiments, we demonstrated the superior performance of our approach compared to CPLEX and investigated the impact of each acceleration technique on runtimes. In addition, we analyzed and compared particular layouts for Munich Airport Terminal 1 in different planning scenarios.

Solving the AGLP provides valuable decision support to planning experts. First, the optimal layout shows how to make the best use of the available space, especially for complex terminal geometries. Second, the optimal solution can be used as a benchmark for layout alternatives created by planning experts. Finally, when a new terminal is to be built and the layout of the building has not yet been determined, solving the AGLP can be used to evaluate different terminal layouts in terms of their potential to handle as many of the expected aircraft as possible at contact gates. Our decomposition approach considerably reduces the time needed to solve the AGLP for a given instance, which allows us to solve instances that are intractable for CPLEX. Thus, we can solve the AGLP with increased planning granularity for a particular terminal building, resulting in a better solution.

To conclude, we outline some possibilities for future research on the AGLP. First, we have not considered the affiliation of aircraft to airlines so far. Airlines that operate a base at an airport tend to have their own terminal or terminal areas where all flights of these airlines are handled. Of course, these terminal areas do not necessarily correspond to the terminal areas we define in our solution approach. However, different airlines are associated with different fleet compositions. Therefore, if the affiliation of aircraft to airlines is to be taken into account, the parking positions in each terminal area must reflect the fleet mix of the particular airline. Second, to reduce operational complexity on the apron, the AGLP may be extended by another objective function to minimize the number of lead-in lines, especially those that are non-orthogonal to the terminal facade, used across all demand patterns. Third, it might be useful for decision makers to receive not only one solution, but several alternative layouts that are equivalent with respect to the given objectives and constraints, provided there is more than one optimal solution. While different layouts may be equally optimal from the perspective of our model, one alternative layout might be easier to realize than others, especially in brownfield situations. Finally, future work should be concerned with further shortening the runtimes of the AGLP. Ideally, planners will be equipped with an interactive optimization tool in which the optimal layout can be manually adjusted and then re-optimized. In particular, approaches that reduce the runtime of the relaxed subproblems could be promising.

Acknowledgments

We would like to thank Munich Airport, especially Carolin Aust and Leander von Preysing, for the constructive exchange throughout the research project. We would also like to thank Alexander Druska, who initiated the research on the topic with his master's thesis, and Dr. Stephen Starck for his support in refining the language of the paper.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Declaration of Interest

None.

References

Airbus (2022) *Airport operations - autocad 3d view aircraft drawings*. URL https://web.archive.org/web/20211027010324/https:/www.airbus.com/aircraft/support-services/airport-operations-and-technical-data/autocad-3-view-aircraft-drawings.html.

Airport Improvement Magazine (2010) *"like new" boarding bridges return to service at daytona beach*. URL https://airportimprovement.com/article/new-boarding-bridges-return-service-daytona-beach.

Anjos MF, Vieira MV (2017) *Mathematical optimization approaches for facility layout problems: The state-of-the-art and future research directions*. *European Journal of Operational Research* 261(1):1–16.

Bandara SJ, Wirasinghe SC (1989) *Airport gate position estimation under uncertainty*. *Transportation Research Record Journal of the Transportation Research Board* 1199:41–48.

Briskorn D, Dienstknecht M (2019) *Mixed-integer programming models for tower crane selection and positioning with respect to mutual interference*. *European Journal of Operational Research* 273(1):160–174.

Caves R (1994) *A search for more airport apron capacity*. *Journal of Air Transport Management* 1(2):109–120.

Cheng CH, Ho SC, Kwan CL (2012) *The use of meta-heuristics for airport gate assignment*. *Expert Systems with Applications* 39(16):12430–12437.

Daş GS, Gzara F, Stützle T (2020) *A review on airport gate assignment problems: Single versus multi objective approaches*. *Omega* 92:102–146.

Dorndorf U, Drexl A, Nikulin Y, Pesch E (2007) *Flight gate scheduling: State-of-the-art and recent developments*. *Omega* 35(3):326–334.

Dorndorf U, Jaehn F, Pesch E (2008) *Modelling robust flight-gate scheduling as a clique partitioning problem*. *Transportation Science* 42(3):292–301.

Dorndorf U, Jaehn F, Pesch E (2012) *Flight gate scheduling with respect to a reference schedule*. *Annals of Operations Research* 194(1):177–187.

Dorndorf U, Jaehn F, Pesch E (2017) *Flight gate assignment and recovery strategies with stochastic arrival and departure times*. *OR Spectrum* 39(1):65–93.

Drira A, Pierreval H, Hajri-Gabouj S (2007) *Facility layout problems: A survey*. *Annual Reviews in Control* 31(2):255–267.

European Aviation Safety Agency (EASA) (2017) *Certification specifications and guidance material for aerodromes design cs-adr-dsn*. URL https://www.faa.gov/documentLibrary/media/Advisory_Circular/150-5300-13A-chg1-interactive-201907.pdf.

Federal Aviation Administration (FAA) (2012) *Ac 150/5300-13a - airport design*.

Guépet J, Acuna-Agost R, Briant O, Gayon JP (2015) *Exact and heuristic approaches to the airport stand allocation problem*. *European Journal of Operational Research* 246(2):597–608.

Hagspihl T, Kolisch R, Ruf C, Schiffels S (2022) *Dynamic gate configurations at airports: A network optimization approach*. *European Journal of Operational Research* 301(3):1133–1148.

Hassounah MI, Steuart GN (1993) *Demand for aircraft gates*. *Transportation Research Record* 1423(1423):26–33.

Huang C, Wong CK, Tam CM (2011) *Optimization of tower crane and material supply locations in a high-rise building site by mixed-integer linear programming*. *Automation in Construction* 20(5):571–580.

International Civil Aviation Organization (ICAO) (2018) *Annex 14 to the convention on international civil aviation: Aerodromes: Volume 1 aerodrome design and operations*. URL https://www.easa.europa.eu/sites/default/files/dfu/Annex%20to%20EDD%202017-021-R%20-%20CS-ADR-DSN%20Issue%204_0.pdf.

Laporte G, Desrochers M, Nobert Y (1984) *Two exact algorithms for the distance-constrained vehicle routing problem. Networks* 14(1):161–172.

Martins EQV (1984) *On a multicriteria shortest path problem. European Journal of Operational Research* 16(2):236–245.

Mirkovic B, Tosic V (2014) *Airport apron capacity: estimation, representation, and flexibility. Journal of Advanced Transportation* 48(2):97–118.

Mirković B, Tošić V (2016) *Apron capacity at hub airports-the impact of wave-system structure. Journal of Advanced Transportation* 50(7):1489–1505.

Mirković B, Tošić V (2017) *The difference between hub and non-hub airports – an airside capacity perspective. Journal of Air Transport Management* 62:121–128.

Munich Airport (2020) *Annual traffic report 2019*. URL https://www.munich-airport.com/_b/0000000000000008934656bb5e9eb677/annual-traffic-report-2019.pdf.

Narciso ME, Piera MA (2015) *Robust gate assignment procedures from an airport management perspective. Omega* (50):82–95.

National Academies of Sciences, Engineering, and Medicine (NASEM) (2010) *Airport Passenger Terminal Planning and Design: Volume 1: Guidebook* (Washington, DC: The National Academies Press).

Stephan K, Weidinger F, Boysen N (2021) *Layout design of parking lots with mathematical programming. Transportation Science* 55(4):930–945.

Steuart GN (1974) *Gate position requirements at metropolitan airports. Transportation Science* 8(2):169–189.

Travel PR News (2019) *Budapest airport installs brand new passenger boarding bridges*. URL https://travelprnews.com/budapest-airport-installs-brand-new-passenger-boarding-bridges-649599/travel-press-release/2019/05/27/.

Tung Tung C, Lin Chew K (1992) *A multicriteria pareto-optimal path algorithm. European Journal of Operational Research* 62(2):203–209.

Wirasinghe SC, Bandara SJ (1990) *Airport gate position estimation for minimum total costs—approximate closed form solution. Transportation Research Part B: Methodological* 24(4):287–297.

Appendix A Notation

Table A.1: Notation for AGLP model (1a)-(1p)

Sets, parameters	
$\mathcal{A} = \{1, \ldots, A\}$	aircraft classes
$\mathcal{L} = \{1, \ldots, L\}$	lead-in lines
$\mathcal{G} = \{1, \ldots, G\}$	gates
$\mathcal{K} = \{1, \ldots, K\}$	demand patterns
$\mathcal{A}^{\text{small}}$	subset of $\mathcal{A}$, which includes the aircraft classes from which up to two aircraft can be processed at one gate simultaneously
$\mathcal{A}^{\text{large}}$	subset of $\mathcal{A}$, which includes the aircraft classes from which only one aircraft can be processed at one gate at one point in time
C_{al}	1, if an aircraft of class $a \in \mathcal{A}$ can be parked at lead-in-line $l \in \mathcal{L}$; 0, otherwise
E_{lmab}	1, if aircraft of classes $a \in \mathcal{A}$ and $b \in \mathcal{A}$ cannot be parked simultaneously at lead-in lines $l \in \mathcal{L}$ and $m \in \mathcal{L} \setminus \{l\}$; 0, otherwise
F_{lg}	1, if lead-in line $l \in \mathcal{L}$ can be assigned to gate $g \in \mathcal{G}$; 0, otherwise
H_{ga}	1, if aircraft of class $a \in \mathcal{A}$ could be parked at gate $g \in \mathcal{G}$ in the past; 0, otherwise
D_{ak}	number of aircraft of class $a \in \mathcal{A}$ that need to be parked simultaneously for demand pattern $k \in \mathcal{K}$
W_{ak}	weighting factor for aircraft class $a \in \mathcal{A}$ and demand pattern $k \in \mathcal{K}$
Δ	distance between adjacent lead-in line starting points
κ	rotation angle between the lead-in lines that share the same starting point
Decision variables	
$y_l \in \{0, 1\}$	1, if lead-in line $l \in \mathcal{L}$ is used to park an aircraft for at least one demand pattern; 0, otherwise
$x_{glak} \in \{0, 1\}$	1, if lead-in line $l \in \mathcal{L}$ is assigned to gate $g \in \mathcal{G}$ and is used to park an aircraft of class $a \in \mathcal{A}$ for demand pattern $k \in \mathcal{K}$; 0, otherwise
$u_{gl} \in \{0, 1\}$	1, if lead-in line $l \in \mathcal{L}$ is assigned to gate $g \in \mathcal{G}$; 0, otherwise
$q_{ak} \geq 0$	number of aircraft belonging to class $a \in \mathcal{A}$ that cannot be parked at any contact gate for demand pattern $k \in \mathcal{K}$
$v_{ga} \in \{0, 1\}$	1, if aircraft of class $a \in \mathcal{A}$ can be parked at gate $g \in \mathcal{G}$; 0, otherwise

Table A.2: Notation for solution approach

Sets, parameters	
$\mathcal{S} = \{1, \ldots, S\}$	airport areas that are independent with respect to Constraints (1j)
$\mathcal{C} = \{1, \ldots, C\}$	demand decompositions
$\mathcal{C}_k$	set of demand decompositions pretending that demand pattern $k \in \mathcal{K}$ is the only demand pattern that exists
$\mathcal{L}_s$	subset of $\mathcal{L}$ containing all lead-in lines belonging to area $s \in \mathcal{S}$
$\mathcal{G}_s$	subset of $\mathcal{G}$ containing all gates belonging to area $s \in \mathcal{S}$
$\mathcal{P}_s$	set of efficient parking patterns associated with area $s \in \mathcal{S}$
$\mathcal{P}_{cs}$	set of parking patterns $p \in \mathcal{P}_s$ contained in demand decomposition $c \in \mathcal{C}$ for area $s \in \mathcal{S}$
r_{ap}	number of aircraft of class $a \in \mathcal{A}$ contained in parking pattern $p \in \mathcal{P}_s$
$p\,(c, s, k)$	parking pattern from $\mathcal{P}_s$ that is contained in demand decomposition $c \in \mathcal{C}$ for area $s \in \mathcal{S}$ and demand pattern $k \in \mathcal{K}$
π	a path through the network created to find efficient parking patterns and to check whether a solution to the relaxed subproblem is feasible for a parking pattern
z_1^*, z_2^*	optimal values of Objective Functions (1a) and (1b), respectively
z_{1k}^*	optimal value of Objective Function (1a) that can be reached for demand pattern $k \in \mathcal{K}$
z_{1c}^*, z_{2c}^*	optimal values for Objective Functions (1a) and (1b) when aircraft are assigned to areas according to demand decomposition $c \in \mathcal{C}$
z_{2cs}^*	optimal value for Objective Function (1b) in area $s \in \mathcal{S}$ when aircraft are assigned to areas according to demand decomposition $c \in \mathcal{C}$
Variables	
LB_c	lower bound for the value of Objective Function (1a) when aircraft are assigned to areas according to demand decomposition $c \in \mathcal{C}$
LB_{cs}	lower bound for the value of Objective Function (1a) in area $s \in \mathcal{S}$ when aircraft are assigned to areas according to demand decomposition $c \in \mathcal{C}$
UB	upper bound for the value of Objective Function (1a)
$\sigma_{a\pi}$	number of class $a \in \mathcal{A}$ aircraft parked in path π
ω_{csa}	minimum number of gates equipped for class $a \in \mathcal{A}$ aircraft required in area $s \in \mathcal{S}$ to park all aircraft contained in $\mathcal{P}_{cs}$
ψ_{csak}	number of class $a \in \mathcal{A}$ aircraft that can be parked at gates which are equipped for aircraft of a larger class $a' > a$ in area $s \in \mathcal{S}$ for demand pattern $k \in \mathcal{K}$ when aircraft are assigned to areas according to demand decomposition $c \in \mathcal{C}$

ζ_{cak} number of aircraft belonging to classes $\{a, \ldots, A\}$ that are parked at all areas for demand pattern $k \in \mathcal{K}$ when aircraft are assigned to areas according to demand decomposition $c \in \mathcal{C}$

Appendix B Process to compute aircraft safety envelopes

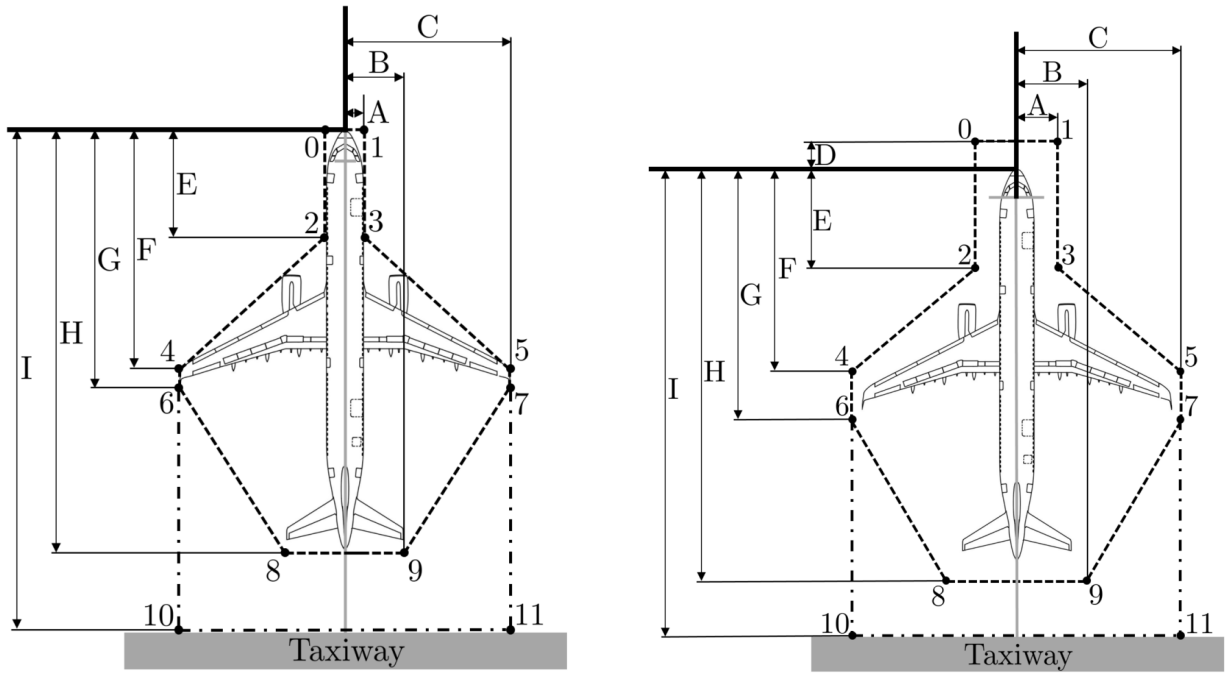
To compute the safety envelope for each aircraft class, we first determine the dimensions of a so-called *critical design aircraft*. We then compute the aircraft safety envelope based on the critical design aircraft.

Critical design aircraft The critical design aircraft represents a fictitious aircraft type, which in all respects has the maximum dimensions of all real aircraft types included in the respective ARC letter. Thus, all real aircraft belonging to a class can be handled at a parking position if the critical design aircraft corresponding to the class can be parked there.

To determine the dimensions of the critical design aircraft of a class, the following key figures need to be known for each aircraft type belonging to that class: The width of the fuselage, the wingspan, the width of the horizontal stabilizer, the linear positions of the wingbox, leading and trailing edges of the wingtip, and the overall length of the aircraft. The critical design aircraft of a class is initialized using the dimensions of the longest aircraft belonging to the class. Next, the maximum wingspan, the maximum fuselage width, and the maximum width of the horizontal stabilizer are identified from all aircraft belonging to the class and transferred to the critical design aircraft. Finally, the critical design aircraft is compared to each aircraft type belonging to the class and it is checked whether the outer shape of the respective aircraft is covered entirely by the outer shape of the critical design aircraft. In case the outer shape of the critical design aircraft does not entirely cover another aircraft of the class, its geometry is changed accordingly.

Aircraft safety envelopes We can now compute the safety envelope for the critical design aircraft of each class. Figure B.1a shows the aircraft safety envelope without minimum safety clearance for one aircraft.

The aircraft safety envelope as shown in Figure B.1a is defined by 6 pairs of points, where each pair is located symmetrically to the aircraft center line. As the aircraft is accommodated at a parking position, the aircraft center line equals the lead-in line of the parking position. Points (0, 1) are located at the nose of the aircraft, with a lateral distance of half the fuselage width (A) to the lead-in line. Points (4, 5) and (6, 7) are located in front of (F) and behind (G) the wingtips, where the lateral distance of each point from the lead-in line equals half the wingspan (C). The position of points (2, 3) is chosen such that the distance to the nose of the aircraft (E) is maximized while the edges (2, 4) and (3, 5) do not interfer with the engines or the wingbox of the aircraft. Points (8, 9) are located at the tail of the aircraft (H), where the lateral distance to the lead-in line equals the width of the horizontal stabilizer (B). Finally, the path to the taxiway is approximated



(a) Without safety clearance

(b) With safety clearance

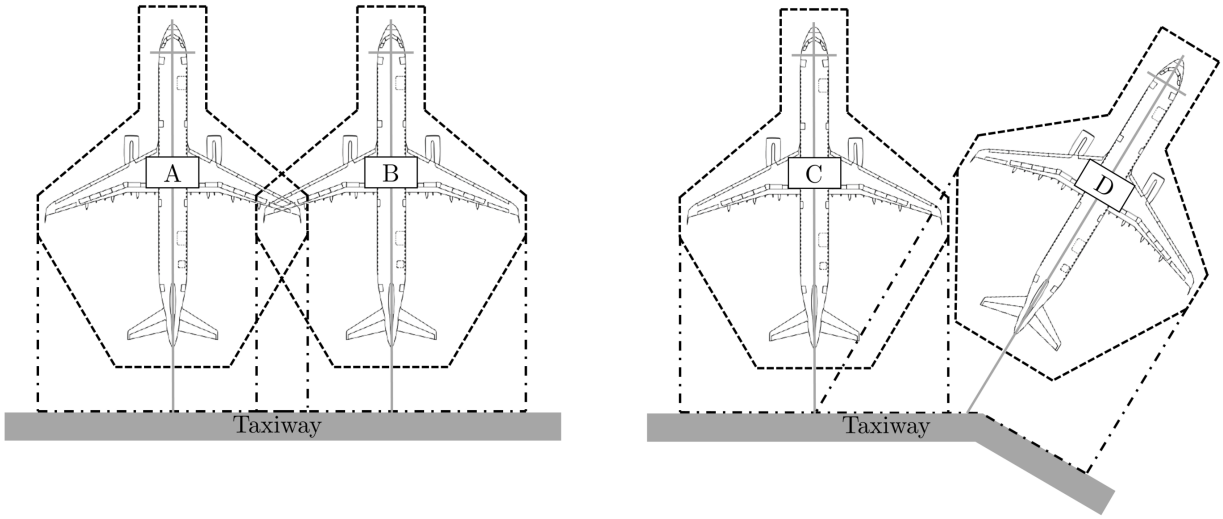
Figure B.1: Aircraft safety envelopes (Source of aircraft models: Airbus 2022)

using two additional points (10, 11). These points are positioned at the intersection of the lead-in line and the taxiway (I), and the lateral distance between the points and the lead-in line equals half the wingspan of the aircraft (C).

For our purpose, the aircraft safety envelope must be extended to include the minimum safety clearances, see Figure B.1b, where the safety clearance is added to both coordinates of each point, so that the points are shifted away from the aircraft.

Appendix C Collisions of aircraft safety envelopes

We consider two types of collisions, illustrated in Figure C.1. Figure C.1a shows the case where the safety envelopes of two parked aircraft overlap. Figure C.1b demonstrates the case where the safety envelopes of two aircraft parked adjacently do not overlap when both aircraft are in their parking positions, but where the pushback of aircraft D would lead to an infringement of the safety envelope of aircraft C.



(a) Collision when both aircraft are parked

(b) Collision when one aircraft is moving

Figure C.1: Collisions of aircraft safety envelopes (Source of aircraft models: Airbus 2022)

Appendix D Merging Constraints (1j)

We use the following algorithm to merge constraints of type (1j).

Algorithm D.1 Algorithm to merge Constraints (1j)

for all $k \in \mathcal{K}$ **do**
  Initialize matrix $\tilde{E}$ with dimensions L, L, A, A, all entries 0
  for all $l \in \mathcal{L}$ and $a \in \mathcal{A}$ **do**
    Initialize P_{la} as the list of tuples (m, b) for which $E_{lmab} = 1$
    for all tuples $(m, b) \in P_{la}$ **do**
      if $\tilde{E}_{lmab} = 1$ **then**
        Remove (m, b) from P_{la}
      end if
    end for
    while P_{la} is not empty **do**
      Initialize new constraint B as $\sum_{g \in \mathcal{G}: F_{lg}=1} x_{glak} \leq 1$
      Initialize $\hat{P}_{la}$ as an empty list of tuples, add (l, a) to $\hat{P}_{la}$
      for all tuples (m, b) in P_{la} **do**
        if $E_{nmcb} = 1$ for all tuples (n, c) in $\hat{P}_{la}$ **then**
          Add $\sum_{h \in \mathcal{G}: F_{mh}=1} x_{hmbk}$ to the left side of Constraint B
          $\tilde{E}_{nmcb} \leftarrow 1$ for all tuples (n, c) in $\hat{P}_{la}$
          Add (m, b) to $\hat{P}_{la}$
          Remove (m, b) from P_{la}
        end if
      end for
      Add Constraint B to the model
    end while
  end for
end for
Remove all Constraints (1j) from the model

Appendix E Subproblems

The subproblem for demand decomposition $c \in \mathcal{C}$ and area $s \in \mathcal{S}$ is defined as follows:

$$\min \; z_{2cs} = \sum_{g \in \mathcal{G}_s} \sum_{a \in \mathcal{A}} v_{ga} \tag{E.1a}$$

subject to

$$\sum_{g \in \mathcal{G}_s} \sum_{l \in \mathcal{L}_s : C_{al}=1 \cap F_{lg}=1} x_{glak} = r_{ap(c,s,k)} \qquad \forall a \in \mathcal{A}; k \in \mathcal{K} \tag{E.1b}$$

$$\sum_{g\in\mathcal{G}_s:F_{lg}=1}\sum_{a\in\mathcal{A}:C_{al}=1} x_{glak} \le y_l \qquad \forall l\in\mathcal{L}_s; k\in\mathcal{K} \qquad \text{(E.1c)}$$

$$\sum_{g\in\mathcal{G}_s:F_{lg}=1} u_{gl} = y_l \qquad \forall l\in\mathcal{L}_s \qquad \text{(E.1d)}$$

$$x_{glak} \le z_{gl} \qquad \forall g\in\mathcal{G}_s; l\in\mathcal{L}_s; a\in\mathcal{A}; k\in\mathcal{K}: \quad F_{lg}=1; C_{al}=1 \qquad \text{(E.1e)}$$

$$\sum_{l\in\mathcal{L}_s:F_{lg}=1}\sum_{a\in\mathcal{A}:C_{al}=1} a\cdot x_{glak} \le \sum_{a\in\mathcal{A}} v_{ga} \qquad \forall g\in\mathcal{G}_s; k\in\mathcal{K} \qquad \text{(E.1f)}$$

$$v_{ga} \le v_{gb} \qquad \forall g\in\mathcal{G}_s; a\in\{2,\dots,A\}; \quad b=a-1 \qquad \text{(E.1g)}$$

$$\sum_{l\in\mathcal{L}_s:F_{lg}=1}\left(\sum_{a\in\mathcal{A}^{\text{small}}:C_{al}=1} x_{glak} + \sum_{a\in\mathcal{A}^{\text{large}}:C_{al}=1} 2\cdot x_{glak}\right) \le 2 \qquad \forall g\in\mathcal{G}_s; k\in\mathcal{K} \qquad \text{(E.1h)}$$

$$\sum_{g\in\mathcal{G}_s:F_{lg}=1} x_{glak} + \sum_{h\in\mathcal{G}_s:F_{mh}=1} x_{hmbk} \le 1 \qquad \forall l,m\in\mathcal{L}_s; a,b\in\mathcal{A}; k\in\mathcal{K}: \quad E_{lmab}=C_{al}=C_{bm}=1; l<m \qquad \text{(E.1i)}$$

$$v_{ga} = H_{ga} \qquad \forall g\in\mathcal{G}_s; a\in\mathcal{A}: \sum_{\hat{a}\in\mathcal{A}} H_{g\hat{a}} > 0 \qquad \text{(E.1j)}$$

$$v_{ga}\in\{0,1\} \qquad \forall g\in\mathcal{G}_s; a\in\mathcal{A} \qquad \text{(E.1k)}$$

$$x_{glak}\in\{0,1\} \qquad \forall g\in\mathcal{G}_s; l\in\mathcal{L}_s; a\in\mathcal{A}; k\in\mathcal{K}: \quad F_{lg}=1; C_{al}=1 \qquad \text{(E.1l)}$$

$$y_l\in\{0,1\} \qquad \forall l\in\mathcal{L}_s \qquad \text{(E.1m)}$$

$$u_{gl}\in\{0,1\} \qquad \forall g\in\mathcal{G}_s; l\in\mathcal{L}_s: F_{lg}=1 \qquad \text{(E.1n)}$$

In contrast to Model (1a)-(1o), D_{ak} is substituted by parking patterns $\mathcal{P}_{cs}$. The q_{ak} variables are eliminated, because the optimal value of z_1 has been determined already and all aircraft contained in parking patterns $\mathcal{P}_{cs}$ have to be parked at contact gates. Demand constraints (1c) are changed to Constraints (E.1b) accordingly. Sets $\mathcal{G}$ and $\mathcal{L}$ are replaced throughout the model by the area-specific subsets $\mathcal{G}_s$ and $\mathcal{L}_s$.

Appendix F Algorithm to compute lower bounds for demand decompositions

Let $\omega_{csa}\in\mathbb{N}$ be the minimum number of gates equipped for class $a\in\mathcal{A}$ aircraft required in area $s\in\mathcal{S}$ to park all aircraft contained in $\mathcal{P}_{cs}$. Furthermore, let $\psi_{csak}\in\mathbb{N}$ be the number of class $a\in\mathcal{A}$ aircraft that can be parked at gates which are equipped for aircraft of class $a'>a$ in

area $s \in \mathcal{S}$ for demand pattern $k \in \mathcal{K}$ when aircraft are assigned to areas according to demand decomposition $c \in \mathcal{C}$. Then, Algorithm F.1 describes our procedure to calculate LB_{cs} in detail. First, we determine the minimum number of gates required for aircraft of the largest class A as $\max_{k \in \mathcal{K}} \left\{ r_{Ap(c,s,k)} \right\}$ in line 6. Next, we consider the next smaller aircraft class $A-1$. Here, for each demand pattern $k \in \mathcal{K}$, we first compute how many of the class $A-1$ aircraft contained in $p\,(c,s,k)$ can be parked at the gates equipped for class A aircraft, but not occupied by a class A aircraft for demand pattern k (lines 9 to 14). For the remaining aircraft, additional gates for class $A-1$ aircraft must be provided (line 15). We continue according to this scheme until all aircraft classes have been considered. As soon as the aircraft class under consideration is no longer included in $\mathcal{A}^{\text{large}}$ but in $\mathcal{A}^{\text{small}}$, we assume that the MARS mode can always be used, i.e., that two aircraft of class $a \in \mathcal{A}^{\text{small}}$ can be parked simultaneously at gates which are equipped for aircraft of the classes belonging to $\mathcal{A}^{\text{large}}$ (line 12).

Algorithm F.1 Algorithm to determine the value of LB^{cs}

```
1: Initialize LB_cs, ω_csa, and ψ_csak
2: for all aircraft classes a ∈ A (in descending order) do
3:    ω_csa ← 0
4:    for all demand patterns k ∈ K do
5:       if a = A then
6:          ω_csa ← max{ω_csa, r_ap(c,s,k)}
7:       else
8:          if r_ap(c,s,k) > ω_csa then
9:             if a ∈ A^large \ {A} then
10:               ψ_csak ← Σ_{a'∈A:a'>a} (ω_csa' − r_a'p(c,s,k))
11:            else if a ∈ A^small then
12:               ψ_csak ← Σ_{a'∈A:a'>a, a'∈A^large} 2 · (ω_csa' − r_a'p(c,s,k))
13:                       + Σ_{a'∈A:a'>a, a'∈A^small} (ω_csa' − r_a'p(c,s,k))
14:            end if
15:            ω_csa ← (r_ap(c,s,k) − ψ_csak)
16:         end if
17:      end if
18:   end for
19: end for
20: LB_cs ← Σ_{a∈A} a · ω_csa
21: return LB_cs
```

In brownfield settings, we compare the results from Algorithm F.1 with the already existing gates and adjust gate quantities ω_{csa} accordingly if necessary. Let parameter $\Omega_{sa} \in \mathbb{N}$ denote the number of existing gates in subarea $s \in \mathcal{S}$ that are equipped to handle aircraft of class $a \in \mathcal{A}$. Algorithm F.2 shows how we match the results from Algorithm F.1 with Ω_{sa} and adapt the values

of ω_{csa} accordingly if necessary.

Algorithm F.2 Algorithm to determine the value of LB_{cs} in brownfield scenarios

```
1: Initialize Ω_sa and integer counter variable f
2: Perform Alorithm F.1, lines 1 to 19
3: for all aircraft classes a ∈ A do
4:    Ω_sa ← Σ_{g∈G_s} H_ga
5: end for
6: for all aircraft classes a ∈ A (in descending order) do
7:    while Ω_sa > ω_csa do
8:       ω_csa ← (ω_csa + 1)
9:       if a ∈ A^small : a > 1 then
10:         for all a' ∈ A^small : a' < a (in descending order) do
11:            if ω_csa' > 0 then
12:               ω_csa' ← (ω_csa' − 1)
13:               break
14:            end if
15:         end for
16:      else if a ∈ A^large then
17:         f ← 2
18:         for all a' ∈ A^large : a' < a (in descending order) do
19:            if ω_csa' > 0 then
20:               ω_csa' ← (ω_csa' − 1)
21:               f ← 0
22:               break
23:            end if
24:         end for
25:         for all a' ∈ A^small : a' < a (in descending order) do
26:            if f > 0 and ω_csa' > 1 then
27:               ω_csa' ← (ω_csa' − 2)
28:               f ← 0
29:               break
30:            else if f > 0 and ω_csa' > 0 then
31:               ω_csa' ← (ω_csa' − 1)
32:               f ← (f − 1)
33:            end if
34:         end for
35:      end if
36:   end while
37: end for
38: LB_cs ← Σ_{a∈A} a · ω_csa
39: return LB_cs
```

According to Constraints (1p), already existing gates must not be changed in a brownfield scenario. Hence, ω_{csa} cannot be smaller than Ω_{sa} in any area $s \in \mathcal{S}$ and any aircraft class $a \in \mathcal{A}$. If

ω_{csa} has to be increased for that reason and $a > 1$, it follows from the downward compatibility of gates that $\omega_{cs\hat{a}}$ can be reduced for a smaller aircraft class $\hat{a} < a$. Again, we explicitly consider the MARS mode. That is, if ω_{csa} is increased by 1 for an aircraft class $a \in \mathcal{A}^{\text{large}}$, $\omega_{cs\hat{a}}$ for an aircraft class $\hat{a} \in \mathcal{A}^{\text{small}}$ can be reduced by 2.

Appendix G Algorithm to compute consistency of a demand decomposition

Let $n_{sk}^{a'} \in \mathbb{N}$ be defined as the aggregated number of aircraft belonging to classes $a', \ldots, A \in \mathcal{A}$ that must be parked at area $s \in \mathcal{S}$ for demand pattern $k \in \mathcal{K}$ when aircraft are assigned to areas according to demand decomposition $c \in \mathcal{C}$. The procedure we employ to determine whether a demand decomposition is consistent or not is provided in Algorithm G.1.

Algorithm G.1 Algorithm to determine whether a demand decomposition is consistent or not

Initialize n_{sk}^{a}
for all aircraft classes $a_{\min} \in \mathcal{A}^{\text{large}}$ **do**
  $n_{sk}^{a_{\min}} \leftarrow \sum_{a \in \mathcal{A}^{\text{large}}: a \geq a_{\min}} r_{ap(c,s,k)}$
  for all demand patterns $k_1 \in \mathcal{K}$ **do**
    for all demand patterns $k_2 \in \mathcal{K} : k_2 > k_1$ **do**
      if $\sum_{a \in \mathcal{A}^{\text{large}}: a \geq a_{\min}} D_{ak_1} \leq \sum_{a \in \mathcal{A}^{\text{large}}: a \geq a_{\min}} D_{ak_2}$ **then**
        if $n_{sk_1}^{a_{\min}} > n_{sk_2}^{a_{\min}}$ **then**
          return false ▷ demand decomposition is not consistent
        end if
      end if
    end for
  end for
end for
return true ▷ demand decomposition is consistent

Appendix H Algorithm to check feasibility of a solution from a relaxed subproblem

Let $\hat{y}_l$ and $\hat{u}_{gl}$ denote the values of variables y_l and u_{gl} in the current solution of the relaxed subproblem, respectively. Furthermore, let $\bar{\beta}_{ga} \in \mathbb{N}$ denote the number of class $a \in \mathcal{A}$ aircraft that are handled at gate $g \in \mathcal{G}_s$ for demand pattern $\bar{k}$ simultaneously. Then, we check compliance of a path π with Constraints (E.1d), (E.1f), and (E.1h) as shown in Algorithm H.1.

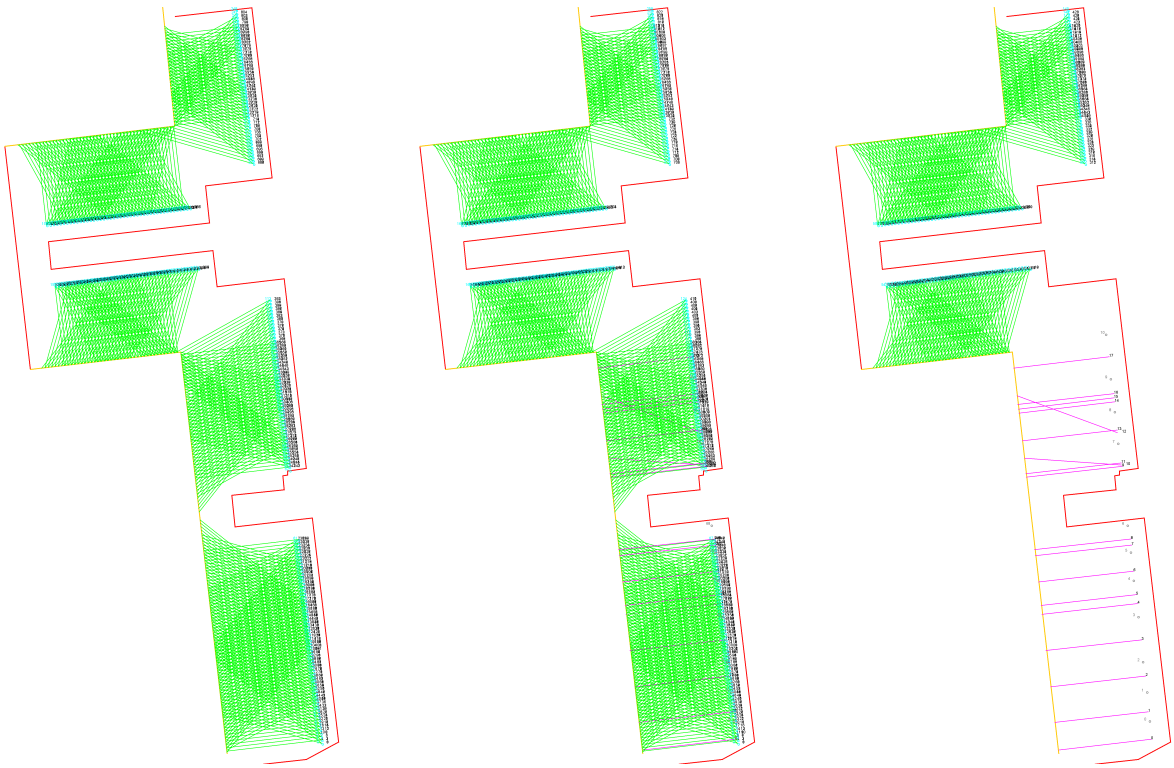
Algorithm H.1 Heuristic algorithm to check compliance of a path with Constraints (E.1d), (E.1f), and (E.1h)

Initialize $\bar{\beta}_{ga}$ and boolean variable ς
for all nodes in path π **do**
  $\varsigma \leftarrow$ false
  Get $l \in \mathcal{L}_s$ and $a \in \mathcal{A}$ represented by the node
  for all $g \in \mathcal{G}_s$ **if** $F_{lg} = 1$ and $\hat{v}_{ga} = 1$ **do** ▷ Constraints (E.1f)
    if not $(\hat{y}_l = 1 \wedge \hat{u}_{gl} = 0)$ **then** ▷ Constraints (E.1d)
      if $\sum_{a \in \mathcal{A}^{\text{small}}} \bar{\beta}_{ga} + 2 \cdot \sum_{a \in \mathcal{A}^{\text{large}}} \bar{\beta}_{ga} < 2$ **then** ▷ Constraints (E.1h)
        $\bar{\beta}_{ga} \leftarrow \bar{\beta}_{ga} + 1$
        $\varsigma \leftarrow$ true
        break
      end if
    end if
  end for
  if $\varsigma =$ false **then**
    return false ▷ solution is infeasible
  end if
end for
return true ▷ solution is feasible

Algorithm H.1 can be described as a greedy heuristic that tries to assign parked aircraft to the first available gate. Hence, there is no guarantee that Algorithm H.1 returns that a given solution is feasible if it actually *is* feasible; however, if the solution is actually *in*feasible, Algorithm H.1 will always return that it is infeasible.

Appendix I Planning scenarios for Munich Airport Terminal 1

The following figures illustrate the sets of gates $\mathcal{G}$ and lead-in lines $\mathcal{L}$ for Munich Airport Terminal 1 in the different planning scenarios, with $\Delta = 5\text{m}$ and $\kappa = 22.5°$. Figure I.1a depicts the greenfield scenario, Figure I.1b shows the soft brownfield case, and Figure I.1c displays the true brownfield instance. In all figures, physical obstacles that must not be touched by aircraft at any time are visualized by red lines, taxiways are represented by yellow lines. Furthermore, existing gates are marked by gray circles, all other gates are represented by cyan circles. Finally, existing lead-in lines are represented by purple lines, all other lead-in lines by green lines.

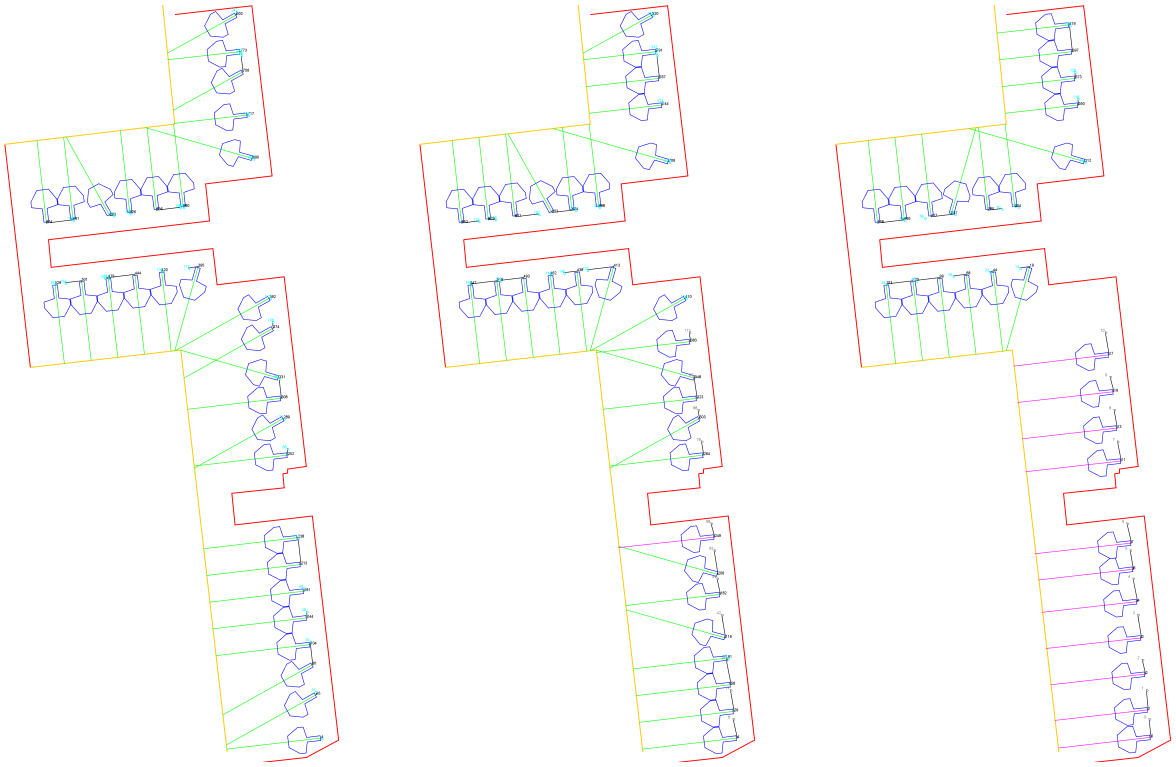


(a) Greenfield (b) Soft brownfield (c) True brownfield

Figure I.1: Lead-in lines and gates at Munich Airport Terminal 1 in different planning scenarios

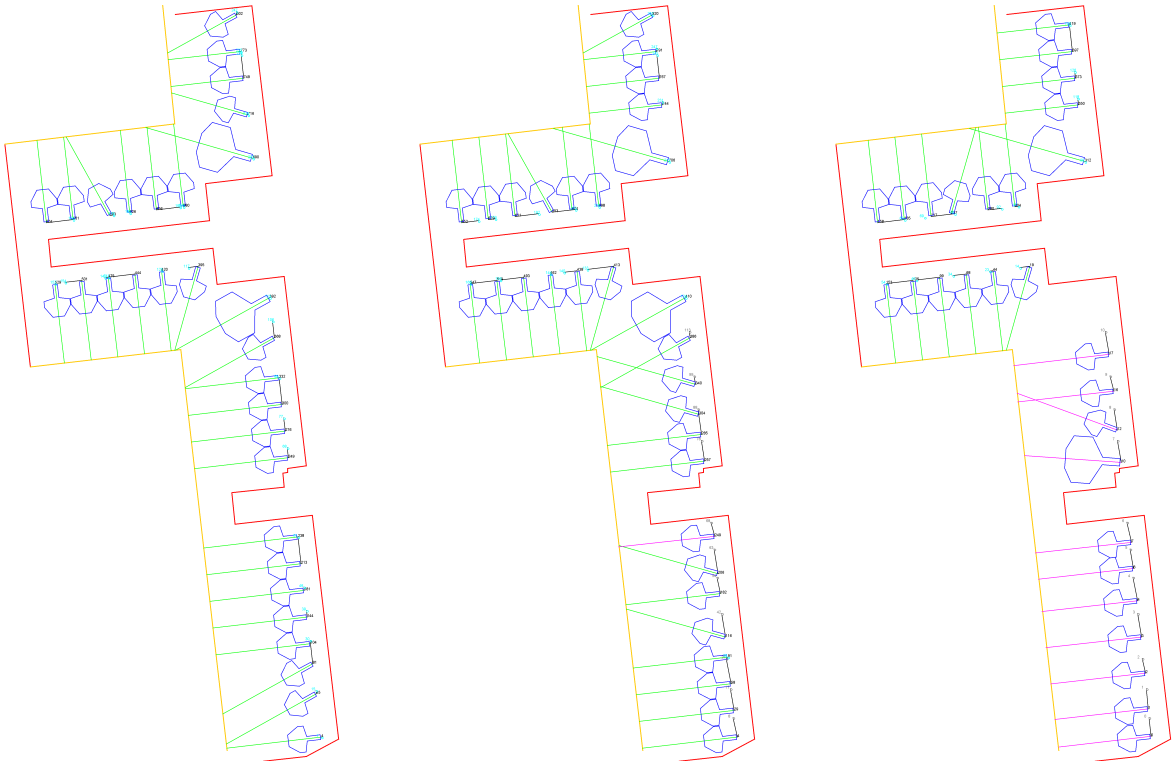
Appendix J Optimal layouts for Munich Airport Terminal 1, demand patterns 1-4

The following figures are analogous to Figure 5 and provide the optimal layouts of all planning scenarios for demand patterns 1 to 4.



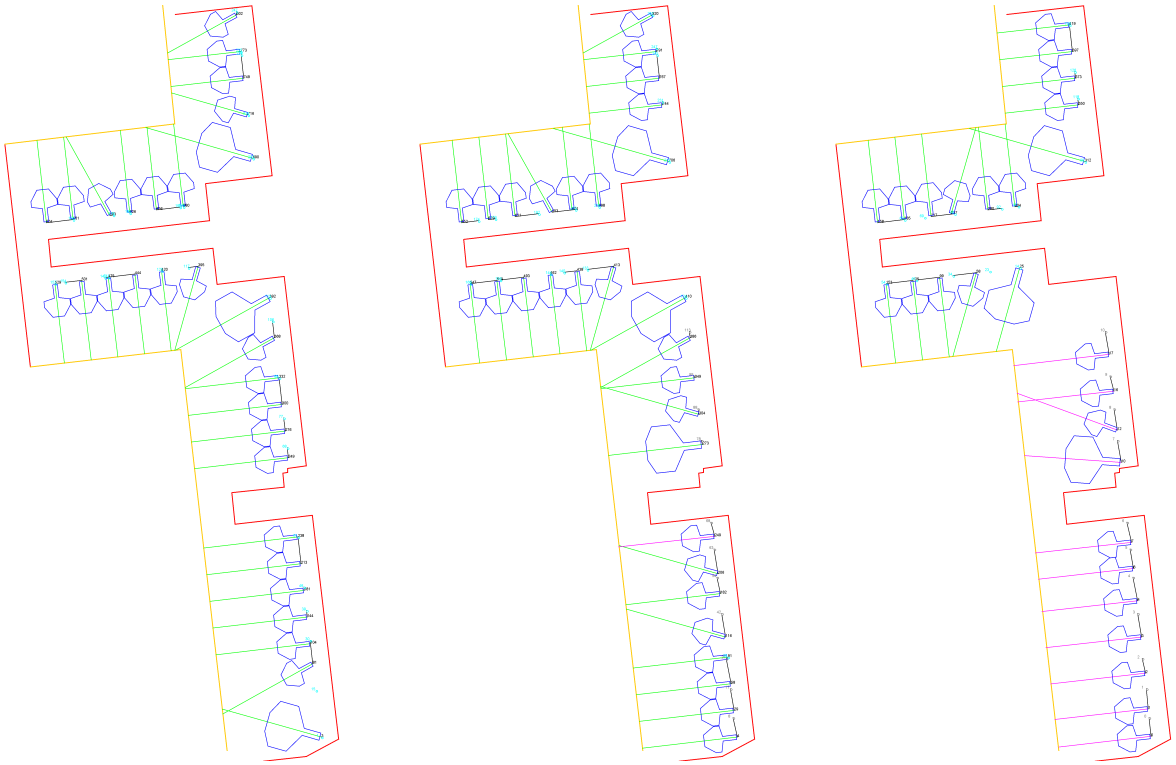
(a) Greenfield (b) Soft brownfield (c) True brownfield

Figure J.1: Optimal solutions for different planning scenarios, demand pattern 1



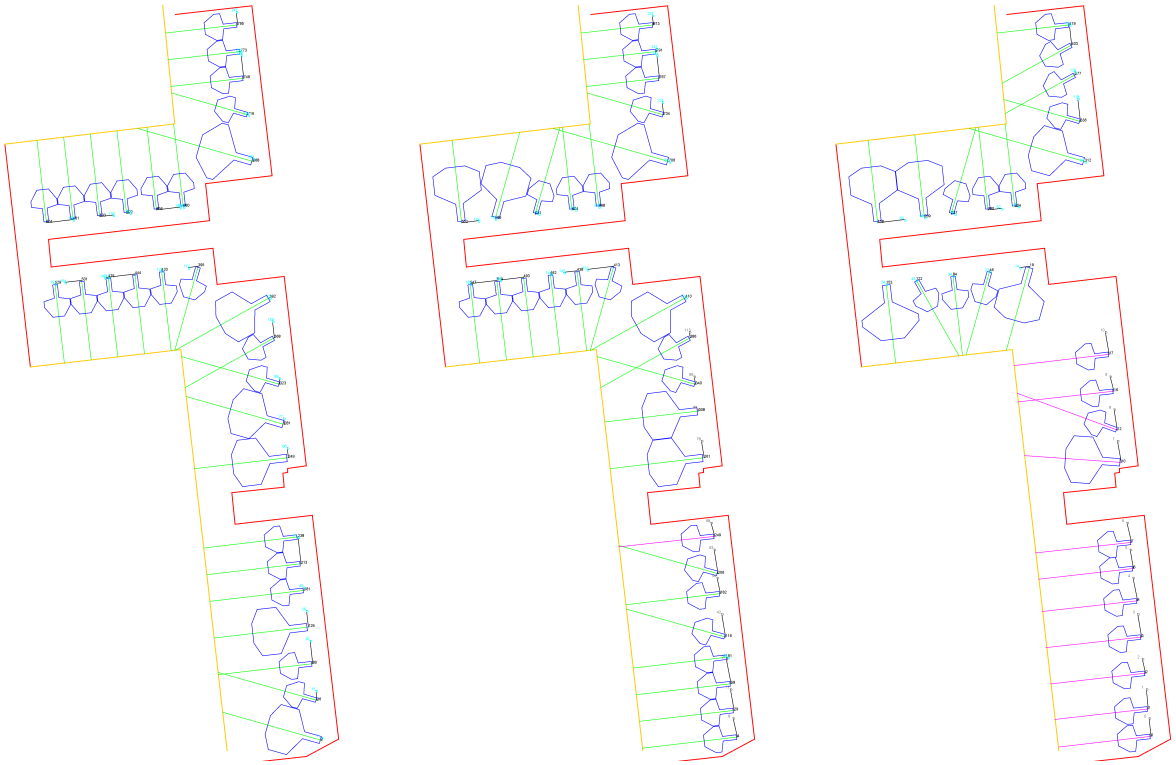
(a) Greenfield (b) Soft brownfield (c) True brownfield

Figure J.2: Optimal solutions for different planning scenarios, demand pattern 2



(a) Greenfield (b) Soft brownfield (c) True brownfield

Figure J.3: Optimal solutions for different planning scenarios, demand pattern 3



(a) Greenfield (b) Soft brownfield (c) True brownfield

Figure J.4: Optimal solutions for different planning scenarios, demand pattern 4