Operational Research (2022) 22:135–163
https://doi.org/10.1007/s12351-020-00563-9

ORIGINAL PAPER

Integer and constraint programming model formulations for flight-gate assignment problem

M. Arslan Ornek[1] · Cemalettin Ozturk[2] · Ipek Sugut[3]

Received: 16 January 2019 / Revised: 8 March 2020 / Accepted: 23 March 2020 / Published online: 8 April 2020

Abstract

Flight-gate assignment problems are complex real world problems involving different constraints. Some of these constraints include plane-gate eligibility, assigning planes of the same airline and planes getting service from the same ground handling companies to adjacent gates, buffers for changes in flight schedules, night stand flights, priority of some gates over others, and so on. In literature there are numerous models to solve this highly complicated problem and tackle its complexity. In this study, first, we propose two different integer programming models, namely, timetabling and assignment based models, and then a scheduling based constraint programming model to solve the problem to optimality. These models prove to be highly efficient in that the computational times are quite short. We also present the results for one day operation of an airport using real data. Finally, we present our conclusions based on our study along with the possible further research.

Keywords Flight-gate assignment problem · Integer programming · Constraint programming

✉ Cemalettin Ozturk
OzturkC@rtx.com

M. Arslan Ornek
arslan.ornek@yasar.edu.tr

Ipek Sugut
sugut@thy.com

[1] Department of Industrial Engineering, Yasar University, Izmir 35100, Turkey

[2] Raytheon Technologies, United Technologies Research Center Ireland, Cork, Ireland

[3] Turkish Airlines, Istanbul, Turkey

1 Introduction and literature survey

A gate is an important resource at the disposal of an airline in a hub-spoke airport. With an intense air-traffic increase in recent years (doubled since early 1980s), improper assignment of gates may result in flight delays, inefficient use of the resource, customer's dissatisfaction and hence the need to efficiently use these to reduce operating costs, increase passenger satisfaction, and alleviate congestion has become more prevalent in modern times (de Man 2011; Ballis 2002). The flight gate assignment problem is encountered by gate managers at an airport on a periodic basis, usually daily. This assignment should be made so as to balance carrier efficiency and passenger comfort, while providing buffers for unexpected events that cause assignment disruptions. A typical metropolitan airport has more than fifty gates and handles hundreds of flights a day for thousands of passengers. Gate assignment is a complicated and over-constrained problem as it deals with a wide range of interdependent resources including aircrafts, gates, gate facilities, and crews (Ding et al. 2005).

Airport gate assignment is a critical issue for the operation management of an airport. It means assigning flights to gates according to their real-time arrival time and departure time, such that each flight is assigned to exactly one gate, and there is no conflict between two consecutive flights assigned to the same gate. Determining which aircraft is assigned to which gate is the Gate Assignment Problem (GAP). Since aircraft hardly ever arrives/departs on time, an assignment plan for the upcoming day is to be prepared, based on currently available flight-schedule information in such a way that a reasonable deviation from the scheduled arrival/departure of any of the flights does not result in an infeasible assignment plan. The assignment plan should be robust against such small deviations during the actual day of operations. Typically, gates are first pre-assigned to the scheduled arriving and departing aircrafts ahead of time to ensure a smooth operation. Although gate assignment problem produces only a static schedule, it provides a basis for making last-minute changes to handle operational uncertainty caused by unexpected events such as flight delays, machine failures, and severe weather conditions.

Different methods have been developed to solve this problem with different objectives. For passengers, the objectives include, to name a few, minimizing passenger walking distance and waiting time; for flights and gates, minimizing connection times, cargo handling costs of the gate assignments, slack times, i.e., idle time between two successive assignments of the gate, number of gate conflicts of any two adjacent flights that are assigned to the same gate, aircraft waiting time on the apron, number of ungated flights, off-gate events. Other objectives considered are analyzing the effects of stochastic flight delays on static gate assignments, maximizing the preferences of total gate assignment, the robustness of the resulting schedule and finding gate assignment efficiency which represents rational compromises between waiting time for gate and apron operations (Bouras et al. 2014; Cheng et al. 2012).

From a mathematical view, GAP has been formulated as integer, binary, or mixed integer, general linear or nonlinear models. Some publications proposed specific formulations as binary or mixed binary quadratic models. Quadratic assignment

problem (QAP), clique partitioning problem (CPP), and scheduling problem, which are well-known related problems in combinatorial optimization have been used to formulate GAP. From a different point, stochastic or robust optimization was used in few publications. The use of exact solution methods is certainly preferable, however, the use of meta-heuristics is also common (Cheng et al. 2012).

Lim et al. (2005) formulated GAP as an integer programming model and proposed two models with time windows. The first model minimised the passenger walking distance and the second model cargo handling costs of the gate assignments. They used an IP solver to find the optimal solution in the first model; however several heuristic algorithms were used to generate solutions in the second model. According to the results, heuristics gave better results than the IP solver in both CPU time and solutions quality. To minimize the passenger walking distance, the other reference, Bihr (1990), developed a binary integer model. This model was used to solve a sample problem using primal–dual simplex algorithm. Bolat (1999), developed a mixed integer program for GAP with the objective of minimizing the slack times. Li (2009) developed a nonlinear binary mixed integer model with a constraint programming (Dechter 2003) which minimizes the number of gate conflicts of any two adjacent flights that are assigned to the same gate. A study for a stochastic GAP was designed by Yan and Tang (2007). In this analysis, the flight delays are stochastic. It had three parts: the gate assignment model, a rule for the reassignments, and two adjustment methods for penalties. The performance was analyzed and evaluated by a simulation-based method. Drexl and Nikulin (2008), formulated the multi-criteria airport gate assignment as a quadratic assignment problem (QAP) and solved it using Pareto simulated annealing. The objectives are minimizing connection times or total passenger walking distances, maximizing the preferences of total gate assignment, and minimizing the number of ungated flights. Li (2010) modeled GAP as a parallel machine scheduling problem and applied dynamic scheduling and the direct graph model to solve it. For solving the small size problems, branch and bound was used while the large size problems were solved by using dynamic scheduling. Diepen et al. (2012) modeled a completely new IP formulation with a robust objective function expressed as the maximization of an allocation of a maximum possible idle time between each pair of consecutive flights such that the probability of flights overlapping is minimised and the gate schedule becomes more robust.

As an example to multi-objective gate assignment study, Deng et al. (2017) considered the minimum walking distances of passengers, the minimum idle time variance of each gate, the minimum number of flights at parking apron and the most reasonable utilization of large gates as the optimization objectives. They solved the developed model using an improved adaptive particle swarm optimization (DOAD-APO) algorithm based on making full use of the advantages of Alpha-stable distribution and dynamic fractional calculus.

For more references on the types of the problem and the relevant solution approaches to assigning flights to gates, see (Sügüt 2016). Finally, not only assigning incoming flights to external gates but also to internal gates (aka common use check-in counters) is quite important in smooth running of airport operations (Lee et al. 2016; Ornek et al. 2019).

This paper considers gate assignment problem for a real medium size airport, which is regarded to be a highly complex problem. There are various considerations that are involved while assigning gates to incoming and outgoing flights at an airport. Gates have different restrictions, such as adjacency, LIFO and push time, which is known in advance based on the structure of the airport. As aforementioned, many different objective functions have been used in literature. In this paper, we propose a novel objective function, different from those found in literature, based on gate utility (importance) concept to reflect practical considerations of the airport authority. The basis of the utility function is that a utility associated with a gate and the utility of a flight being assigned to that specific gate determines together how flights are allocated to the gates. Besides, we propose two novel optimization models, namely, Timetabling and Assignment based integer programming models (IP) and a scheduling based constraint programming model (CP) subject to airport specific business rules formulated as constraints. We solve these models, present, and compare the results for daily operations of an airport using real data sets. The rest of the paper is organised as follows. Section 2 defines the problem and develops the model formulations. Section 3 discusses the performance evaluations of the proposed formulations and provides numerical examples of which data are obtained from a medium-sized airport. Finally, we present our conclusions and elaborate on future research directions.

2 Problem definition and the model formulations

The airport under consideration has a number of open park areas and bridge-equipped gates. Airport management prefers flights to be assigned to bridge-equipped gates as it facilitates embarking and disembarking of passengers. Also, after aircrafts arrive, they need to be refueled, replenished, all the waste has to be taken off-board. If all gates are engaged, then flights are to be assigned to open park areas. Also, night stand flights are assigned to open park areas. Some gates are for emergencies only. These are large enough for allocation of larger planes. For instance, if the bridges 26th and 42nd are full, large planes are assigned to 24th or 25th bridge-equipped parking area. Some airlines have a priority to be assigned to the same gates. Normally, no other flight is assigned to those gates unless that gate is available.

Airline companies that use the same ground handling services firms are assigned adjacent to each other in order to prevent apron traffic. Departure and arrival of a plane is also considered. For instance, if a plane's departure is international, it has a priority for bridge-equipped parking areas in the international terminal. Similarly, if its departure is domestic, it has a priority in the domestic terminal. Some gates have priority due to their proximity to facilities in the airport. Not every plane fits in every parking area. Hence, some flights cannot be assigned to some parking areas, which we call plane-gate eligibility. We consider improving gate utilization as our primary objective.

In this study, the effectiveness of the gate assignment to a flight is measured by the term "utility". In other words, a utility value shows how appropriate a gate is

for the flight. Maximizing the total utility of the flight-gate assignment under some restrictions is the main objective for the IP formulations. Each gate has a utility value and also each gate takes a different utility value based on the flight assigned to it. These utility values are defined after many observations and the meeting with the operation managers. Therefore, the product of these utility values gives us the total utility of this assignment problem for the IP formulations.

Due to combinatorial nature of the problem, we propose two different integer programming (IP) formulations, namely Timetabling and Assignment based. The basic difference between these two formulations is the number of variables created when a flight is assigned to a gate. Time-tabling model creates number of variables equal to the number of periods a flight remains connected to a gate, whereas the assignment model creates just one when a flight is assigned to (arrives at) the gate. First, we develop a general time-tabling based IP formulation, and then we introduce the tuple concept to dramatically reduce the number of variables unnecessarily created by the model. Next, we provide assignment formulation based on the tuple concept, and then compare their performances. Furthermore, due to its success in solving combinatorial problems, we also propose a constraint programming (CP) based model formulation with customized search heuristic.

Before giving the IP and CP model formulations, we introduce the notation used throughout the paper.

Sets and indices

i: Index of periods, $i \in S = \{1, 2, \ldots, |S|\}$
j, b: Index of flights, $j, b \in U = \{1, 2, \ldots, |U|\}$
k, r: Index of parking areas, $k, r \in N = \{1, 2, \ldots, |M|, |M+1|, \ldots, |N|\}$
Where the first $|M|$ parking areas represent the bridge-equipped parking areas in which night-stand planes cannot be assigned to,
d: Index of parking areas that are occupied from the previous day, $d \in D \subset N$.
c: Index of ground service firms, $c \in C = \{1, 2, \ldots, |C|\}$
y: Index of night-stand flights, $y \in Y = \{1, 2, \ldots, |Y|\}$

Parameters

a_j: Scheduled arrival period of flight j, $a_j \in S$
g_j: Scheduled departure period of flight j, $g_j \in S$.
It is assumed that flight j left the airport in period $(g_j - 1)$. Hence, the same parking area can be used by another flight starting from the beginning of g_j. In other words, any flight occupies the assigned parking area in time interval $[a_j, g_j)$. Note that buffer periods for changes in flight schedules are also added to the g_j.
f_j: Ground Service Company of flight j, $f_j \in C$

$$L_{kr} = \begin{cases} 1 & \textit{If parking areas } k \textit{ and } r \textit{ are adjacent} \\ 0 & o/w \end{cases}$$

$$B_{jk} = \begin{cases} 1 & \textit{If flight } j \textit{ can be assigned to the parking area } k \\ 0 & o/w \end{cases}$$

t_k: Earliest available period of parking area $k \in D$, $t_k \in S$
W_{jk}: Utility of assigning flight j to parking area k, $W_{jk} \in \mathbb{R}^+$
w_k: Utility of parking area k, $w_k \in \mathbb{R}^+$

The decision maker should determine the values of the coefficients W_{jk} and w_k. In calculating the appropriate values, pairwise comparisons of objectives were used as a good starting point. This comparison scheme is employed in Analytic Hierarchy Process (AHP) (Saaty 1977). AHP is a method of multi-criteria decision making through pairwise comparisons derived from the judgments of experts. The comparisons are made on scales that represent decisions among the criteria. With the use of AHP, the inclusion of intangible factors in the objective function is also possible (Öztürk et al. 2017).

Decision variables

$$x_{ijk} = \begin{cases} 1 & \textit{If flight } j \textit{ is assigned to parking area } k \textit{ in period } i \\ 0 & o/w \end{cases}$$

2.1 Timetabling based integer programming model

In this section, we provide a Timetabling based IP model for the flight-gate assignment problem. Although, we assume parking areas as the limited resources and flights (i.e., aircrafts) as the resource consumers as in Li (2009), different from the literature, in this model we initialize a variable for each flight at each eligible parking area during the service time. In other words, if a flight j can be assigned to parking area k, we define (g_j–a_j) binary variables for flight j at parking area k for periods $[a_j, g_j)$.

$$\text{Maximize} \quad \sum_{j \in U} \sum_{k \in N} W_{jk} w_k x_{a_j jk} \tag{1}$$

s.t.

$$\sum_{k \in N} x_{a_j jk} = 1, \quad \forall j \in U \tag{2}$$

$$x_{ijk} = 0, \quad \forall i \in S, \quad \forall j \in U, \quad \forall k \in N | (i < a_j) \vee (i \geq g_j) \tag{3}$$

$$\sum_{j \in U} x_{ijk} \leq 1, \quad \forall i \in S, \quad \forall k \in N \tag{4}$$

$$\sum_{i \in S | a_j \le i < g_j} x_{ijk} = (g_j - a_j) x_{a_j jk}, \quad \forall j \in U, \quad \forall k \in N \tag{5}$$

$$\begin{aligned} & x_{ijr} + x_{ibk} \le 1, \quad \forall i \in S, \forall j, b \in U, \forall k, r \in N | (j < b) \\ & \wedge [(a_j \le i < g_j) \vee (a_b \le i < g_b)]^{\wedge} (f_j \ne f_b) \\ & \wedge (L_{kr} = 1) \wedge [(a_b \le a_j \wedge g_b > a_j) \vee (a_j \le a_b \wedge g_j > a_b)] \end{aligned} \tag{6}$$

$$x_{a_j jk} = 0, \quad \forall j \in Y, \quad \forall k \in M \tag{7}$$

$$x_{a_j jk} = 0, \quad \forall j \in U, \quad \forall k \in D | a_j \le \mathrm{t}_k \tag{8}$$

$$x_{a_j jk} \le B_{jk}, \quad \forall j \in U, \quad \forall k \in N \tag{9}$$

$$x_{ijk} \in \{0, 1\} \quad \forall j \in U, \quad \forall i \in S, \quad \forall k \in N \tag{10}$$

Objective function (1) maximizes total utility of flight-gate assignment plan which is a product of the utility (importance) of a gate the utility of a flight being assigned to that gate. Constraints (2) ensure that each flight is assigned to exactly one gate at its arrival time. Constraints (3) forbid assigning a flight to a gate before its arrival and after its departure. Note that if a flight's arrival and departure times are a_j and g_j respectively, the plane of that flight will occupy that gate from start of a_j to the start of g_j. For example, assume that a flight's (e.g., flight 7) arrival and departure times are 64th and 72nd time periods, respectively. Then, the corresponding Gantt chart for the assigned gate (e.g., gate 29) will be as in the following (see Fig. 1).

All binary variables corresponding to periods between 64 (included) and 72 (excluded) equal 1 in the solution and others 0.

$$\sum_{i=a_j\,(i=64)}^{g_j-1\,(72-1=71)} x_{i,7,29} = (g_7 - a_7) x_{64,7,29} \quad x_{64,7,29} + x_{65,7,29} + \cdots + x_{71,7,29} = 8 = 72 - 64$$

In other words, if a flight arrives in period i, it indicates that the assigned gate is used (i.e., occupied by that plane) in between i and $i+1$. Constraints (4) ensure that a parking area is occupied by at most one flight at any period. Constraints (5) guarantee that the same parking area is used during the service period of the flight [see the above numerical example given for constraints (3)]. Since the variable is equal

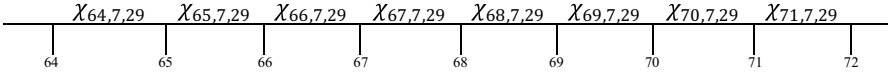

Fig. 1 Partial schedule of Parking Area 29 between the beginning of 64th and 72nd periods

to 1 for only one gate [due to constraints (2)], right hand side of constraints (5) will be equal to the service period of the flight for only the assigned parking area. And in that case, left hand side of the constraint will enforce that sum of the binary assignment variables in the consecutive periods will be equal to the service time period. Similarly, if flight j is not assigned to parking area k at its arrival period a_j, right hand side of the constraint (5) will be "0" and therefore enforce that the sum of the binary variables for the next consecutive periods during the service period will be equal to zero. In other words, that flight cannot be assigned to those periods. Note that this constraint is formulated for the periods between the arrival and departure times of the flight and is executed once for each flight and parking area when the period is equal to that flights' arrival period. Constraints (6) ensure that any two flights that are served with different companies and their service periods overlap must be assigned to non-neighbour parking areas. Constraints (7) forbid assigning night-stand flights to bridge-equipped parking areas, while constraints (8) ensure that a parking area can't be used before it becomes available. Constraints (9) guarantee that each flight is assigned to only eligible parking areas. Finally, constraints (10) give variable domain. Right hand side of Constraints (6) requires further explanation as in the following:

($j < b$): Generate this constraint for flight numbers $j < b$. Do not generate for $j > b$.
$[(a_j \leq i < g_j) \vee (a_b \leq i < g_b)]$: Generate this constraint for the largest time interval from the earliest arrival to the latest departure of the flights j and b. In other words, i is within the following interval $\min(a_j, a_b) \leq i < \max(g_j, g_b)$.
($f_j \neq f_b$): Generate this constraint if and only if flights j and b belong to different ground service companies.
($L_{kr} = 1$): Generate this constraint for only adjacent parking areas.
$[(a_b \leq a_j \wedge g_b > a_j) \vee (a_j \leq a_b \wedge g_j > a_b)]$: Generate this constraint if and only if flight j arrives while flight b is already in the airport and departure of b is later than arrival of j ($a_b \leq a_j \wedge g_b > a_j$) or vice versa ($a_j \leq a_b \wedge g_j > a_b$). This filtering guarantees that only overlapping periods are considered.

The proposed model is bounded with $|S|\mathrm{x}|U|\mathrm{x}|N|$ variables and $|S|\mathrm{x}|U|^2\mathrm{x}|N|^2$ constraints.

Next, we provide a time-tabling formulation based on the tuple concept. To prevent generating binary variables for non-eligible gates and the periods that the flight j does not stay in the airport, we define the following feasible parking period, flight and parking area triplets.

z, z' Index of all feasible period, flight and park assignment triplets,

$$z, z' \in Assignments \subset S \times U \times N$$
$$= \left\{ \begin{array}{l} < period, flight, park >, \quad \forall period \in S, \quad \forall flight \in U, \quad \forall park \in N| \\ (a_{flight} \leq period < g_{flight}) \wedge (B_{flight,park} = 1) \wedge \neg(flight \in Y \wedge park \in M) \wedge \\ \neg(park \in D \wedge a_{flight} \leq t_{park}) \end{array} \right\}$$

where $z \cdot period, z \cdot flight$ and $z \cdot park$ refer to the allowed period, flight and park indices for the corresponding assignment triplet z, respectively. The filters used to create only feasible triplets are explained as follows.

($a_{flight} \leq period < g_{flight}$): A feasible assignment variable (triplet) is generated for each flight at each park for each period interval while the corresponding flight is being served on the ground. In other words, (g_j-a_j) assignment triplets are generated for each eligible parking area to represent duration of stay for each flight.
$B_{flight,park} = 1$: Only eligible flight-parking area tuples are generated.
$\neg(flight \in Y \wedge park \in M)$: Any flight-parking area assignment is excluded if the flight is a night-stand one and the corresponding parking area is bridge-equipped.
$\neg(park \in D \wedge a_{flight} \leq t_{park})$: Any flight-period-parking area assignment is excluded if the parking area has been occupied from the previous day until it becomes available.

Feasible triplets generated in *Assignments* set are demonstrated in the following table for two flights.

Based on these feasible triplets $z \in Assignments$, we introduce binary variables for each of them to be used in the Timetabling based model formulation.

$$x_z = \begin{cases} 1 \ If z \cdot flight \text{ is assigned to } z \cdot park \text{ in period } z \cdot period \\ 0 \ o/w \end{cases}$$

Timetabling Based IP model

$$Maximize \sum_{z \in Assignments | z.period = a_{z \cdot flight}} W_{z \cdot flight, z.park} w_{z.park} x_z \tag{11}$$

Subject to

$$\sum_{z \in Assignments | (z \cdot flight = j) \wedge (z.period = a_j)} x_z = 1 \quad \forall j \in U \tag{12}$$

$$\sum_{z \in Assignments | (z \cdot period = i) \wedge (z \cdot park = k)} x_z \leq 1 \quad \forall i \in S, \quad \forall k \in N \tag{13}$$

$$\sum_{z' \in Assignments | (z' \cdot flight = z \cdot flight) \wedge (z' \cdot park = z \cdot park) \wedge (a_{z \cdot flight} \leq z' \cdot period < g_{z \cdot flight})} x_{z'} = (g_{z \cdot flight} - a_{z \cdot flight}) x_z$$
$$\forall z \in Assignments | z \cdot period = a_{z \cdot flight} \tag{14}$$

$$x_z + x_{z'} \leq 1 \quad \forall z, z' \in Assignments \Big| (z \cdot flight < z' \cdot flight) \wedge (z \cdot period = z' \cdot period) \wedge$$
$$(f_{z \cdot flight} \neq f_{z' \cdot flight}) \wedge (L_{z \cdot park, z' \cdot park} = 1) \wedge$$
$$[(a_{z' \cdot flight} \leq a_{z \cdot flight} \wedge g_{z' \cdot flight} > a_{z \cdot flight}) \vee (a_{z \cdot flight} \leq a_{z' \cdot flight} \wedge g_{z \cdot flight} > a_{z' \cdot flight})] \tag{15}$$

$$x_z \in \{0, 1\} \quad \forall z \in Assignments \tag{16}$$

Objective function (11) maximizes total utility of flight-gate assignment plan. Constraints (12) assign each flight to exactly one gate at its arrival time. Constraints (13) ensure that a parking area is occupied by at most one flight at any period. Constraints (14) guarantee that the same parking area is used during the service period of the flight. Since x_z variable is equal to "1" for only one parking area, [due to constraints (12)], right hand side of constraint will be equal to the service period of the flight for only the assigned parking area. And in this case, left hand side of the constraint will enforce that sum of the binary assignment variables in the consecutive periods will be equal to the total number of service time for the assigned parking area *z.park*. Similarly, if corresponding flight *z·flight* is not assigned to the corresponding parking area *z.park* at its arrival period $a_{z\cdot flight}$ (which is also equal to the $z \cdot period$ due to the constraint filtering), right hand side of the constraint (14) will be "0" and therefore it enforces that the sum of the binary variables for the next consecutive periods during the service period will be equal to zero for the corresponding parking area *z.park*. In other words, flight *z·flight* cannot be assigned to periods between $[a_{z\cdot flight}, g_{z\cdot flight}]$. The following example demonstrates the use of the constraint where flight 7 is assigned to parking area 29 when it arrived to the airport at the beginning of period 64 and scheduled to take off at period 71. Note that, due to the flight safety, we assume that the parking area is occupied by the same aircraft during the period of take-off and therefore, we assume take-off period of flight 7 as 72nd period. Hence, parking area 29 is available to a new flight at the beginning of period 72. In other words, parking area 29 is occupied by flight 7 during time interval (64, 72). Partial schedule of parking area 29 is demonstrated in Fig. 1.

Flight 7 starts using parking area 29 at the beginning of period 64, i.e. its scheduled arrival period. Hence, corresponding binary variable which shows the assignment of flight 7 to parking area 29 at the beginning of period 64 is equal to 1 ($x_{<64,7,29>} = 1$). Note that, $x_{<64,7,29>} = 1$ assignment also means that parking area 29 is used by flight 7 from the beginning to the end of the 64th period. Since the parking area is occupied by flight 7 during the next 8 periods (g_j–a_j = 72–64 = 8), corresponding binary variables for those periods are also equal to 1 due to constraints (4). $x_{<64,7,29>} + x_{<65,7,29>} + \cdots + x_{<71,7,29>} = (g_7 - a_7)x_{<64,7,29>} \sum_{i=a_7}^{g_7} x_{<i,7,29>} = (g_7 - a_7)$ $x_{<64,7,29>}$ which means $\sum_{i=64}^{72} x_{<i,7,29>} = (72 - 64)x_{<64,7,29>}$ or $\sum_{i=64}^{72} x_{<i,7,29>} = 8x_{<64,7,29>}$. This equation holds if and only if $x_{<64,7,29>} = x_{<65,7,29>} = \cdots x_{<71,7,29>} = 1$ or 0. If a feasible solution results that the flight 7 is assigned to parking area 29 at its arrival ($x_{<64,7,29>} = 1$), all consecutive binary variables in the equation are also equal to 1 ($x_{<64,7,29>} = x_{<65,7,29>} = \cdots = x_{<71,7,29>} = 1$). Constraints (14) also guarantee that if flight 7 is not assigned to parking area 29 at its arrival period ($x_{<64,7,29>} = 1$), all other consecutive binary variables on this parking area are also equal to 0, $x_{<64,7,29>} = x_{<65,7,29>} = \cdots = x_{<71,7,29>} = 0$.

Constraints (15) ensure that for any two possible assignments, if time periods are overlapping and corresponding ground service companies are different, they are assigned to non-adjacent parking areas. Right hand terms that are used to filter

possible assignment tuples to be constrained in inequality (15) are explained in the following:

$z \cdot flight < z' \cdot flight$: Prevents comparing any two flights twice.
$z \cdot period = z' \cdot period$: Ensures that constraints are generated for periods such that both flights are in the airport.
$f_{z \cdot flight} \neq f_{z' \cdot flight}$: Guarantees that constraints are generated if ground service companies of the corresponding flights are different.
$L_{z \cdot park, z' \cdot park} = 1$: States that constraints are generated if and only if parking areas of the corresponding assignments are adjacent.
$[(a_{z' \cdot flight} \leq a_{z \cdot flight} \wedge g_{z' \cdot flight} > a_{z \cdot flight}) \vee (a_{z \cdot flight} \leq a_{z' \cdot flight} \wedge g_{z \cdot flight} > a_{z' \cdot flight})]$: These two possible assignments are considered in the constraint if and only if flight z·*flight* arrives while $z' \cdot flight$ has been already in the airport or vice versa.

Finally, constraints (16) give variable domains. Note that, due to the filtering in the definition of the allowed assignments, we do not need to include original constraints (3), (7), (8) and (9) in the tuple formulation.

2.2 Assignment based integer programming model

In the second alternative IP model formulation, to reduce the number of binary variables, we define binary variables for only those periods where corresponding flights arrive at the airport. Before giving the new IP model, we introduce re-defined sets, indices and variables as in the following.

z, z' Index of all feasible flight, park, period assignment triplets,

$$z, z' \in Assignments \subset S \times U \times N = \left\{ \begin{array}{l} < period, flight, park >, \quad \forall period \in S, \quad \forall flight \in U, \quad \forall park \in N | \\ (period = a_{flight}) \wedge (B_{flight,park} = 1) \wedge \neg(flight \in Y \wedge park \in M) \wedge \\ \neg(park \in D \wedge a_{flight} \leq t_{park}) \end{array} \right\}$$

As stated in $(period = a_{flight})$ filtering operator, a single <*period*, *flight*, *park*> triplet is created for each flight and parking area. Changes in the revised *Assignments* set compared to Table 1 is demonstrated in the following table where feasible triplets are generated for each flight and eligible parking area for only the arrival period of the flight (Table 2).

Furthermore, we need the following set to formulate constraints for occupying the same parking area during the duration of stay for each flight.

h: Index of feasible flight-park pairs.

$$h \in FlightParkPairs = \{< z \cdot flight, z \cdot park >\}, \quad \forall z \in Assignments\}$$

Finally, we redefine the binary variables as:

Table 1 Feasible assignment triplets for two flights

Flight, j	Arrival period, a_j	Departure period, g_j	Eligible parking areas, k where $B_{jk}=1$	Feasible assignment triplets, $z \in Assignments$
3	5	8	2,7	<5, 3, 2>, <6, 3, 2>, <7, 3, 2> <5, 3, 7>, <6, 3, 7>, <7, 3, 7>
4	12	15	1, 3, 5	<12, 4, 1>, <13, 4, 1>, <14, 4, 1> <12, 4, 3>, <13, 4, 3>, <14, 4, 3> <12, 4, 5>, <13, 4, 5>, <14, 4, 5>

Table 2 Revised feasible assignment triplets for two flights

Flight, j	Arrival period, a_j	Departure period, g_j	Eligible parking areas, k where $B_{jk}=1$	Feasible assignment triplets, $z \in Assignments$
3	5	8	2, 7	<5, 3, 2>, <5, 3, 7>
4	12	15	1, 3, 5	<12, 4, 1>, <12, 4, 3>, <12, 4, 5>

$$x_z = \begin{cases} 1 \ \textit{If flight } z \cdot \textit{flight is assigned to park } z \cdot \textit{park at its arrival}, z \cdot \textit{period} \\ 0 \ \text{o/w} \end{cases}$$

Based on the redefined sets and variables, the assignment based IP model is formulated as in the following:

$$Maximize \sum_{z \in Assignments} W_{z \cdot flight, z \cdot park} w_{z \cdot park} x_z \tag{17}$$

subject to

$$\sum_{z \in Assignments | (z \cdot flight = j)} x_z = 1 \quad \forall j \in U \tag{18}$$

$$\sum_{z \in Assignments | (z \cdot flight \neq j) \wedge (z.period = i) \wedge (z.park = k)} x_z \leq |U|(1 - x_{<a_j, j, k>}) \quad \forall i \in S, \forall j \in U, \forall k \in N \Big| < j, k > \ in \ FlightParkPairs \wedge (a_j \leq i < g_j) \tag{19}$$

$$\sum_{z' \in Assignments | (z' \cdot flight \neq z \cdot flight) \wedge (i = z' \cdot period) \wedge (f_{z' \cdot flight} \neq f_{z \cdot flight}) \wedge (L_{z' \cdot park, z \cdot park} = 1)} x_{z'} \leq |U|(1 - x_z) \quad \forall i \in S, \quad \forall z \in Assignments \Big| a_{z \cdot flight} \leq i < g_{z \cdot flight} \tag{20}$$

and constraints (13) and (16).

Constraints (17), (18) (19), (20), are reformulation of constraints (11), (12), (14), and (15) respectively. Constraints (13) and (16) given in the Timetabling model are used as given in the timetabling based model just with redefined *Assignments* set. Note that the cardinality of the set of flights,$|U|$ is used as big-M in linearization of constraints (19) and (20).

2.3 Scheduling based constraint programming model

Although IP models developed in the previous sections capture the problem with all aspects, they suffer from scalability issues. However, constraint programming (CP) is a powerful alternative to model and solve complex problems such as over-constrained flight-gate assignment problem. Reader may refer to Ornek and Öztürk (2016) for detailed comparison and application of CP and IP methods in planning and scheduling.

In this section, due to its time dependent nature, we develop a scheduling based CP model for the problem. Since the formulation of the CP models is basically related to the software system used, we represent our variables and constraints with CP Optimizer (IBM 2017) constructs that are used to implement and solve the problem.

Flights in this problem are defined as intervals α_j, $j \in U$. Each interval α_j is associated with three variables as $endOf(\alpha_j)$, $startOf(\alpha_j)$ and $sizeOf(\alpha_j)$ which represents the end, start and duration of stay in the airport for each flight respectively. All these variables are ranging in $[a_j, g_j)$. As described in (IBM, 2017), an important property of interval variables is that they can be designed as optional which means that they might not occur in the solution and may not be considered. Since each eligible parking area is an alternative for each flight, α_j interval variables may be performed in any of them. Therefore, we also define optional interval variables β_h for each feasible flight park assignment pair $h \in FlightParkPairs$ defined for the Assignment based IP model formulation in the previous section.

Since each flight interval has to be processed on one of its eligible parking areas, $k \in N$, parking areas are defined as disjunctive resources γ_k which can be used by at most one flight at a time. Each sequence γ_k is a collection of all corresponding optional flight intervals. More formally, $\gamma_k = \{B_h, h \in FlightParkPairs | h.park = k\}$. Note that disjunctive resources such as parking areas in this paper are defined as "*sequence*" in CP Optimizer (IBM 2017).

In addition to the interval and sequence variables, we define a final integer variable $park_j \in N$ which indicates the parking area of flight j assigned to. Finally, we define additional sets and indices used in the formulation of CP model and the search strategy as follows:

$parkOfFlight_j = \{k| < j, k > \in FlightParkPairs\}$: the set of all eligible parking areas for flight j.
$forbiddenParkOfFlight_j = N - parkOfFlight_j$: the set of non-eligible parking areas for flight j.

$averageUtilityOfFlight_j = (\sum_{k \in N} W_{jk}) / \left| parkOfFlight_j \right|$: the average utility of each flight.

Before formulating our problem, we introduce global constraints that are provided by CP Optimizer and used in this paper. See (IBM 2017) for further details.

alternative(*interval*(α), *setofoptionalintervals*(β)): This global constraint models that if interval α present, then exactly one of intervals in the interval β.

allowedAssignments(*setofintegervalues*, *setofintegervariables*): It is used to define the allowed combinations of values for several integer decision variables.

forbiddenAssignments(*setofintegervalues*, *setofintegervariables*): Similarly, this global constraint is used to define the forbidden combinations of values for several integer decision variables.

noOverlap(*sequence*): This global constraint is used to formulate disjunctive nature of sequences and prevents overlapping of the intervals in the sequence.

presenceOf(optional interval): This global constraint is used to capture existence of an optional interval.

forbidStart(*intervalvariable*,*forbiddeninterval*): This global constraint ensure that a given interval variable cannot be started during the forbidden interval.

The CP model of the problem is formulated as in the following:

$$maximize \quad \sum_{j \, in \, U} W_{j,park_j} w_{park_j} \tag{21}$$

subject to

$$allowedAssignments(parkFlight_j park_j) \quad \forall j \in U \tag{22}$$

$$forbiddenAssignments(forbiddenParkOfFlight_j, park_j) \quad \forall j \in U \tag{23}$$

$$noOverlap(\gamma_k) \quad \forall k \in N \tag{24}$$

$$alternative(\alpha_j, all(h \, in \, FlightParkPairs \, : \, h \cdot flight = j)\beta_h) \quad \forall j \in U \tag{25}$$

$$\begin{aligned} &presenceOf(\beta_{<j,k>}) \wedge presenceOf(\beta_{<b,r>}) \Rightarrow \\ &(endOf(\alpha_j) \leq startOf(\alpha_b)) \vee \\ &(endOf(\alpha_b) \leq startOf(\alpha_j)) \\ &\forall < j,k >, < b,r > \; in \, FlightParkPairs | (j < b) \wedge (f_j \neq f_b) \\ &\wedge (L_{kr} = 1) \wedge [(a_j \leq a_b \wedge g_b > a_j) \vee (a_b \geq a_j \wedge g_j > a_b)] \end{aligned} \tag{26}$$

$$forbidStart\left(\beta_h, \left[0, t_k\right)\right) \quad \forall h \in FlighParkPairs \left| h \cdot park \in D \wedge a_{h \cdot flight} \leq t_k \right. \tag{27}$$

Objective function (21) is a reformulated version of the total utility given in (11) and (17). Note that we use variable indexing feature of CP formulation, which enhances the problem representation where $park_j$ returns the index of the parking area used by flight j. Constraints (22) and (23) define and prune domains of parking area variables for each flight respectively. Constraints (24) prevent overlapping of flight interval variables for each parking area k. Recall that γ_k is the collection of all optional flight interval variables that can be assigned to parking area k. Constraints (25) map each flight interval variable to all optional interval variables and guarantee that for each flight interval α_j, exactly one of the alternative β_h intervals is selected. Constraints (26) ensure that any two optional interval flight variables cannot be served in adjacent parking areas at the same time if they belong to the different ground services. If they are assigned to adjacent parking areas, one has to be finished before starting the other. Finally constraints (27) prevent starting of any optional interval variable before the assigned parking area becomes available.

A well-designed and customized search strategy greatly affects the performance of a CP model (Dechter 2003). Hence, we employ a two-phase search strategy for variable (i.e., ordering the variables to branch on in the search tree) and value ordering (the order of value to be instantiated to the selected variable). In the first phase, we start searching with selecting a flight with minimum number of parking areas in its domain (i.e., fail-first). Then we assign the selected flight to an eligible parking area with maximum utility w_k in favor of the objective function. Once the parking variable is instantiated, in the second phase, we select a parking area (i.e., *sequence*) with maximum number of possible flight that can be assigned to and select a flight with maximum average utility $averageUtilityOfFlight_j$. See (IBM ILOG CPLEX Optimization Studio 2017) for details on implementation of search phases.

3 Performance evaluation of the models and a numerical example

In this section, we provide results of implementing developed models by a real case and three more synthetically generated instances. Before presenting the numerical results, in the following table, we compare the size of the proposed models in terms of theoretical bounds in the number of variables and constraints.

As shown in Table 3, while the number of variables depends on the duration of a stay of a flight in Timetabling based IP model, it is independent from this parameter in Assignment based IP model and Scheduling based CP model. Note that the number of variables in the developed CP model is bounded with $|U||N| + |U| + |N|$ of which $|U||N|$ are optional interval variables (β_h), $|N|$ are sequence variables (γ_k) and $|U|$ are parking area assignment variables ($park_j$). The developed models are tested using a realistic size instance provided by a main airline operator in Turkey with 35 parking areas, 19 of which are bridge-equipped, 105 flights, and four different ground service companies. Models have been run on a Windows 7 64 bit machine with 8 GB RAM and Intel i7 processor. IBM CPLEX 12.6 has been used to implement and solve the proposed formulations, which includes IBM CPLEX and IBM CP optimizer to solve integer and constraint programming models respectively. Performances of the developed models are summarized in Table 4.

As seen in Table 4, both Timetabling and Assignment based IP models are able to find the optimal solution in very short time. Although theoretical bounds for the Timetabling and Assignment models are the same as reported in Table 3, instance based reported number of variables and constraints in Table 4 are different. Because; solver's pre-solver tries to reduce the size of the problem by making inferences about the nature of the problem and as both Timetabling and Assignment based models are formulated differently, CPLEX presolve aggregator eliminates different number of columns and rows and results different number of variables and constraints (https://www.ibm.com/support/knowledgecenter/SSSA5P_12.8.0/ilog.odms.cplex.help/CPLEX/UsrMan/topics/cont_optim/simplex/15_preprocess.html).

Although it has the power of expressing complicated constraints efficiently, Scheduling based CP model has lack of ability to use LP relaxation in the search tree (Öztürk et al. 2013) and therefore, it couldn't improve the best solution found in 30 min within 4 h. Another reason that CP model is not able to find the optimal solution is the number of symmetric solutions which returns the same objective function value with different value assignment to variables. Table 4 obviously shows this fact with almost one magnitude more number of solutions found in Scheduling based CP model.

Furthermore, to evaluate performance of aforementioned models in different settings we generated three more synthetic instances (n = 1, 2, 3) by changing duration of stay for each flight by an amount of between $\pm K^n\%$ where $K^n \in Z$ varies between 3, 5 and 10 percent for each of three instances respectively. In short, random duration of stays has been generated for each flight in each instance as in the following formula.

$$g_j^n = a_j + \left(g_j - a_j\right)\left[1 + uniform(-K^n, K^n)/100\right]$$
$$\forall j \in U, \forall n \in \{1,2,3\}, K^1 = 3, K^2 = 5, K^3 = 10$$

The results of the additional experiments are provided in the following tables.

Although both theoretical and instance based results show that there are more variables and constraints in Timetabling based IP formulation, it was expected to have tighter bounds in Timetabling based IP model since it might lead a stronger LP-relaxation due to not relying on big-M based constraints. However, in practice, assignment based model creates a smaller branch and bound tree to be traversed as it requires at least one order of magnitude less number of variables and constraints. This is why Assignment based IP model reaches optimal solution much faster than the Timetabling based IP model. It is also interesting to note that even though we

Table 3 Theoretical bounds for the size of the models

Formulation	Model (approach)	Number of variables	Number of constraints
Integer programming	Timetabling	$\sum_{j \in U}\left(g_j - a_j\right)\lvert N\rvert$	$(\lvert U\rvert\lvert N\rvert)(\lvert U\rvert\lvert N\rvert - 1)/2$
	Assignment	$\lvert U\rvert\lvert N\rvert$	$(\lvert U\rvert\lvert N\rvert)(\lvert U\rvert\lvert N\rvert - 1)/2$
Constraint programming	Scheduling	$\lvert U\rvert\lvert N\rvert + \lvert U\rvert + \lvert N\rvert$	$(\lvert U\rvert\lvert N\rvert)(\lvert U\rvert\lvert N\rvert - 1)/2$

Table 4 Performance evaluation of the developed models on a real size instance where * indicates optimal solution

Model	Number of variables	Number of constraints	Number of nodes	Number of iterations	Number of solutions	Number of branches	Number of fails	Objective	Time (s)
Timetabling	96,025	649,207	16,859	478,203	9	NA	NA	8832	164*
Assignment	3467	202,340	31,245	1,144,972	8	NA	NA	8832	111*
Scheduling	3712	225,056	N/A	N/A	80	5,132,556	2,514,072	8803	1800

introduce 3% and 5% average deviation from the initial duration of stay in the real data in first two synthetic instances, the models are able to assign flights to the same utility valued gates and hence returns the same objective function value.

Although its lack of finding optimal solution, proposed constraint programming method gives a strong baseline for simply adding new business rules emerging from airports' operational needs as constraints. Furthermore, for larger size instances, constraint programming model serves as a quick solution finder for a heuristic framework like large neighborhood search. Indeed, results presented in Tables 4, 5, 6 and 7 shows that CP model is able to traverse wider solution space with large number of solutions found which indicates it can be used as a good constructive solution finder in such heuristic frameworks to escape from local optima. It is noteworthy that as in the initial experiments with real data; we limit CP experiments with synthetic data with 30 min.

Instance data and detailed results are provided in "Appendix 1".

4 Conclusions and further research

With the increase in the intensity of air-traffic in recent years, the management of airport gates has become more important and complicated. This is simply because improper assignment of gates to incoming and outgoing flights may result in flight delays, customer dissatisfaction, and increase in operational costs. As a result, many studies have been undertaken to efficiently use these resources.

In this paper, we propose two IP models and a CP model to solve a highly complicated and over-constrained fight-gate assignment problem to optimality. IP models are able to find the optimal solution in about 100 s, whereas CP model terminates after 1800 s with a near optimal solution. However, this does not imply that IP models are superior to CP model, but points out that CP model is to be revised in terms of search methods. Nevertheless, CP model in this study is one of the pioneering attempts to tackle these types of problems in literature.

For further research directions, two studies are in progress. The first one is to hybridize IP and CP utilizing their powerful properties and the second one is to develop a heuristic algorithm for companies that do not own commercial optimization solvers. Hybridizing a constraint programming based multi-dimensional placement model formulation with large neighborhood search meta-heuristic seems to be a promising and challenging area of interest, too. Application of those hybrid methods to different combinatorial problems arising in airports, such as airline boarding problem (Soolaki et al. 2012) is also a challenging research direction. Evaluating different search heuristics on different CP formulations of the problem could be an important research direction too, as well as symmetry breaking constraints for traversing the search tree faster. Furthermore, incorporating multiple stake holders (airlines, airport authority, ground handlers etc.) into the optimization framework within a distributed manner using distributed constraint programming methods (Rolf and Kuchcinski 2011) is a quite promising research direction for holistic improvement of airport operations as well as multi-objective nature of the problem under full or limited information sharing conditions.

Table 5 Performance evaluation of the developed models on randomly generated instance ($n=1$, $K^1=3$) where * indicates optimal solution

Model	Number of variables	Number of constraints	Number of nodes	Number of iterations	Number of solutions	Number of branches	Number of fails	Objective	Time (s)
Timetabling	95,737	640,093	36,064	1,145,666	9	N/A	N/A	8832	244*
Assignment	3467	201,904	27,516	1,009,900	7	N/A	N/A	8832	104*
Scheduling	3712	224,622	N/A	N/A	68	5,311,738	2,606,256	8772	1800

Table 6 Performance evaluation of developed models on randomly generated instance ($n=2$, $K^2=5$) where * indicates optimal solution

Model	Number of variables	Number of constraints	Number of nodes	Number of iterations	Number of solutions	Number of branches	Number of fails	Objective	Time (s)
Timetabling	96,361	640,647	10,683	720,114	12	N/A	N/A	8832	197*
Assignment	3467	203,150	14,140	718,942	7	N/A	N/A	8832	106*
Scheduling	3712	226,358	N/A	N/A	71	5,745,897	2,804,306	8780	1800

Table 7 Performance evaluation of the developed models on randomly generated instance ($n = 3$, $K^3 = 10$) where * indicates optimal solution

Model	Number of variables	Number of constraints	Number of nodes	Number of iterations	Number of solutions	Number of branches	Number of fails	Objective	Time (s)
Timetabling	96,404	639,124	20,596	880,774	11	N/A	N/A	8825	203*
Assignment	3467	203,343	10,462	627,579	14	N/A	N/A	8825	91*
Scheduling	3712	226,372	N/A	N/A	72	5,263,787	2,575,134	8751	1800

Appendix 1: numerical results

See Tables 8 and 9.

Table 8 The arrival, original departure and departure with extra periods of the flights and the ground service data of flights

Flight no.	Landing period	Original departure period	Departure period with extra 15 min	Ground service no.
1	240	288	291	1
2	153	163	166	3
3	177	187	190	2
4	147	159	162	2
5	76	86	89	2
6	158	170	173	2
7	64	69	72	2
8	110	115	118	3
9	148	153	156	3
10	170	175	178	2
11	68	106	109	1
12	24	29	32	2
13	206	211	214	2
14	187	192	195	3
15	36	72	75	1
16	1	18	21	1
17	155	168	171	3
18	203	288	291	2
19	83	95	98	1
20	88	93	96	2
21	107	158	161	2
22	180	240	243	2
23	90	132	135	4
24	57	69	72	2
25	103	110	113	2
26	145	152	155	2
27	226	288	291	2
28	49	54	57	2
29	235	288	291	2
30	52	57	60	2
31	109	117	120	3
32	161	166	169	2
33	170	203	206	3
34	210	288	291	2

Table 8 (continued)

Flight no.	Landing period	Original departure period	Departure period with extra 15 min	Ground service no.
35	56	68	71	2
36	99	288	291	2
37	44	49	52	2
38	103	111	114	2
39	146	153	156	2
40	186	203	206	3
41	238	288	291	2
42	31	42	45	3
43	48	57	60	3
44	193	288	291	1
45	162	172	175	3
46	55	66	69	3
47	70	79	82	3
48	103	114	117	3
49	205	288	291	3
50	211	288	291	3
51	139	150	153	3
52	171	180	183	3
53	31	40	43	3
54	127	136	139	1
55	115	126	129	1
56	43	54	57	3
57	175	186	189	3
58	244	288	291	3
59	67	78	81	3
60	103	162	165	3
61	253	288	291	3
62	199	207	210	3
63	79	90	93	3
64	187	198	201	3
65	27	36	39	1
66	194	204	207	3
67	130	140	143	3
68	118	134	137	3
69	72	82	85	1
70	141	147	150	2
71	35	41	44	3
72	52	58	61	3
73	97	103	106	3
74	232	288	291	3
75	119	125	128	3

Table 8 (continued)

Flight no.	Landing period	Original departure period	Departure period with extra 15 min	Ground service no.
76	186	192	195	2
77	109	118	121	2
78	209	288	291	3
79	55	61	64	3
80	121	136	139	3
81	219	288	291	3
82	228	288	291	3
83	9	30	33	3
84	114	123	126	3
85	51	74	77	3
86	158	176	179	3
87	222	288	291	3
88	60	85	88	3
89	154	175	178	1
90	212	219	222	3
91	60	72	75	3
92	163	176	179	3
93	230	288	291	3
94	60	72	75	3
95	151	167	170	1
96	199	209	212	1
97	59	75	78	3
98	157	177	180	2
99	210	288	291	3
100	7	30	33	3
101	62	73	76	3
102	161	178	181	3
103	220	288	291	3
104	129	138	141	1
105	47	62	65	3

Table 9 The optimal assignment of flights on each gate for the Timetabling based integer programming model

Gate number	Flight number	Arrival time	Departure time
1	–	–	–
2	–	–	–
3	–	–	–
4	93	230	291
5	23	90	135
	10	170	178
	87	222	291
6	34	210	291
7	36	99	291
8	–	–	–
9	18	203	291
10	–	–	–
11	–	–	–
12	–	–	–
13	–	–	–
14	–	–	–
15	–	–	–
16	–	–	–
17	–	–	–
18	–	–	–
19	16	1	21
	105	47	65
20	97	59	78
	86	158	179
	103	220	291
21	88	60	88
	78	209	291
22	35	56	71
	82	228	291
23	81	219	291
24	85	51	77
	77	109	121
	104	129	141
	2	153	166
	33	170	206
	99	210	291

Table 9 (continued)

Gate number	Flight number	Arrival time	Departure time
25	83	9	33
	56	43	57
	94	60	75
	5	76	89
	80	121	139
	17	155	171
	3	177	190
	58	244	291
26	28	49	57
	101	62	76
	60	103	165
	96	199	211
	90	212	222
	1	240	291
27	91	60	75
	21	107	161
	44	193	291
28	65	27	39
	37	44	52
	79	55	64
	11	68	109
	8	110	118
	26	145	155
	102	161	181
	40	186	205
	13	206	214
	41	238	291
29	12	24	32
	30	52	60
	7	64	72
	69	73	85
	31	109	120
	67	130	143
	39	146	156
	6	158	173
	14	187	195
	27	226	291

Table 9 (continued)

Gate number	Flight number	Arrival time	Departure time
30	24	57	72
	20	88	96
	25	103	113
	68	118	137
	9	148	156
	32	161	169
	66	194	207
	29	235	291
31	38	103	114
	84	115	126
	98	157	180
32	42	31	45
	46	55	69
	47	70	82
	55	115	129
	89	154	178
	76	186	195
	49	205	291
33	53	31	43
	43	48	60
	59	67	81
	73	97	106
	70	141	150
	95	151	170
	52	171	183
	64	187	201
	74	232	291
34	100	7	33
	71	35	44
	72	52	61
	63	79	93
	75	119	128
	51	139	153
	45	162	175
	57	176	189
	62	199	210
	50	211	291

Table 9 (continued)

Gate number	Flight number	Arrival time	Departure time
35	15	36	75
	19	83	98
	48	103	117
	54	127	139
	4	147	162
	92	163	179
	22	180	243
	61	253	291

References

Ballis A (2002) Simulation of airport terminal facilities in the Greek airports of Kavala and Alexandroupolis. Oper Res Int Journal 2(3):391–406

Bihr RA (1990) A conceptual solution to the aircraft gate assignment problem using 0,1 linear programming. Comput Ind Eng 19(1–4):280–284

Bolat A (1999) Assigning arriving flights at an airport to the available gates. J Oper Res Soc 50(1):23–34

Bouras A, Ghaleb MA, Suryahatmaja US, Salem AM (2014) The airport gate assignment problem: a survey. Sci World J 923859:27

Cheng C, Ho SC, Kwan C (2012) The use of meta-heuristics for airport gate assignment. Expert Syst Appl 39:12430–12437

de Man S (2011) Flight to gate assignment: solution methods and complexities at planning horizons. Aerlines 53:1–5

Dechter R (2003) Constraint Processing. Morgan Kaufmann, Burlington

Deng W, Zhao H, Yang X, Xiong J, Suna M, Lia B (2017) Study on an improved adaptive PSO algorithm for solving multi-objective gate assignment. Appl Soft Comput 59:288–302

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

Ding H, Lim A, Rodrigues B, Zhu Y (2005) The over-constrained airport gate assignment problem. Comput Oper Res 32:186–1880

Drexl A, Nikulin Y (2008) Multicriteria airport gate assignment and Pareto simulated annealing. IIE Trans 40(4):385–397

IBM ILOG CPLEX Optimization Studio (2017). https://www.ibm.com/analytics/cplex-optimizer. Accessed 18 Dec 2017

Lee J, Im H, Kim KH, Xi S, Lee C (2016) Airport gate assignment for improving terminals' internal gate efficiency. Int J Ind EngTheory Appl Pract 23(6):431–444

Li C (2009) Airport gate assignment. A hybrid model and implementation, http://arxiv.org/abs/0903.2528

Li M (2010) Optimized assignment of civil airport gate. In: Proceedings of the ınternational conference on ıntelligent system design and engineering application (ISDEA'10), Changsha, China, pp 33–38

Lim A, Rodrigues B, Zhu Y (2005) Airport gate scheduling with time windows. Artif Intell Rev 24(1):5–31

Ornek MA, Öztürk C (2016) Optimisation and constraint based heuristic methods for advanced planning and scheduling systems. Int J Ind EngTheory Appl Pract 23(1):26–48

Ornek MA, Öztürk C, Sugut I (2019) Model-based heuristic for counter assignment problem with operational constraints: a case study. J Air Transp Manag 77:57–64

Öztürk C, Tunalı S, Hnich B et al (2013) Balancing and scheduling of flexible mixed model assembly lines. Constraints 18:434–469. https://doi.org/10.1007/s10601-013-9142-6

Öztürk C, Sargut FZ, Örnek MA, Eliiyi DT (2017) Optimisation and heuristic approaches for assigning inbound containers to outbound carriers. Maritime Policy Manag 44(7):825–836

Rolf CC, Kuchcinski K (2011) Distributed constraint programming with agents. In: Bouchachia A (ed) Adaptive and intelligent systems. ICAIS 2011. Lecture notes in computer science, vol 6943. Springer, Berlin

Saaty TL (1977) A Scaling Method for Priorities in Hierarchical Structures. J Math Psychol 15:234–281

Soolaki M, Mahdavi I, Mahdavi-Amiri N, Hassanzadeh R, Agjajani A (2012) A new linear programming approach and genetic algorithm for solving airline boarding problem. Appl Math Model 36:4060–4072

Süğüt I (2016) Optimization approaches for specific airport operations", unpublished MSc thesis, Izmir University of Economics, Industrial Engineering Department

Yan S, Tang CH (2007) A heuristic approach for airport gate assignments for stochastic flight delays. Eur J Oper Res 180(2):547–567