Learning-Driven Feasible and Infeasible Tabu Search for Airport Gate Assignment

Mingjie Li[a,b], Jin-Kao Hao[b] and Qinghua Wu[a,*]

[a]*School of Management, Huazhong University of Science and Technology, No. 1037, Luoyu Road, Wuhan, China*

[b]*LERIA, Université d'Angers, 2 bd Lavoisier, 49045 Angers Cedex 01, France*

Abstract

The gate assignment problem (GAP) is an important task in airport management. This study investigates an original probability learning based heuristic algorithm for solving the problem. The proposed algorithm relies on a mixed search strategy exploring both feasible and infeasible solutions with the tabu search method and employs a reinforcement learning mechanism to guide the search toward new promising regions. The algorithm is compared with several reference algorithms on three sets of real-world benchmark instances in the literature. Computational results show the high competitiveness of the algorithm in terms of solution quality and computation time. Especially, it reports improved best solutions (new upper bounds) for all the 180 tested real-world benchmark instances in the literature. The key components of the algorithm are analyzed. The code of the algorithm will be publicly available.

Keywords: Heuristics; gate assignment; probability learning; feasible and infeasible tabu search.

1 Introduction

The rapid development of the international transportation market has significantly increased the volume of air traffic over the past decades. According to the prediction of the International Air Transport Association (IATA), the number of air passengers is anticipated to be over 7.2 billion in 2035, which

* Corresponding author.
Email addresses: `lmj@hust.edu.cn` (Mingjie Li), `jin-kao.hao@univ-angers.fr` (Jin-Kao Hao), `qinghuawu1005@gmail.com` (Qinghua Wu).

nearly doubles the number of 3.8 billion passengers in 2016 [32]. The considerable rise in air traffic leads to air traffic congestion on the airport surface, which increases airborne delays and air pollutants. Although the expansion of airport capacity is the most obvious solution for air traffic congestion, it is unrealistic in some cases due to space limitations. A more practical method is to utilize highly efficient decision support systems based on sophisticated optimization methodologies to manage existing resources effectively. Airport gates are core resources and important operating facilities of airports because they provide various services, such as passengers boarding and disembarking, aircraft parking, baggage loading, and aviation transferring. Therefore, optimizing the use of airport gates becomes a key issue in the daily management of airports.

The task of the airport gate assignment problem (GAP), as an important decision problem that airport professionals must face every day, is to assign each aircraft to an available gate to meet operational requirements while maximizing the convenience for passengers and airport operation efficiency [4, 14, 33]. Gate assignment is a complex airline planning problem due to various stakeholders involved and multiple objectives considered, some of which are even in conflict. By focusing on different perspectives, the objectives of GAP can be mainly classified into two classes: the airline/airport- and passenger-oriented objectives. For the airline/airport-oriented objectives, the goals mainly include maximizing the utilization of the available gates and terminal [8, 9, 25], minimizing aircraft towing activities (cost) [5, 19, 22, 23, 36, 44, 47, 61], maximizing the total gate preferences [5, 22–24, 36, 47], minimizing the number of un-gated aircraft [16, 21–24], minimizing the flights delay [13, 37, 54], and minimizing the deviations in the schedules [5, 9, 44, 50]. Meanwhile, the passenger-oriented objectives mainly involve minimizing the total passenger walking distance to improve customer satisfaction [2, 12, 13, 21, 29, 40, 57]. The reason is that metropolitan airports are usually very large; thus, traveling within the airport may take a significant amount of time and effort for a passenger [1, 33]. Other studies aim to minimize alternative objectives, such as the passenger waiting time [58, 59], the passenger transit time [34, 35], and the baggage transferring distance [5].

GAP is an NP-hard problem because it can be reduced to the classic quadratic assignment problem [48]. Considering its relevance and complexity, several solution methods have been proposed to solve the problem in the literature, which can be mainly classified into exact approaches and heuristic/metaheuristic methods. Exact approaches have the advantage of provable optimal guarantee to the found solutions but generally need a computation time that grows exponentially with the problem size. Nevertheless, many efforts have been made to develop powerful exact approaches for GAP. Earlier exact approaches are generally based on the branch and bound (B&B) method and its variants [2, 8, 9]. Furthermore, various integer programming and mixed integer programming (MIP) formulations have been studied and solved either by standard solvers or dedicated exact algorithms [6, 28, 39, 40, 58]. Very recently, Karsu et al. [33] proposed a GAP model aiming to minimize the total walking distance of all passengers subject to the minimum number of aircraft assigned to the apron. They proposed a B&B algorithm, a beam search (BS), and a filtered beam search (FBS) that showed excellent performances for instances with up to 200 aircraft and 38 gates.

Meanwhile, exact approaches may require a prohibitive time in practice due to the intrinsic difficulty of the GAP problem. Therefore, in the daily operation of large airports, heuristic/metaheuristic algorithms are typically employed to produce practical solutions in a given short time. For example, Xu et al. [57] proposed a simple tabu search (TS) algorithm to solve the model with the objective of minimizing the overall connection times of passengers. Ding et al. [20, 21] introduced a GAP model to minimize the number of ungated flights and total walking distances and designed a hybrid method combining simulated annealing (SA) and TS. Cheng et al. [12, 13] presented several algorithms to minimize the total walking distance of all passengers, including genetic algorithm, TS, SA, a hybrid approach based on TS and SA, and TS with path relinking algorithm. This work also presented an assessment of the proposed algorithms for the first time using a set of realistic flight data. Yu et al. [61] designed an adaptive large neighborhood search algorithm to solve a multi-objective GAP, including the aircraft tow cost, transfer passenger cost, and the robustness. Yu et al. [60] developed a variable reduce neighborhood search algorithm based on an MIP model for robust GAP where traditional approximate models cannot precisely consider the traveling distance of transfer passengers. Benlic et al. [5] proposed a breakout local search algorithm to optimize a linear combination of nine gate allocation objectives on data provided by Manchester Airport.

In recent years, as swarm intelligence algorithms demonstrate excellent performances in solving nonlinear problems, black-box model problems, multi-dimensional, and multi-objective optimization problems, they have gradually become popular for solving GAP. For example, Deng et al. [16,18] considered a GAP model which minimizes the total walking distances of passengers, the idle time variance of each gate, the number of flights at the apron, and the most reasonable utilization of large gates. They proposed several algorithms based on particle swarm optimization (PSO) to solve the proposed model, including an improved adaptive PSO algorithm with alpha-stable distribution and dynamic fractional calculus and an improved quantum evolutionary algorithm based on the niche co-evolution strategy and enhanced PSO. Later, Deng et al. [17] presented an improved ant colony optimization algorithm for the GAP, where the initial problem is divided into several subproblems, and the ants in the population are divided into elite and common ants to improve the convergence rate and avoid local optimum traps. Experiments on the instances from

Guangzhou Baiyun International Airport shows that their algorithm effectively solves the problem. Marinelli and Dell'Orco with their colleagues [15,41–43] proposed several methods based on bee colony optimization (BCO) for a GAP model considering two main objectives, which minimizes the passenger total walking distance and the remote gate usage. They evaluated their methods on the instance from Milano-Malpensa International Airport with up to 178 flights and 65 gates. Their experimental results demonstrated the excellent performance of these BCO-based methods for solving their problem. For a comprehensive survey of GAP solution methods, the interested readers are referred to [10, 14, 51].

In this study, we follow the majority of the works on GAP in the literature and focus on the passenger-oriented objectives of the problem, which consists of assigning each arriving and departing aircraft to a gate such that the total walking distance of passengers is minimized while ensuring that two aircraft with overlapping time are not allocated to the same gate. Similar to many other studies [2, 12, 13, 21, 29, 40, 57], we consider three types of passengers, namely, arriving passengers, departing passengers and transferring passengers; and three types of walking distances, namely, from a gate to the arrival hall, from a gate to the departure hall, and from one gate to another.

The presented work can be summarized as follows:

- We introduce an original probability learning based feasible and infeasible tabu search (PLFITS) algorithm. The algorithm integrates two important features. First, it employs a probability learning mechanism that adopts the concept of reinforcement learning to maintain and update a set of probability vectors, each probability vector specifying the probability of assigning an aircraft to a particular gate. The probability learning mechanism gathers useful information from visited local optima, which is then advantageously used during the subsequent search process to guide the algorithm toward new promising regions. Second, the algorithm uses a powerful tabu search procedure for local improvement, which relies on a mixed search strategy exploring both feasible and infeasible search spaces due to the following consideration. By relaxing the gate conflict constraint in a controlled manner, the algorithm can go through feasible and infeasible search spaces to locate high-quality solutions that are otherwise difficult to reach. Especially, to prohibit the search from going too far away from the boundary of the feasible search space, we devise an adaptive penalty-based fitness function that is applied to guide the local search process for a fruitful examination of candidate solutions.
- To assess the performance of the proposed algorithm on the GAP models, we use the flight data collected from Incheon International Airport (ICN) by Cheng et al. [12], and two other sets of instances generated by Karsu et al. [33] from Ankara Esenboğa Airport (ESB) and Istanbul Atatürk Air-

port (ISL). Extensive computational results on the three sets of real-world benchmark instances show a highly competitive performance of PLFITS compared with the reference metaheuristic algorithms for GAP. In particular, PLFITS reports improved best-known solutions (new upper bounds) for all the 180 real-world instances with improvement ranging from 0.47% up to 11.85%.
- To the best of our knowledge, existing heuristic approaches for GAP restrict their search to feasible solutions only. This study is the first to mix feasible and infeasible searches in the context of solving GAP. We perform computational assessments to verify the effectiveness of this mixed search strategy for finding high-quality solutions. Moreover, the code of our algorithm will be made available online, which can serve as a useful tool for researchers and practitioners in airport planning and scheduling. Finally, given that the proposed underlying search strategies are of general feature, they can be beneficially adapted to related GAP variants where other objectives and constraints are considered.

The rest of the paper is organized as follows. Section 3 provides the general framework of PLFITS and its key algorithmic components. Section 4 presents the computational results and comparisons with the reference algorithms. Section 5 analyzes some significant characteristics of the proposed algorithm before conclusions are shown in Section 6.

2 Problem statement and mathematical formulation

Many GAP formulations and models have been proposed in the literature by considering different objectives and constraints [10, 14]. Following the mainstream research on GAP, we consider GAP as a single objective problem and focus on improving passenger comfort. Precisely, we adopt a GAP definition similar to that proposed in [12,13,57]. The model assigns each aircraft (flight) to an available gate while ensuring that two aircraft with overlapping times are not assigned to the same gate. The objective of the problem is to minimize the total walking distance of the passengers. Three types of passengers are considered in the problem, namely, the departing, arriving, and transfer passengers. Three types of walking distances of the passengers are calculated, namely, the distance between a gate and exit, the distance between a gate and the check-in desks, and the distance between two gates. The above-mentioned model is a basis of many GAP variants and is widely extended in the literature to other GAP formulations including single-objective GAP and multi-objective GAP [15–18,41–43,60,61]. To formulate the problem as a mixed integer nonlinear program, the following notations are used:

Parameters

G: The set of airport gates;
F: The set of aircraft (flights);
a_i: The arrival time of aircraft i;
d_i: The departure time of aircraft i;
c_{0i}: The number of passengers departing from aircraft i;
c_{i0}: The number of passengers arriving on aircraft i;
c_{ij}: The number of passengers transferring from aircraft i to aircraft j;
w_{lm}: The walking distance between gates l and m;
w_{l0}: The walking distance between gate l and the check-in desks;
w_{0l}: The walking distance between gate l and exit;

Variables

x_{il}: A binary decision variable taking the value of 1 if aircraft i is assigned to gate l, and it is 0 otherwise.

On the basis of the abovementioned parameters and decision variables, GAP can be formulated as the following mixed integer nonlinear program:

$$\min \quad Z(S) = \sum_{i\in F}\sum_{j\in F}\sum_{l\in G}\sum_{m\in G} c_{ij}w_{lm}x_{il}x_{jm} + \sum_{i\in F}\sum_{l\in G}(c_{i0}w_{l0} + c_{0i}w_{0l})x_{il} \tag{1}$$

$$\text{s.t.} \quad \sum_{l\in G} x_{il} = 1, \ \forall i \in F \tag{2}$$

$$x_{il}x_{jl}(d_j - a_i)(d_i - a_j) \geq 0, \ \forall i,j \in F, \ \forall l \in G \tag{3}$$

$$x_{i,l} \in \{0,1\}, \ \forall i \in F, \ \forall l \in G \tag{4}$$

where the objective is to minimize the total walking distance for all passengers. This objective function has been widely adopted in several studies, such as [12,13,57]. The quadratic part of the objective function (1) is the total walking distance for all transfer passengers, while the linear part is the total walking distance for the arriving and departing passengers. Constraints (2) ensure that each aircraft is assigned to exactly one gate. Constraints (3) guarantee that two aircraft's time cannot overlap if they are assigned to the same gate. Constraints (4) indicate that the decision variables are binary.

3 The probability learning based feasible and infeasible tabu search algorithm for GAP

In this section, we present our probability learning based feasible and infeasible tabu search for GAP. The proposed PLFITS algorithm integrates several prominent ingredients responsible for its effectiveness, including a probability learning strategy to generate high-quality solutions, and a feasible and infeasible search with popular tabu technique to further improve the quality of solutions.

Algorithm 1: Pseudo-code of PLFITS for GAP

```
Input: An instance, the number of available gates $k$, and the max running time $t_{max}$.
Output: The best solution $S_{best}$ found so far.
1  $O_{best} \leftarrow +\infty$; //Records the objective value of $S_{best}$
2  for $i = 1, 2, ..., n$ do
3      for $j = 1, 2, ..., k$ do
4          $p_{ij} \leftarrow 1/k$;
5  while $Time() < t_{max}$ do
6      $S_{initial} \leftarrow Gate\_Selection(P)$;
7      $S_{target} \leftarrow Feasible\_And\_Infeasible\_Tabu\_Search(S_{initial})$;
8      if $Z(S_{target}) < O_{best}$ then
9          $O_{best} \leftarrow Z(S_{target})$;
10         $S_{best} \leftarrow S_{target}$;
11     $P \leftarrow Probability\_Updating(S_{initial}, S_{target})$;
12     $P \leftarrow Probability\_Smoothing(P)$;
13 return $S_{best}$;
```

3.1 General Scheme

The PLFITS algorithm integrates a feasible and infeasible tabu search procedure [26] within the general learning based framework designed for the well-known graph coloring problem [62]. During the PLFITS search process, problem-specific knowledge is learned via its probability learning procedure. The learned information, recorded in a probability matrix, is in turn applied to guide the algorithm toward promising search regions. As such, PLFITS iteratively explores the given search space by alternating between the probability learning procedure and the feasible and infeasible tabu search procedure to attain a better balance of search diversification and intensification. Its general scheme is summarized in Algorithm 1, which is composed of four main components: a feasible and infeasible tabu search procedure, a gate selection strategy, a probability smoothing technique, and a probability updating rule.

We adopt the probability matrix P of size $n \times k$, where n is the number of aircraft and k is the number of gates (i.e., $n = |F|$, $k = |G|$). Each element p_{ij} ($i \in F$, $j \in G$) indicates the probability that aircraft i is assigned to gate j. Thus, the i-th row in matrix P, denoted by p_i, forms a probability vector indicating the probability of an aircraft $i \in F$ selecting each gate (refer to Fig. 1 as an example). At the beginning of PLFITS, all the elements in the

$$P_{n\times k} = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1k} \\ p_{21} & p_{22} & \cdots & p_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n1} & p_{n2} & \cdots & p_{nk} \end{bmatrix}$$

Fig. 1. Probability Matrix.

probability matrix are set to $\frac{1}{k}$, which means that all the aircraft are assigned to a gate from the set of all available gates with equal probability.

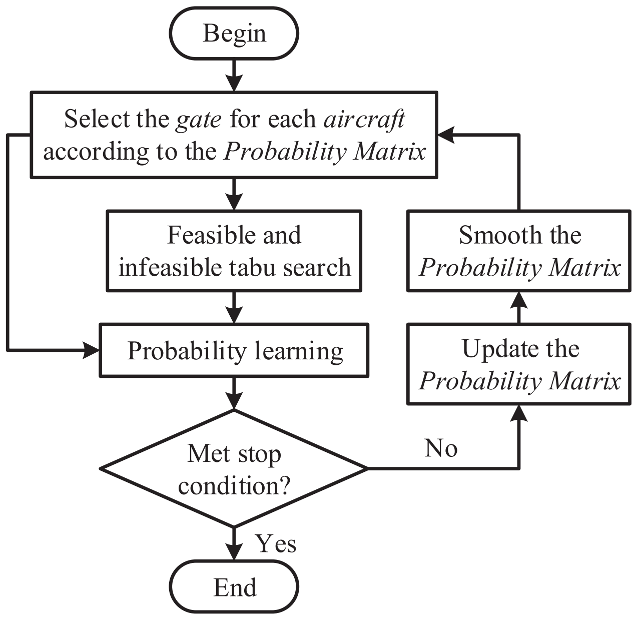

Begin
Select the *gate* for each *aircraft* according to the *Probability Matrix*
Feasible and infeasible tabu search
Probability learning
Met stop condition?
Yes
End
No
Update the *Probability Matrix*
Smooth the *Probability Matrix*

Fig. 2. Schematic of PLFITS for GAP.

As shown in Algorithm 1, PLFITS first initializes the probability matrix P by setting each element in P with equal probability. Then, the algorithm enters its main search procedure in the "while" loop (lines 5–12 in Algorithm 1). Each iteration of PLFITS consists of first assigning aircraft $i \in F$ to a gate $j \in G$ according to a gate selection strategy (Section 3.2) that is based on the probability matrix P. Once all the aircraft are assigned to a gate forming an initial solution $S_{initial}$, this solution is then further improved by the feasible and infeasible tabu search procedure to reach a local optimum denoted as S_{target} (Section 3.3). Moreover, the best found solution S_{best} is updated by the resulting solution S_{target} if S_{target} is better than S_{best} in terms of the objective function (Equation (1)). Thereafter, the probability matrix P is updated by comparing $S_{initial}$ and S_{target} using the probability updating rule (Section 3.4), which rewards, penalizes, or compensates a gate. Finally, a probability smoothing technique (Section 3.5) is applied to smooth each aircraft's probability vector for forgetting some old decisions. The abovementioned process is repeated until a given time limit has been reached, and the best solution S_{best} found is returned as the final result of the PLFITS approach. The schematic of PLFITS for GAP is depicted in Fig. 2. The four main components of the PLFITS algorithm are described in detail in the following subsections.

3.2 Gate selection

At the beginning of each iteration of PLFITS, the gate selection procedure is applied to construct an initial solution $S_{initail}$, which is then submitted to the feasible and infeasible tabu search for further improvement. In the gate selection procedure, each aircraft $i \in F$ needs to select a gate $j \in G$ from the k available gates according to the probability matrix P. Different selection strategies have been proposed in the literature, such as greedy selection, random selection, hybrid selection, and roulette wheel selection [62].

After experimenting with the abovementioned selection strategies, we use the hybrid selection strategy, a combination of greedy selection and random selection. Specifically, it assigns the aircraft one by one. For each considered aircraft $i \in F$, it is assigned to a gate selected at random with a probability ε, while with probability $1 - \varepsilon$, it is assigned to the gate j having the maximum associated probability in p_i (i.e., $j = \text{argmax}_{m \in \{1,\ldots,k\}}\{p_{im}\}$). Such a hybrid scheme is expected to take advantage of the greedy selection rule favoring the gate j having the maximum associated probability and the random selection rule bringing considerable randomness into the search to prevent the algorithm from being too greedy.

Notably, the resulting solution produced by the gate selection procedure is not necessarily a feasible one given that it may violate the gate conflict constraint, which requires that the time for two aircraft assigned to the same gate cannot overlap. Starting from such an initial solution, the feasible and infeasible tabu search is launched to find an improved feasible solution.

3.3 Feasible and infeasible tabu search procedure

Once the gate selection phase obtains an initial starting solution S, the feasible and infeasible tabu search procedure is invoked to further improve S with the aim of finding a high-quality feasible solution. The proposed feasible and infeasible tabu search shares a similar idea to the more general strategic oscillation strategy [27], which permits the visit of the infeasible search space to bring more search freedom. From this point of view, our local search method can be viewed as an oscillation-based tabu search procedure. As shown in several studies on strongly constrained problems [49, 52, 64], visiting intermediary infeasible solutions during the search process can efficiently enhance the performance of neighborhood-based local search because it may facilitate transitions between structurally different feasible solutions. Fig. 3 illustrates such a situation in the case of GAP. As shown in the figure where $\{F_1, F_2, F3\}$ is the set of aircraft waiting for gate assignment, $\{Gate1, Gate2\}$ is the set of gates, and the X-axis indicates the arrival time and the departure time for each aircraft, the solutions S^1 and S^4 in Fig. 3(a) are two feasible solutions, and we assume that the objective value of solution S^4 is better than that of the solution S^1. Owing to the presentation of the gate conflict constraint, considering only feasible solutions can easily make a neighborhood-based local search blocked. The reason is that any move operation in a local search, typically defined as re-assigning an aircraft or swapping two aircraft in different gates, can lead to infeasible solutions. In such cases, constraint relaxation is an attractive strategy. By relaxing the gate conflict constraint and visiting two intermediary infeasible solutions S^2 and S^3 as shown in Fig. 3(b), the high-quality feasible solution S^4 can be easily accessed from the feasible solution S^1.

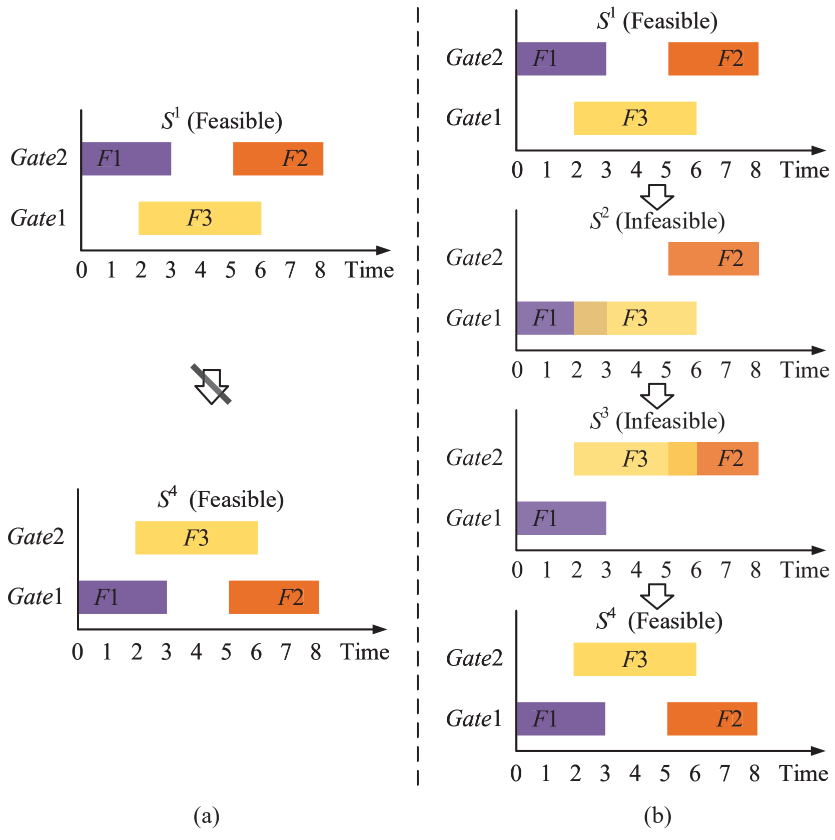


Fig. 3. Example of feasible and infeasible tabu search.

Following the abovementioned idea, the proposed feasible and infeasible tabu search procedure explores an enlarged search space including both feasible and infeasible solutions. To allow the search to oscillate between the feasible and infeasible search spaces effectively, we design an extended evaluation function f that combines the objective function of GAP (the total walking distance, refer to Section 2) with a penalty function associated with the degree of solution infeasibility (Section 3.3.1).

Algorithm 2: Pseudo-code of feasible and infeasible tabu search

```
Input: An instance; $S_{initial}$: an initial solution; $\omega$: the search depth, $u_p$: the
       frequency of updating the penalty parameter.
Output: The local optimum feasible solution $S_{target}$ found so far.
1  $\varphi \leftarrow 0$;//Penalty parameter
2  $Iter \leftarrow 0$; //Iteration counter
3  $NI \leftarrow 0$; //Counts the consecutive iterations $S_{target}$ is not improved
4  $S_c \leftarrow S_{initial}$;//Solution of the current iteration
5  $S \leftarrow S_{initial}$;//Solution with the best objective value
6  $O_{target} \leftarrow +\infty$; //Objective value of the feasible solution $S_{target}$
7  while $NI < \omega$ do
8      Choose a best admissible neighboring solution $S' \in N(S_c)$;
9      $S_c \leftarrow S'$;
10     $NI \leftarrow NI + 1$;
11     $Iter \leftarrow Iter + 1$;
12     if $S_c$ is a feasible solution and $f(S_c) < O_{target}$ then
13         $O_{target} \leftarrow f(S_c)$;
14         $S_{target} \leftarrow S_c$;
15         $NI \leftarrow 0$;
16     if $f(S_c) < f(S)$ then
17         $S \leftarrow S_c$;
18         $NI \leftarrow 0$;
19     if $Iter$ mod $u_p = 0$ then
20         Update penalty parameter $\varphi$;
21 return $S_{target}$;
```

The general scheme of the feasible and infeasible tabu search is summarized in Algorithm 2, while its main components are presented in detail in the following sections. Starting from an initial solution $S_{initial}$, which can be feasible or infeasible, each iteration of the tabu search procedure consists of determining an overall best admissible solution (with a maximum extended evaluation function value) from the neighborhood of the current solution S_c produced by the *Re-assigning* move operator. During the search process, the best-found solution S_{target} is updated with S_c if S_c is feasible, and if its quality in terms of the total walking distance is better than that of S_{target}. If an improvement in terms of the objective function is not achieved during ω consecutive iterations, then the best feasible solution S_{target} found during the search is returned as the final output of the feasible and infeasible tabu search procedure.

3.3.1 Search space and penalty-based evaluation function

A candidate solution to GAP is to partition all aircraft into k subsets S_1, S_2, ... , S_k, such that each subset includes all aircraft assigned to the same gate. Then, the search space explored by our proposed algorithm, composed of both feasible and infeasible solutions, is formally defined as follows:

$$\Omega = \{\{S_1, S_2, ..., S_k\} : \bigcup_{i=1}^{k} S_i = F, S_i \cap S_j = \emptyset, i \neq j, 1 \leq i, j \leq k\} \tag{5}$$

Given a candidate solution $S \in \Omega$ in the enlarged search space, its quality is evaluated by an extended evaluation function (fitness) f which is a linear combination of the basic objective function Z considering the walking distance (Equation. (1) in Section 2) with a penalty function considering the degree of solution infeasibility:

$$f(S) = \sum_{i \in F} \sum_{j \in F} c_{ij} w_{r(i)r(j)} + \sum_{i \in F} (c_{i0} w_{r(i)0} + c_{0i} w_{0r(i)}) + \varphi \sum_{i \in F} t_{ir(i)} \tag{6}$$

where $r(i)$ $(i \in F)$ represents the gate assigned to aircraft i, and $t_{ir(i)}$ denotes the number of aircraft which are allocated to the same gate as aircraft i and have overlapping time with i. Thus, $\sum_{i \in F} t_{ir(i)}$ evaluates the total infeasibility degree of the solution S by considering the number of conflicting aircraft. φ is a parameter used to control the relative importance of the penalty function $\sum_{i \in F} t_{ir(i)}$. In general, increasing the value of φ will decrease the attractiveness of infeasible solutions, while decreasing the value of φ makes infeasible solutions more attractive. In our feasible and infeasible tabu search, φ is adaptively adjusted according to the search history. Precisely, φ is increased by setting $\varphi = \varphi + \tau$ if u_p consecutively visited solutions are all feasible ones (u_p is a parameter). Meanwhile, φ is decreased by setting $\varphi = \varphi - \tau$ if u_p consecutively visited solutions are all infeasible ones ($\varphi = 0$, if $\varphi < 0$). The motivation behind this adaptive strategy is to keep a search balance between feasible and infeasible search spaces. If the search is confined too much in the feasible search spaces, then it is encouraged to move into an infeasible search space by reducing φ. Conversely, if too many infeasible solutions are visited, then the search is forced to move back to feasible search spaces by increasing φ.

3.3.2 Move operator and neighborhood

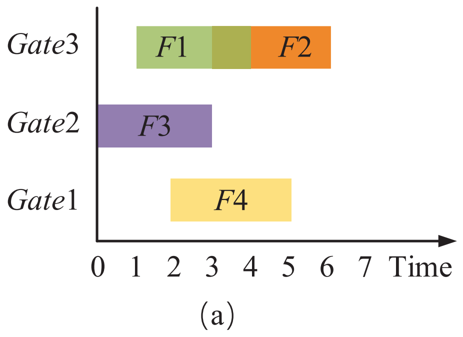
To move from one solution to another in the search space, the feasible and infeasible tabu search procedure employs the *Re-assigning* move, which displaces a single aircraft $a \in A$ from its current gate i to another gate j.

To efficiently calculate the move gain of a *Re-assigning* move denoted by $< s, i, j >$, which indicates the variation in the fitness value f induced by the *Re-assigning* move, we adopt a matrix T of size of $n \times k$ ($n = |F|$, $k = |G|$) where element t_{ij} ($i \in F$, $j \in G$) indicates the number of aircraft which are allocated to gate j and have overlapping time with aircraft i. With this T, the move gain value of a *Re-assigning* move can be computed in $O(n)$ time by

$$\Delta(< s, i, j >) = \varphi(t_{sj} - t_{si}) - c_{ss}(w_{jj} - w_{ii}) + \sum_{u=1}^{n}(c_{su} + c_{us})(w_{r(u)j} - w_{r(u)i}) \quad (7)$$

Notably, the arrival and departure aircraft of some transferring passengers could be the same, and passengers need to get off the arrival aircraft first and then board the same aircraft in this situation. Therefore, we have $c_{ss} \geq 0$ ($s \in F$) and $w_{ii} \geq 0$ ($i \in G$).

After a *Re-assigning* move $< s, i, j >$ is performed, the matrix T is updated in $O(n)$ time by amending the value of the elements affected by the move. Specifically, for each aircraft $v \in F \setminus \{s\}$, if it has overlapping time with s, then the value of the element t_{vj} is increased by one, while the element t_{vi} is decreased by one. An illustrative example for this updating is given in Fig. 4, where the left and middle parts respectively present a solution of GAP and its associated matrix T, and the right part gives the updated matrix T after re-assigning aircraft $F2$ from $Gate3$ to $Gate2$.



$$T_{4\times 3} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

(b)

$$T_{4\times 3} = \begin{bmatrix} 1 & 2 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 2 & 1 \end{bmatrix}$$

(c)

Fig. 4. Illustrative example for move gain matrix update: (a) a given solution, (b) its associated matrix T, and (c) updated matrix T after moving $F2$ from $Gate3$ to $Gate2$.

3.3.3 Tabu list and tabu tenure management

To prevent the search from short-term cycling, each time an aircraft s is moved from its original gate i to another gate j, it is forbidden to move the aircraft s back to gate j for the next tt iterations (tt is a parameter called the tabu tenure) [11,26]. However, the tabu status of a move is overridden if this move leads to a solution of better quality (in terms of f) than the best solution found during the search (aspiration criterion).

3.4 Probability updating

After the feasible and infeasible tabu search procedure is applied to improve the initial solution $S_{initial}$, the probability learning procedure is triggered to update the probability matrix P by comparing the starting solution $S_{initial}$ and the improved solution S_{target} for checking whether an aircraft was moved from its original gate to another gate or if it stayed in the same gate as in $S_{initial}$.

Specifically, for each aircraft $i \in F$, if its gate $l \in G$ is not changed, then we reward its original gate l and update its probability vector p_i as follows (t is the current generation number):

$$p_{ij}(t+1) = \begin{cases} \alpha + (1-\alpha)p_{ij}(t) & j = l \\ (1-\alpha)p_{ij}(t) & \text{otherwise.} \end{cases} \tag{8}$$

where α $(0 < \alpha < 1)$ is a reward factor.

If the aircraft $i \in F$ is moved from its original gate $l \in G$ to another gate $m \in G \setminus \{l\}$ in the improved solution S_{target}, then we penalize its former gate l, compensate the new gate m, and update its probability vector p_i as follows:

$$p_{ij}(t+1) = \begin{cases} (1-\gamma)(1-\beta)p_{ij}(t) & j = l \\ \gamma + (1-\gamma)\dfrac{\beta}{k-1} + (1-\gamma)(1-\beta)p_{ij}(t) & j = m \\ (1-\gamma)\dfrac{\beta}{k-1} + (1-\gamma)(1-\beta)p_{ij}(t) & \text{otherwise.} \end{cases} \tag{9}$$

where β $(0 < \beta < 1)$ is a penalization factor and γ $(0 < \gamma < 1)$ is a compensation factor.

With the help of the learning schemes (8) and (9) applied in each iteration of the algorithm, the probability for an aircraft to select a correct gate is expected to increase gradually. The probability updating scheme is based on the reinforcement learning mechanism proposed by Zhou et al. [62, 63] which is inspired by the learning automata (LA) [46]. The LA is a policy iteration method where the optimal policy is directly determined in the space of candidate policies [53]. An action probability vector is maintained and updated in LA according to a specific probability learning technique or reinforcement scheme. Well-known reinforcement schemes include linear reward-penalty and linear reward-inaction and aim to increase the probability of selecting an action in the event of success and decrease the probability in the case of failure [55].

3.5 Probability smoothing

After the probability learning procedure is applied to update the probability matrix P, we employ a probability smoothing technique to further smooth each aircraft's probability vector due to the following consideration. Old decisions made long ago can become useless and may even mislead the search. Therefore, these aged decisions are considered less critical and should be erased periodically.

The probability smoothing technique adopted in our PLFITS algorithm is based on work of [62,63] inspired by the forgetting mechanisms in local search algorithms for satisfiability [31]. It works as follows. For each element p_{im} ($i \in A$, $m \in G$), if its value achieves a given predefined threshold p_0, then it is reduced by multiplying a smoothing coefficient ρ $(0 < \rho < 1)$ to forget some earlier decisions. To ensure that the probability vector p_i for each aircraft i has a sum value of one after probability smoothing, we scale all k probabilities $p_{ij}(1 \leq j \leq k)$ by dividing them with a coefficient $1-(1-\rho)*p_{im}$. Additional details about the probability smoothing technique can be found in [62,63].

4 Computational assessment

In this section, we report extensive computational results of the proposed PLFITS algorithm on three sets of real-world benchmark instances. Comparisons are made with several reference algorithms.

4.1 Benchmark instances and parameter setting

The first set contains 20 benchmark instances collected by Cheng et al. [12] arising from ICN, South Korea. ICN is the largest airport in South Korea, and it has two terminals. Following the work of [12] we only consider the gates and exits of Terminal 1, as shown in Fig. 5. Terminal 1 has 74 gates and 14 exits.

In these 20 instances, all 74 fixed gates are used, and all gates can handle the largest aircraft. The number of flights varies from 279 to 304. ICN daily flight data are obtained from FlightStats [1] .

The 20 instances can be classified into three groups by setting the percentage of transfer passengers π respectively to 0.1, 0.3, and 0.5.

- Group 1 includes seven instances (one instance per weekday) with $\pi = 0.1$ (Unfortunately, the instance corresponding to Thursday is no more available in the literature).
- Group 2 includes seven instances with $\pi = 0.3$.
- Group 3 includes seven instances with $\pi = 0.5$.

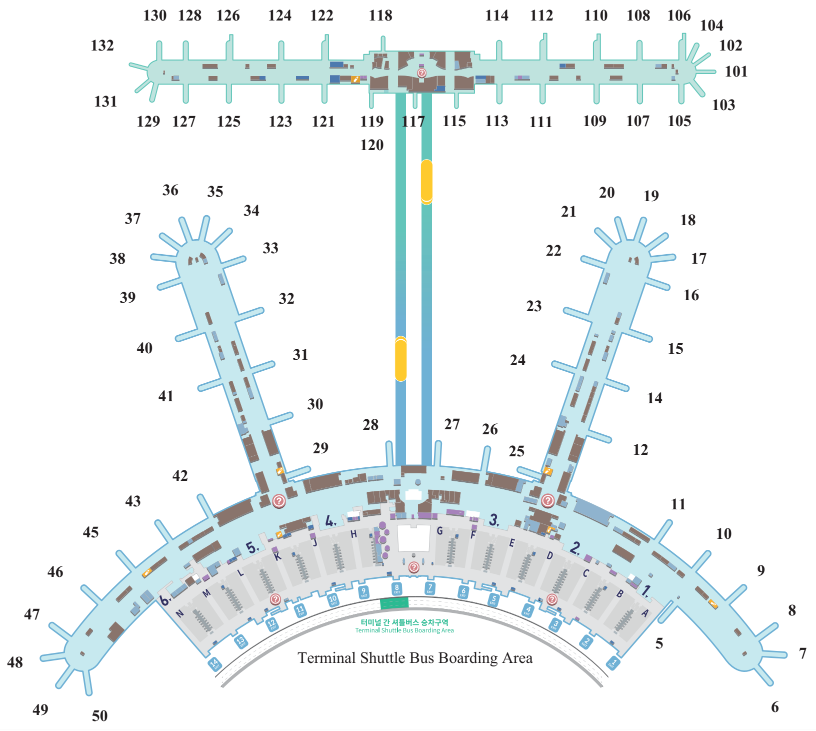


Fig. 5. Gates and exits at Terminal 1 of ICN (Source: https://www.airport.kr/ap/ch/map/mapInfo.do. Accessed on October 12, 2021).

The two other sets of real-world instances are generated by Karsu et al. [33] and are respectively based on ESB and ISL. The two airports respectively have 18 gates (nine domestic, nine international) and 38 gates (12 domestic, 26 international). The numbers of aircraft in these instances are set to be 50, 100, 150, and 200. According to the apron (remote gate) requirement, the 160 instances can be classified into the following two types:

- Low apron requirement: this group includes instances where the arrival time

[1] http://www.flightstats.com/.

a_i of flight i is randomly taken in the interval $[0, 300]$, while the departure time d_i is randomly taken in the interval $[a_i + 30, a_i + 60]$.
- High apron requirement: this group includes instances where the arrival time a_i of flight i is randomly taken in the interval $[0, 150]$, while the departure time d_i is randomly taken in the interval $[a_i + 60, a_i + 120]$.

The proposed PLFITS algorithm is implemented in C++ and compiled by GNU g++ with the -O3 option flag [2]. All experiments are conducted on a 2.6 GHz Intel E5-2670 computer with 2G RAM, running Linux.

Table 1
Parameter setting.

Parameter	Section	Description	Considered values	Final value
ε	3.2	noise probability	{0.1, 0.2, 0.3, 0.4, 0.5}	0.3
ω	3.3	search depth of the local search	{100, 200, 300, 400, 500}	200
u_p	3.3	frequency of updating the penalty parameter	{3, 5, 7, 9, 11}	5
τ	3.3	adjustment to the value of the penalty parameter	{5000, 10000, 15000, 20000, 25000}	10000
α	3.4	reward factor	{0.05, 0.10, 0.15, 0.20, 0.25}	0.10
β	3.4	penalization factor	{0.2, 0.3, 0.4, 0.5, 0.6}	0.5
γ	3.4	compensation factor	{0.1, 0.2, 0.3, 0.4, 0.5}	0.3
ρ	3.5	smoothing coefficient	{0.1, 0.3, 0.5, 0.7, 0.9}	0.5
p_0	3.5	smoothing threshold	{0.80, 0.85, 0.90, 0.95, 0.99}	0.99

The PLFITS algorithm requires nine parameters (Table 1). To tune these parameters, we utilize the IRACE tool, which employs the iterated racing method [3, 7, 38] to automatically determine the required parameters from a set of finite parameter configurations. IRACE is run on five randomly selected instances, and the tuning budget is set to be 2000 PLFITS executions with a time limit of 200 seconds. The setting of the parameters recommended by IRACE is shown as the final value in Table 1 and used for our experiments.

4.2 Computational results and comparisons with reference methods

Although various heuristic approaches have been proposed in the literature to solve GAP, most of these approaches were tested on different benchmark instances. Furthermore, most of these instances are not publicly available. For this reason, we use those algorithms reporting results on the three sets of real-world instances as the reference algorithms. These reference algorithms include a simulated annealing tabu search (SATS) [12], a tabu search with path relinking (TSPR) [13], and a filtered beam search (FBS) [33]. We also re-implement two recent cutting-edge swarm intelligence algorithms, namely, an improved ant colony optimization (REICMPACO) algorithm [17], and a fuzzy bee colony optimization (REFBCO) algorithm [15]. As shown in [17] and [15], REICMPACO and REFBCO achieve excellent performances for GAP. Thus,

[2] The code of our algorithm will be publicly available at https://github.com/MINGJIE666/GAP.

the two algorithms can be considered the cutting-edge algorithms for GAP. Following the same experimental protocol of SATS and TSPR, we run PLFIT-S, REICMPACO, and REFBCO independently with ten runs per instance, each run is limited to 200 seconds. For the computational comparisons, given that the source codes of SATS, TSPR, and FBS are unavailable, the numerical results reported by the two algorithms are directly compiled from the original papers.

Notably, the stopping conditions of these compared algorithms are different. Specifically, TSPR terminates when 1100 iterations are reached. The termination criterion of SATS is that the cooling temperature attained a given predefined threshold $T = 0.01$. FBS is set a time limit of one hour. Moreover, these compared algorithms are executed under different computing platforms. For instance, PLFITS, REICMPACO, and REFBCO are executed on a 2.6 GHz Intel E5-2670 computer, the TSPR and SATS are conducted on a Pentium 4 2.8 GHz computer, and FBS is conducted on an Intel Core i7 2.70 GHz computer. By applying the Standard Performance Evaluation Corporation tool (www.spec.org) to compare the speeds of different processors [56], the obtained scale ratios ($\frac{2.6}{2.8} = 0.93$, $\frac{2.6}{2.7} = 0.96$) of the CPU frequencies indicate that the processor used by our algorithm is slightly slower than those used by the three other reference algorithms. Conducting an absolutely fair comparison between PLFITS and the reference algorithms is difficult due to the differences in computing platforms and termination conditions. The comparison mainly focuses on the solution quality in terms of the objective values, while the timing information is provided only for indicative purposes.

4.2.1 Computational results on the ICN instances

Tables 2–4 summarize the comparative results of our PLFITS algorithm with the four reference algorithms on the three groups of 20 benchmark instances with $\pi = 0.1, 0.3, 0.5$, respectively. Column 'f_{bk}' presents the best-known results obtained with the five reference algorithms. Columns 'f_{best}', 'f_{avg}' and 't_{avg}' respectively show the best objective value, average objective value, and average running time in seconds to reach the final objective value across the ten independent runs. Notably, that the average objective values of the reference algorithms are unavailable. Column 'Gap' presents the percent gap between the best objective value attained with PLFITS and the best-known result 'f_{bk}', which is computed as $100 * (f_{best} - f_{bk})/f_{bk}$. A negative gap value indicates an improved best result. In addition, the row 'p-value' presents the results from the non-parametric Friedman test applied to the results of PLFITS and each reference algorithm, where a p-value smaller than 0.05 implies a statistically significant difference between the compared results.

From the results reported in Tables 2–4, we can make the following observations.

First, the proposed PLFITS algorithm competes very favorably with the reference algorithms in terms of solution quality. For the f_{best} indicator, the algorithm improves the current best-known results for the instances (as indicated by the negative gap values). With the increase in π, the difficulty of the instance also increases because the network of transferring passengers becomes more complex. The improvement ranges from at least 2.6% for all the instances with $\pi = 0.1$ up to 11.85% for the instances with $\pi = 0.5$. Given that $\pi = 0.5$ implies that half of the passengers are transit passengers, these results indicate that the PLFITS algorithm is particularly effective for handling this most challenging situation. The small p-values (less than 0.05) from the non-parametric Friedman test further confirm the significant differences between PLFITS and the compared algorithms on this group of instances.

Second, in terms of computational efficiency, the PLFITS algorithm also remains very competitive. In fact, compared with the reference algorithms, our algorithm requires less computation time to attain better solutions for all the instances. This comparative assessment shows high competitiveness of the proposed PLFITS algorithm on the ICN instances.

To further confirm the effectiveness of the proposed algorithm, we report the comparative results of PLFITS with FBS, REICMPACO, and REFBCO on the ESB and ISL instances in the next section.

4.2.2 Computational results on the ESB and ISL instances

To further evaluate the performance of the proposed PLFITS algorithm, we also test our PLFITS algorithm on an extended GAP model proposed by Karsu et al. [33] very recently. In this model, the number of fixed gates is insufficient to accommodate all aircraft. In this case, an aircraft will be assigned to the apron (remote gate) as long as no gate nearing the terminal is available. Usually, passengers from the gates arrive at their terminal by walking, while passengers from the apron are brought to the terminal by transfer buses or people movers. Thus, assigning an aircraft to the apron will increase passengers' discomfort and airport's operating burden. The objective of this GAP model is first to minimize the number of aircraft assigned to the apron and then to minimize the total passenger walking distance among the solutions with the minimum apron usage.

In general, the distance to the apron is significantly larger than that to any fixed gate. As a result, assigning an aircraft to an apron will also lead to solutions with a much larger objective value in terms of the total walking distance traveled by all passengers. Thus, even if our PLFITS algorithm only simply considers the objective of minimizing the passenger walking distance, it can also potentially minimize the number of apron assignments. Therefore, we can adapt the proposed PLFITS algorithm directly to solve the extended GAP model proposed by Karsu et al. [33].

Tables 5 and 6 show the computational results of PLFITS and the three reference algorithms FBS, REICMPACO, and REFBCO on the ESB and ISL instances. In Tables 5 and 6, each row includes ten instances of the same size indicated by $m \times n$, where m is the numbers of aircraft and n is the number of fixed gates. Columns 'Avg f_{best}' and 'Avg t_{best}' respectively report the average value of the minimum passenger walking distance obtained by each algorithm on each set of ten instances and the average running time in seconds required by these reference algorithms to reach their minimum passenger walking distance on the ten instances. Column '#Apron' reports the number of instances for which the minimum number of apron usage is achieved out of the ten instances. Notably, the number of minimum apron usage can be obtained by solving the maximum cost network flow model [33]. However, the assignment of the aircraft to the gate (apron) cannot be determined by this model, and other exact or heuristic methods are needed for this assignment.

From Tables 5 and 6, we observe that the results obtained by our PLFITS algorithm are highly competitive with those obtained by FBS, REICMPACO, and REFBCO in terms of the passenger walking distance and the number of minimum apron usage. Specifically, in terms of the walking distance, PLFITS can achieve the new best solutions for all the 160 instances, with a reduction rate to the total walking distance ranging from 0.47% up to 5.22%. In terms of the number of minimum apron usage, PLFITS can reach the minimum number on 123 cases out of the 160 instances (56 ESB instances and 67 ISL instances), while this performance can be done by FBS, REICMPACO, and REFBCO, respectively on 78, 90, and 87 cases. Furthermore, the small p-values (less than 0.05) from the non-parametric Friedman test confirm the statistically significant differences between PLFITS and the reference algorithms in terms of the passenger walking distance and the number of minimum apron usage.

In summary, this comparative assessment further confirms the effectiveness and robustness of the proposed PLFITS algorithm on different GAP models and different data sets. In the next section, we show additional experiments to analyze several vital algorithmic components of the PLFITS algorithm to shed light on the understanding of its performance.

Table 2
Comparative results on the 7 ICN instances with $\pi = 0.1$.

			SATS		TSPR		REICMPACO			REFBCO			PLFITS			
Date	Instance size	f_{bk}	f_{best}	t_{avg}	f_{best}	t_{avg}	f_{best}	f_{avg}	t_{avg}	f_{best}	f_{avg}	t_{avg}	f_{best}	f_{avg}	t_{avg}	Gap
Friday	294x74	26945560	27091415	170	26945560	185	26958710	27011620.50	176.53	26883430	27012830.00	185.65	**25926530**	26081164.50	114.82	-3.78
Saturday	290x74	26800315	27001350	174	26800315	185	26925450	27019545.00	159.43	26879255	26943900.50	174.65	**25966585**	26087735.00	95.25	-3.11
Sunday	304x74	29764555	30016505	193	29764555	201	29672665	29868690.50	169.63	29784010	29900010.50	165.32	**28910295**	29053005.00	104.92	-2.87
Monday	297x74	27554290	27554290	185	27668210	207	27370585	27435915.50	158.52	27054520	27210995.00	186.65	**26699195**	26852055.50	122.00	-3.10
Tuesday	290x74	25780535	26055045	180	25780535	196	25740330	25870370.00	189.98	25682365	25835390.00	179.53	**24906875**	24983420.50	122.11	-3.39
Wednesday	279x74	24875240	25092430	151	24875240	173	24552775	24689300.00	198.92	24883320	25087735.00	200.01	**24227945**	24315138.00	127.04	-2.60
Thursday*	289x74	27155365	27515505	168	27155365	190										
p-value			0.014	0.014	0.014	0.014	0.014	0.014	0.014	0.014	0.014	0.014				

* The data of Thursday instance is no more available in the literature.

Table 3
Comparative results on the 7 ICN instances with $\pi = 0.3$.

			SATS		TSPR		REICMPACO			REFBCO			PLFITS			
Date	Instance size	f_{bk}	f_{best}	t_{avg}	f_{best}	t_{avg}	f_{best}	f_{avg}	t_{avg}	f_{best}	f_{avg}	t_{avg}	f_{best}	f_{avg}	t_{avg}	Gap
Friday	294x74	29274210	29325270	184	29274210	196	28452540	28754265.00	185.45	29330105	29609035.50	175.65	**27158465**	27269086.00	143.94	-7.23
Saturday	290x74	28904000	29378545	194	29272310	224	29020050	29128185.00	167.14	28940975	29058645.00	165.41	**26977360**	27130970.50	117.11	-6.67
Sunday	304x74	31642165	31690910	199	31642165	226	31745625	31846000.50	178.63	31578310	31668465.00	184.63	**29673555**	29788407.00	136.52	-6.22
Monday	297x74	29798700	29798700	219	30025230	228	29837770	29965120.00	163.21	29413050	29601815.00	145.32	**27739185**	27887095.50	155.72	-6.91
Tuesday	290x74	27588135	28050095	189	27898320	211	27894450	27984420.50	167.58	27596960	27645165.00	167.34	**25712685**	25807973.00	131.25	-6.80
Wednesday	279x74	27588215	27816840	156	27588215	174	27577455	27605650.50	192.36	27605650	27711180.50	187.65	**25456930**	25637756.50	145.80	-7.73
Thursday	289x74	29402315	29472105	186	29402315	205	28509725	28798030.50	132.02	28347840	28696670.00	196.54	**27056815**	27218636.50	145.57	-7.98
p-value			0.008	0.008	0.008	0.008	0.008	0.008	0.008	0.008	0.008	0.008				

Table 4
Comparative results on the 7 ICN instances with $\pi = 0.5$.

			SATS		TSPR		REICMPACO			REFBCO			PLFITS			
Date	Instance size	f_{bk}	f_{best}	t_{avg}	f_{best}	t_{avg}	f_{best}	f_{avg}	t_{avg}	f_{best}	f_{avg}	t_{avg}	f_{best}	f_{avg}	t_{avg}	Gap
Friday	294x74	31304880	31679360	199	31304880	213	31816275	32139090.00	185.65	31519750	31816275.50	156.57	**27879920**	28099215.00	121.33	-10.94
Saturday	290x74	31337070	31436665	195	31337070	213	30738010	30913920.00	134.61	30868875	30868875.50	168.32	**27971855**	28036805.00	155.17	-10.74
Sunday	304x74	35011240	35011240	217	35050585	261	35403830	36070405.50	147.32	34260520	34434270.00	175.63	**30861150**	30981396.00	166.91	-11.85
Monday	297x74	31900725	31989310	213	31900725	234	30875445	31177835.50	169.31	29872170	30403815.00	197.47	**28559395**	28692073.50	165.01	-10.47
Tuesday	290x74	29858750	30112340	198	30069210	223	29224860	29442845.50	184.23	28912285	29113960.00	179.32	**26664255**	26782922.00	120.78	-10.70
Wednesday	279x74	29667010	29751435	175	29667010	201	27259495	27512960.00	173.65	28135610	28393500.00	185.65	**26457175**	26615683.50	134.41	-10.82
Thursday	289x74	31725500	31896375	194	31755335	228	29323555	29707640.50	197.61	29274900	29420465.00	163.54	**28087180**	28266150.00	155.12	-11.47
p-value			0.008	0.008	0.008	0.008	0.008	0.008	0.008	0.008	0.008	0.008				

Table 5
Comparative results on 80 ESB instances.

		FBS			REICMPACO			REFBCO			PLFITS			
Apron requirement	Instance size	Avg f_{best}	Avg t_{best}	#Apron	Avg f_{best}	Avg t_{best}	#Apron	Avg f_{best}	Avg t_{best}	#Apron	Avg f_{best}	Avg t_{best}	#Apron	Avg Gap
Low	50x18	25482.2	0.54	10/10	27896.3	154.13	10/10	26879.3	96.64	10/10	**25123.6**	72.62	10/10	-1.96
	100x18	91566.9	7.20	2/10	93256.8	114.65	3/10	91641.3	112.87	4/10	**87640.2**	85.21	10/10	-4.40
	150x18	174984.3	30.84	0/10	181234.5	135.63	0/10	184631.2	97.32	0/10	**173854.3**	76.32	0/10	-0.57
	200x18	344250.2	91.32	0/10	365953.1	126.74	0/10	354213.6	73.21	0/10	**342005.3**	115.65	0/10	-0.58
High	50x18	81056.4	0.36	4/10	85214.6	102.52	6/10	83542.4	85.62	5/10	**78345.6**	76.27	10/10	-2.47
	100x18	210763.8	4.71	5/10	232564.5	95.65	6/10	225634.3	114.21	5/10	**209135.5**	85.32	10/10	-0.47
	150x18	297333.2	21.10	5/10	325745.1	89.72	5/10	315263.4	164.96	6/10	**295213.3**	99.14	9/10	-0.67
	200x18	494978.9	67.63	5/10	548564.3	73.25	5/10	536254.7	189.56	6/10	**494635.9**	112.62	7/10	-0.48
#Best		0/80		31/80	0/80		35/80	0/80		36/80	80/80		56/80	
p-value		4.7e-3			4.7e-3			4.7e-3						

Table 6
Comparative results on 80 ISL instances.

		FBS			REICMPACO			REFBCO			PLFITS			
Apron requirement	Instance size	Avg f_{best}	Avg t_{best}	#Apron	Avg f_{best}	Avg t_{best}	#Apron	Avg f_{best}	Avg t_{best}	#Apron	Avg f_{best}	Avg t_{best}	#Apron	Avg Gap
Low	50x38	36178.8	4.30	10/10	38645.8	185.21	10/10	37925.3	54.65	10/10	**35420.2**	118.32	10/10	-2.76
	100x38	84792.7	71.58	10/10	86452.3	146.45	10/10	88126.5	113.21	10/10	**83820.9**	105.21	10/10	-1.18
	150x38	124072.7	361.24	10/10	135483.1	139.69	10/10	128362.2	184.62	10/10	**120532.1**	65.85	10/10	-3.22
	200x38	249024.6	1094.92	4/10	276532.9	95.54	5/10	254563.2	129.63	6/10	**236523.2**	95.32	7/10	-5.22
High	50x38	58165.0	4.24	10/10	61235.6	182.32	10/10	57698.5	132.52	10/10	**56127.5**	78.58	10/10	-3.44
	100x38	278693.7	60.91	1/10	294652.1	132.52	3/10	280321.3	179.32	1/10	**268432.1**	87.12	9/10	-3.60
	150x38	443563.6	281.09	0/10	463252.3	112.87	3/10	456325.6	185.62	2/10	**435216.3**	76.53	8/10	-2.25
	200x38	798524.5	804.25	2/10	812369.5	142.37	2/10	823641.2	156.27	2/10	**796231.2**	143.95	3/10	-0.50
#Best		0/80		47/80	0/80		55/80	0/80		51/80	80/80		67/80	
p-value		4.7e-3			4.7e-3			4.7e-3						

5 Analysis

In this section, we analyze the two essential ingredients of the proposed algorithm to verify their impacts on the algorithm's performance, namely, the probability learning strategy and the feasible and infeasible search strategy. Then, we perform the sensitivity analysis to show the impact of parameters.

5.1 Effect of the probability learning strategy

Table 7
Comparative results between FITS and PLFITS on the 20 ICN instances with $\pi = 0.1, 0.3, 0.5$.

			f_{best}		f_{avg}		t_s	
Instance	size	π	FITS	PLFITS	FITS	PLFITS	FITS	PLFITS
Friday	294x74	0.1	26325490	**25926530**	26555265.0	**26081164.5**	163.29	**114.82**
Saturday	290x74	0.1	26213935	**25966585**	26474211.5	**26087735.0**	182.02	**95.25**
Sunday	304x74	0.1	29024365	**28910295**	29348397.5	**29053005.0**	135.40	**104.92**
Monday	297x74	0.1	27053175	**26699195**	27339034.0	**26852055.5**	153.23	**122.00**
Tuesday	290x74	0.1	25211755	**24906875**	25443803.0	**24983420.5**	164.61	**122.11**
Wednesday	279x74	0.1	24555800	**24227945**	24718988.0	**24315138.0**	159.13	**127.04**
Friday	294x74	0.3	27831740	**27158465**	28007535.5	**27269086.0**	149.63	**143.94**
Saturday	290x74	0.3	27317185	**26977360**	27534134.5	**27130970.5**	152.89	**117.11**
Sunday	304x74	0.3	29910640	**29673555**	30352146.5	**29788407.0**	156.58	**136.52**
Monday	297x74	0.3	28133755	**27739185**	28588243.5	**27887095.5**	163.97	**155.72**
Tuesday	290x74	0.3	25944565	**25712685**	26279554.0	**25807973.0**	173.36	**131.25**
Wednesday	279x74	0.3	26109645	**25456930**	26288654.5	**25637756.5**	181.38	**145.80**
Thursday	289x74	0.3	27535070	**27056815**	27984739.5	**27218636.5**	159.91	**145.57**
Friday	294x74	0.5	28544040	**27879920**	28815379.0	**28099215.0**	179.67	**121.33**
Saturday	290x74	0.5	28232265	**27971855**	28751689.0	**28036805.0**	**135.93**	155.17
Sunday	304x74	0.5	31186605	**30861150**	31690562.5	**30981396.0**	**123.37**	166.91
Monday	297x74	0.5	29026455	**28559395**	29645604.5	**28692073.5**	**156.76**	165.01
Tuesday	290x74	0.5	27205970	**26664255**	27428849.5	**26782922.0**	160.22	**120.78**
Wednesday	279x74	0.5	26705795	**26457175**	27183861.0	**26615683.5**	168.00	**134.41**
Thursday	289x74	0.5	28567675	**28087180**	29000327.5	**28266150.0**	**149.06**	155.12
#Best			0	**20**	0	**20**	4	**16**
Average			27531796.25	**27144667.50**	27871549.00	**27279334.43**	158.42	**134.04**
p-value			8e-6		8e-6		7e-3	

The probability learning strategy is a significant component of our PLFITS algorithm. To shed light on the benefit of this strategy, we compare PLFITS with a PLFITS variant (named by FITS), where (i) the probability learning strategy is removed from PLFITS and (ii) the initial solution for each round of the algorithm is randomly constructed instead of relying on the probability matrix P.

This study is based on the 20 instances used in Section 4. PLFITS and its variant version FITS are run ten runs independently with a time limit of 200 seconds per run to solve each instance. The comparative results are summarized in Table 7 with the same information as in Section 4.2. In addition, the row '#Best' shows the number of instances for which each compared algorithm attains the best results between the two compared approaches.

Table 7 indicates that, in terms of the best and average objective values, PLFITS dominates FITS for all the instances. In terms of the running time, PLFITS is more efficient for 16 out of the 20 instances. Furthermore, the p-values from the non-parametric Friedman test confirm a statistically significant difference between PLFITS and FITS in terms of the best objective values, average objective values, and average computation times. This experiment indicates that the probability learning strategy is important to the performance of the proposed algorithm.

5.2 Usefulness of the feasible and infeasible search

As introduced in Section 3.3, the PLFITS algorithm employs a feasible and infeasible search mechanism to explore the search space of GAP. PLFITS is compared with an algorithmic variant (called PLFTS), where only feasible solutions are allowed to verify the importance of this mixed search mechanism. To achieve this, we strongly penalize any infeasible solution by setting the penalty coefficient φ of the extended evaluation function (Eq. 6, Section 3.3.1) to an extremely large value. We run both methods using the same instances and the same experimental protocol as described in Section 5.1.

Table 8
Comparative results between PLFTS and PLFITS on the 20 ICN instances with $\pi = 0.1, 0.3, 0.5$.

			f_{best}		f_{avg}		t_s	
Instance	size	π	PLFTS	PLFITS	PLFTS	PLFITS	PLFTS	PLFITS
Friday	294x74	0.1	26102510	**25926530**	26146190.5	**26081164.5**	**105.53**	114.82
Saturday	290x74	0.1	26041120	**25966585**	26232517.5	**26087735.0**	107.38	**95.25**
Sunday	304x74	0.1	29161050	**28910295**	29280455.0	**29053005.0**	153.67	**104.92**
Monday	297x74	0.1	26931095	**26699195**	27010025.5	**26852055.5**	134.74	**122.00**
Tuesday	290x74	0.1	**24873590**	24906875	25097676.5	**24983420.5**	**103.51**	122.11
Wednesday	279x74	0.1	24317780	**24227945**	24428795.0	**24315138.0**	**110.41**	127.04
Friday	294x74	0.3	27258735	**27158465**	27398108.0	**27269086.0**	**138.54**	143.94
Saturday	290x74	0.3	27210570	**26977360**	27307298.5	**27130970.5**	120.47	**117.11**
Sunday	304x74	0.3	29879825	**29673555**	30029998.0	**29788407.0**	**108.55**	136.52
Monday	297x74	0.3	27973160	**27739185**	28072872.5	**27887095.5**	**130.23**	155.72
Tuesday	290x74	0.3	25887825	**25712685**	25959695.0	**25807973.0**	148.83	**131.25**
Wednesday	279x74	0.3	25607065	**25456930**	25802067.0	**25637756.5**	110.66	**145.80**
Thursday	289x74	0.3	27262465	**27056815**	27390705.0	**27218636.5**	**144.28**	145.57
Friday	294x74	0.5	28122755	**27879920**	28298139.0	**28099215.0**	124.26	**121.33**
Saturday	290x74	0.5	28045115	**27971855**	28201434.5	**28036805.0**	**106.60**	155.17
Sunday	304x74	0.5	30996540	**30861150**	31194997.5	**30981396.0**	**150.79**	166.91
Monday	297x74	0.5	28740995	**28559395**	28898781.0	**28692073.5**	**133.58**	165.01
Tuesday	290x74	0.5	26791695	**26664255**	26994782.0	**26782922.0**	132.90	**120.78**
Wednesday	279x74	0.5	26525680	**26457175**	26672817.0	**26615683.5**	**113.21**	134.41
Thursday	289x74	0.5	28215415	**28087180**	28405748.5	**28266150.0**	**145.97**	155.12
#Best			1	**19**	0	**20**	**12**	8
Average			27297249.25	**27144667.50**	27441155.18	**27279334.43**	**126.21**	134.04
p-value			6e-6		8e-6		0.18	

Table 8 summarizes the comparative results between PLFITS and PLFTS. From Table 8, one observes that PLFITS fully dominates PLFTS in terms of the best and average objective values, and the statistically significant difference between PLFITS and the variant PLFTS is confirmed by the small p-values from the non-parametric Friedman test. In terms of the average computation time, PLFITS needs slightly more time than PLFTS (134.04 seconds vs. 126.21 seconds) to reach their best solutions, which may be of different quality. This experiment clearly demonstrates that the feasible and infeasible search mechanism positively contributes to the performance of the algorithm.

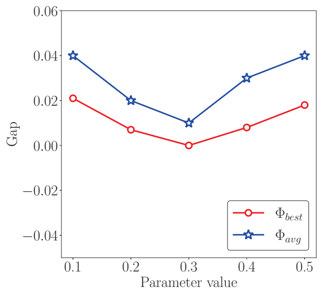
5.3 Sensitivity analysis of the parameters

The PLFITS algorithm requires nine parameters as shown in Table 1, including ε (noise probability), ω (search depth of the local search), u_p (frequency of updating the penalty parameter), τ (adjustment to the value of the penalty parameter), α (reward factor), β (penalization factor), γ (compensation factor), p_0 (smoothing threshold), and ρ (smoothing coefficient).

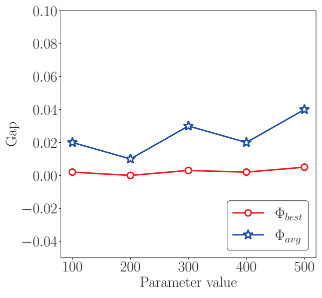
We first perform a 2-level full factorial experiment [45] to test the interaction effects among these nine parameters. The low and high levels of each parameter are respectively set as the smallest and largest values in Table 1. Given that each parameter has two levels, this leads to a total of 512 ($2^9 = 512$) combinations for the nine parameters. The experiment is conducted on five randomly selected instances used in Sections 4.1. Each instance is independently solved 20 times under a time limit of 200 seconds per run for each combination of parameters. Then the average results of the best objective values obtained on the five instances are considered for each parameter combination. The Friedman test indicates no statistically significant difference (p-values > 0.05) in terms of the considered average results, implying that the interaction effects among these nine parameters are not statistically significant.

Then for each single parameter, we perform a one-at-a-time sensitivity analysis [30] to analyze the influence of the parameter on the performance of PLFITS and to determine its most suitable value. To achieve this, we test its value within the range of possible values as listed in Column 4 of Table 1 while fixing the other parameters to their default values in Table 1. The PLFITS algorithm is independently run 20 times under a time limit of 200 seconds per run for each parameter value. We report the obtained best objective value (denoted by Φ_{best}) and the average objective value (denoted by Φ_{avg}) across the 20 runs on the five instances in Fig. 6, where the X-axis indicates the values of each parameter, and the Y-axis presents the best/average gaps to the best-known results over the five instances. From Fig. 6, we observe that the recommended parameter values from this calibration experiment are the same as those recommended by IRACE.

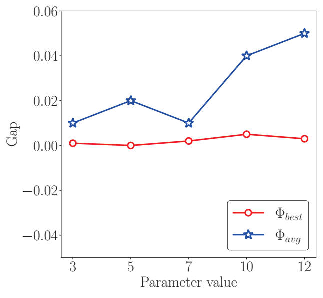
Furthermore, we employ the Friedman test to determine whether there exists a statistically significant difference in solution qualities for different values of a given parameter. The Friedman test indicates that the PLFITS algorithm is sensitive to the setting of ε (with p-value $= 0.008$), and α (with p-value $= 0.038$), while this is not the case for the other parameters. The PLFITS algorithm is sensitive to ε, possibly due to the fact that a too small/large ε may lead to a too greediness/randomness into the probability learning procedure. Similarly, increasing/decreasing the value of α means strengthening/weakening the power of the probability matrix, making the probability learning procedure more greedy/random. Thus, suitable values for the ε and α parameters are critical to the performance of the PLFITS algorithm. To sum up, there are no significant interaction effects among the nine parameters required by PLFITS. Meanwhile, PLFITS is sensitive to the settings of ε and α. Therefore, if the user needs to tune the parameters, more effort should be devoted to ε and α.



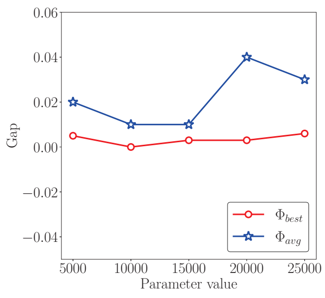
(a) ε: p-value=0.008



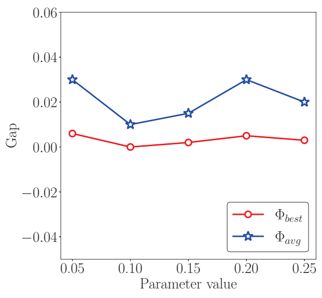
(b) ω: p-value=0.162



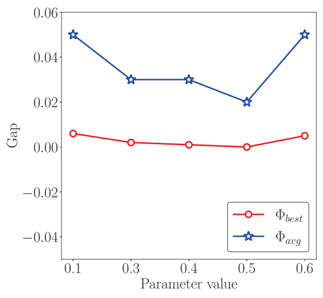
(c) u_p: p-value=0.486



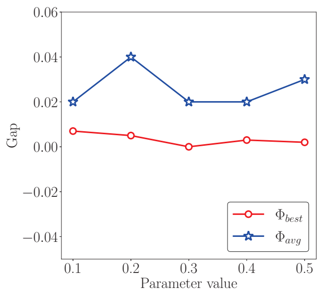
(d) τ: p-value=0.132



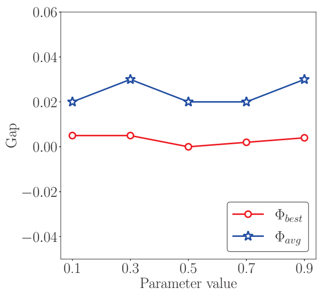
(e) α: p-value=0.038



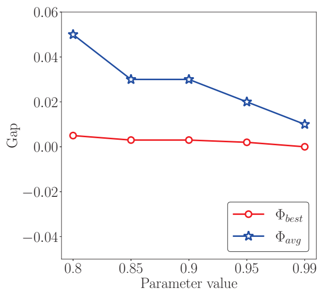
(f) β: p-value=0.061



(g) γ: p-value=0.332



(h) ρ: p-value=0.162



(i) p_0: p-value=0.052

Fig. 6. Sensitivity analysis of the parameters with the significance level of 0.05.

6 Conclusion

We proposed an effective heuristic algorithm for the airport gate assignment problem, which is one complex and vital decision problem in airport facility management. The algorithm relies on a probability learning strategy to build initial assignments, which are improved by a feasible and infeasible tabu search procedure. The probability learning component uses the idea of reinforcement learning to gain useful information from past solutions to create new promising solutions. The feasible and infeasible tabu search component explores both feasible and infeasible candidate solutions to attain high-quality solutions.

We assessed the performance of the proposed algorithm on three sets of 180 real-world GAP benchmark instances collected from ICN [12], ESB [33], and ISL [33]. Experimental evaluations demonstrate that the algorithm competes very favorably with reference GAP methods in the literature by finding improved best-known results (i.e., new upper bounds) for all the 180 real-world instances. We also conducted additional experiments to study the impacts of the probability learning strategy and the feasible and infeasible search mechanism. The code of the algorithm will be publicly available. Practitioners and researchers working on GAP and related problems can benefit from the code to solve their problems.

The proposed algorithm benefits from the probability learning and the mixed feasible-infeasible search strategies. Given that these strategies are rather general and contribute significantly to the performance of the proposed algorithm, it would be very interesting to investigate their interest in tackling other airport gate assignment problems with different objectives and constraints or other problems with complex constraints. We also realize that the proposed algorithm has several weaknesses. First, given that it is a heuristic algorithm, how far the solutions attained by the algorithm are from the optimal solutions cannot be determined. As a result, more efforts are needed to investigate exact and approximation methods with quality guarantees. Second, the proposed algorithm requires nine parameters, which makes the task of parameter tuning delicate. Therefore, efforts on reducing the number of parameters and designing methods for automatic parameter tuning are of great interest.

Acknowledgments

We are grateful to the reviewers whose comments have helped us improve the work significantly. We would like to thank Dr. Chun-Hung Cheng and his coauthors for sharing the ICN instances with the computational results of TSPR and SATS, as well as Dr. Özlem Karsu for sharing the ESB and

ISL instances with the computational results of FBS. This work is partially supported by the National Natural Science Foundation Program of China [Grant No. 72122006, 71771099, 72131008, 71821001].

References

[1] Aktel, A., Yagmahan, B., Özcan, T., Yenisey, M. M., & Sansarcı, E. (2017). The comparison of the metaheuristic algorithms performances on airport gate assignment problem. *Transportation Research Procedia, 22*, 469-478.

[2] Babić, O., Teodorović, D., & Tošić, V. (1984). Aircraft stand assignment to minimize walking. *Journal of Transportation Engineering, 110*(1), 55-66.

[3] Balaprakash, P., Birattari, M., & Stützle, T. (2007, October). Improvement strategies for the F-Race algorithm: Sampling design and iterative refinement. In *International Workshop on Hybrid Metaheuristics* (pp. 108-122). Springer, Berlin, Heidelberg.

[4] Barnhart, C., Belobaba, P., & Odoni, A. R. (2003). Applications of operations research in the air transport industry. *Transportation science, 37*(4), 368-391.

[5] Benlic, U., Burke, E. K., & Woodward, J. R. (2017). Breakout local search for the multi-objective gate allocation problem. *Computers & Operations Research, 78*, 80-93.

[6] Bihr, R. A. (1990). A conceptual solution to the aircraft gate assignment problem using 0, 1 linear programming. *Computers & Industrial Engineering, 19*(1-4), 280-284.

[7] Birattari, M., Stützle, T., Paquete, L., & Varrentrapp, K. (2002, July). A racing algorithm for configuring metaheuristics. In Proceedings of *GECCO-2002* pp. 11-18.

[8] Bolat, A. (1999). Assigning arriving flights at an airport to the available gates. *Journal of the Operational Research Society, 50*(1), 23-34.

[9] Bolat, A. (2000). Procedures for providing robust gate assignments for arriving aircraft. *European Journal of Operational Research, 120*(1), 63-80.

[10] Bouras, A., Ghaleb, M. A., Suryahatmaja, U. S., & Salem, A. M. (2014). The airport gate assignment problem: a survey. *The Scientific World Journal, Article 923859.*

[11] Chen, Y., & Hao, J. K. (2018). Two phased hybrid local search for the periodic capacitated arc routing problem. *European Journal of Operational Research, 264*(1), 55-65.

[12] Cheng, C. H., Ho, S. C., & Kwan, C. L. (2012). The use of meta-heuristics for airport gate assignment. *Expert Systems With Applications, 39*(16), 12430-12437.

[13] Cheng, C. H., Gunasekaran, A., Ho, S. C., Kwan, C. L., & Ng, T. D. (2017). Hybrid tabu searches for effective airport gate management. *International Journal of Operational Research, 30*(4), 484-522.

[14] Daş, G. S., Gzara, F., & Stützle, T. (2020). A review on airport gate assignment problems: Single versus multi objective approaches. *Omega, 92*, 102146.

[15] Dell'Orco, M., Marinelli, M., & Altieri, M. G. (2017). Solving the gate assignment problem through the fuzzy bee colony optimization. *Transportation Research Part C: Emerging Technologies, 80*, 424-438.

[16] Deng, W., Zhao, H., Yang, X., Xiong, J., Sun, M., & Li, B. (2017). Study on an improved adaptive PSO algorithm for solving multi-objective gate assignment. *Applied Soft Computing, 59*, 288-302.

[17] Deng, W., Xu, J., & Zhao, H. (2019). An improved ant colony optimization algorithm based on hybrid strategies for scheduling problem. *IEEE Access, 7*, 20281-20292.

[18] Deng, W., Xu, J., Zhao, H., & Song, Y. (2020). A novel gate resource allocation method using improved PSO-based QEA. *IEEE Transactions on Intelligent Transportation Systems*, 1-9.

[19] Dijk, B., Santos, B. F., & Pita, J. P. (2019). The recoverable robust stand allocation problem: a GRU airport case study. *OR Spectrum, 41*(3), 615-639.

[20] Ding, H., Lim, A., Rodrigues, B., & Zhu, Y. (2004, January). Aircraft and gate scheduling optimization at airports. In *37th Annual Hawaii International Conference on System Sciences, 2004. Proceedings of the* (pp. 8-pp). IEEE.

[21] Ding, H., Lim, A., Rodrigues, B., & Zhu, Y. (2005). The over-constrained airport gate assignment problem. *Computers & Operations Research, 32*(7), 1867-1880.

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

[23] 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.

[24] Drexl, A., & Nikulin, Y. (2008). Multicriteria airport gate assignment and Pareto simulated annealing. *IIE Transactions, 40*(4), 385-397.

[25] Genç, H. M. (2010). A new solution approach for the airport gate assignment problem for handling of uneven gate demands. In *Proceedings of the Word Conference of Transport Research (WCTR-10).*

[26] Glover, F., & Laguna, M. (1998). Tabu search. In *Handbook of combinatorial optimization* (pp. 2093-2229). Springer, Boston, MA.

[27] Glover, F., & Hao, J. K. (2011). The case for strategic oscillation. *Annals of Operations Research, 183*(1), 163-173.

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

[29] Haghani, A., & Chen, M. C. (1998). Optimizing gate assignments at airport terminals. *Transportation Research Part A: Policy and Practice, 32*(6), 437-454.

[30] Hamby, D. M. (1994). A review of techniques for parameter sensitivity analysis of environmental models. Environmental monitoring and assessment, 32(2), 135-154.

[31] Hutter, F., Tompkins, D. A., & Hoos, H. H. (2002, September). Scaling and probabilistic smoothing: Efficient dynamic local search for SAT. In *International Conference on Principles and Practice of Constraint Programming* (pp. 233-248). Springer, Berlin, Heidelberg.

[32] IATA pressroom. (2016). https://www.iata.org/en/pressroom/pr/2016-10-18-02/.

[33] Karsu, Ö., Azizoğlu, M., & Alanlı, K. (2021). Exact and heuristic solution approaches for the airport gate assignment problem. *Omega, 103*, 102422.

[34] Kim, S. H., Feron, E., & Clarke, J. P. (2013). Gate assignment to minimize passenger transit time and aircraft taxi time. *Journal of Guidance, Control, and Dynamics, 36*(2), 467-475.

[35] Kim, S. H., Feron, E., Clarke, J. P., Marzuoli, A., & Delahaye, D. (2013). Airport gate scheduling for passengers, aircraft, and operation. *arXiv preprint arXiv:1301.3535.*

[36] Prem Kumar, V., & Bierlaire, M. (2014). Multiobjective airport gate assignment problem in planning and operations. *Journal of Advanced Transportation, 48*(7), 902-926.

[37] Lim, A., Rodrigues, B., & Zhu, Y. (2005). Airport gate scheduling with time windows. *Artificial Intelligence Review, 24*(1), 5-31.

[38] López-Ibánez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., & Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives, 3*, 43-58.

[39] Maharjan, B., & Matis, T. I. (2012). Multi-commodity flow network model of the flight gate assignment problem. *Computers & Industrial Engineering, 63*(4), 1135-1144.

[40] Mangoubi, R. S., & Mathaisel, D. F. (1985). Optimizing gate assignments at airport terminals. *Transportation Science, 19*(2), 173-188.

[41] Marinelli, M., Dell'Orco, M., & Sassanelli, D. (2015). A metaheuristic approach to solve the flight gate assignment problem. *Transportation Research Procedia, 5*, 211-220.

[42] Marinelli, M., Palmisano, G., Dell'Orco, M., & Ottomanelli, M. (2015). Fusion of two metaheuristic approaches to solve the flight gate assignment problem. *Transportation Research Procedia, 10*, 920-930.

[43] Marinelli, M., Palmisano, G., Dell'Orco, M., & Ottomanelli, M. (2016). Optimizing Airport Gate Assignments Through a Hybrid Metaheuristic Approach. In *Advanced Concepts, Methodologies and Technologies for Transportation and Logistics* (pp. 389-404). Springer, Cham.

[44] Mokhtarimousavi, S., Talebi, D., & Asgari, H. (2018). A non-dominated sorting genetic algorithm approach for optimization of multi-objective airport gate assignment problem. *Transportation Research Record, 2672*(23), 59-70.

[45] Montgomery, D. C. (2017). Design and analysis of experiments. *John wiley & sons.*

[46] Narendra, K. S., & Thathachar, M. A. (1989). *Learning Automata: An Introduction.* Prentice-Hall, New Jersey.

[47] Nikulin, Y., & Drexl, A. (2010). Theoretical aspects of multicriteria flight gate scheduling: deterministic and fuzzy models. *Journal of Scheduling, 13*(3), 261-280.

[48] Obata, T. (1980). Quadratic assignment problem: Evaluation of exact and heuristic algorithms. *Rensselaer Polytechnic Institute*, New York.

[49] Qin, J., Xu, X., Wu, Q., & Cheng, T. C. E. (2016). Hybridization of tabu search with feasible and infeasible local searches for the quadratic multiple knapsack problem. *Computers & Operations Research, 66*, 199-214.

[50] Şeker, M., & Noyan, N. (2012). Stochastic optimization models for the airport gate assignment problem. *Transportation Research Part E: Logistics and Transportation Review, 48*(2), 438-459.

[51] Shone, R., Glazebrook, K., & Zografos, K. G. (2021). Applications of stochastic modeling in air traffic management: Methods, challenges and opportunities for solving air traffic problems under uncertainty. *European Journal of Operational Research, 292*(1), 1-26.

[52] Sun, W., Hao, J. K., Lai, X., & Wu, Q. (2018). Adaptive feasible and infeasible tabu search for weighted vertex coloring. *Information Sciences, 466*, 203-219.

[53] Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction.* MIT press.

[54] van Schaijk, O. R., & Visser, H. G. (2017). Robust flight-to-gate assignment using flight presence probabilities. *Transportation Planning and Technology, 40*(8), 928-945.

[55] Wauters, T., Verbeeck, K., De Causmaecker, P., & Berghe, G. V. (2013). Boosting metaheuristic search using reinforcement learning. In *Hybrid Metaheuristics* (pp. 433-452). Springer, Berlin, Heidelberg.

[56] Wu, Q., & Hao, J. K. (2015). Solving the winner determination problem via a weighted maximum clique heuristic. *Expert Systems with Applications, 42*(1), 355-365.

[57] Xu, J., & Bailey, G. (2001, January). The airport gate assignment problem: mathematical model and a tabu search algorithm. In *Proceedings of the 34th annual Hawaii international conference on system sciences* (pp. 10-pp). IEEE.

[58] Yan, S., & Huo, C. M. (2001). Optimization of multiple objective gate assignments. *Transportation Research Part A: Policy and Practice, 35*(5), 413-432.

[59] Yan, S., & Tang, C. H. (2007). A heuristic approach for airport gate assignments for stochastic flight delays. *European Journal of Operational Research, 180*(2), 547-567.

[60] Yu, C., Zhang, D., & Lau, H. Y. (2016). MIP-based heuristics for solving robust gate assignment problems. *Computers & Industrial Engineering, 93*, 171-191.

[61] Yu, C., Zhang, D., & Lau, H. Y. (2017). An adaptive large neighborhood search heuristic for solving a robust gate assignment problem. *Expert Systems with Applications, 84*, 143-154.

[62] Zhou, Y., Hao, J. K., & Duval, B. (2016). Reinforcement learning based local search for grouping problems: A case study on graph coloring. *Expert Systems with Applications, 64*, 412-422.

[63] Zhou, Y., Duval, B., & Hao, J. K. (2018). Improving probability learning based local search for graph coloring. *Applied Soft Computing, 65*, 542-553.

[64] Zhou, Q., Benlic, U., Wu, Q., & Hao, J. K. (2019). Heuristic search to the capacitated clustering problem. *European Journal of Operational Research, 273*(2), 464-487.