A Bi-Objective Constrained Robust Gate Assignment Problem: Formulation, Instances and Algorithm

Xinye Cai, Member, IEEE, Wenxue Sun, Mustafa Misir, Kai Chen Tan, Fellow, IEEE, Xiaoping Li, Senior Member, IEEE, Tao Xu, and Zhun Fan, Senior Member, IEEE

Abstract—The gate assignment problem (GAP) aims at assigning gates to aircraft considering operational efficiency of airport and satisfaction of passengers. Unlike the existing works, we model the GAP as a bi-objective constrained optimization problem. The total walking distance of passengers and the total robust cost of the gate assignment are the two objectives to be optimized, while satisfying the constraints regarding the limited number of flights assigned to apron, as well as three types of compatibility. A set of real instances is then constructed based on the data obtained from the Baiyun airport (CAN) in Guangzhou, China. A two-phase large neighborhood search (2PLNS) is proposed, which accommodates a greedy and stochastic strategy (GSS) for the large neighborhood search; both to speed up its convergence and to avoid local optima. The empirical analysis and results on both the synthetic instances and the constructed real-world instances show a better performance for the proposed 2PLNS as compared to many state-of-the-art algorithms in literature. An efficient way of choosing the tradeoff from a large number of nondominated solutions is also discussed in this article.

Index Terms—Bi-objective optimization, large neighborhood search (LNS), Pareto local search (PLS), robust gate assignment, solutions of interest.

I. INTRODUCTION

AIRCRAFT have become one of the major transportation choices with the varying customer needs, travelling prices and the growing number of airports. The airport gates (i.e., aircraft stand positions) play a critical role in the efficient utilization of the aircraft. Thus, their assignment to the aircraft needs to be carefully scheduled, as it greatly affects the efficient operation of airports, as well as the passengers’ satisfaction. Such assignment task is usually called the gate assignment problem (GAP) [24]. The GAP requires delivering gate schedules considering various factors, e.g., the aircraft types, the domestic/international attribute of the flight, the flights’ arrival/departure time, the total passenger number of the flight, and the gate preferences [22].

From a practical point of view, the gate assignments should be determined under the consideration of the possible changes on the existing flight schedules. In other words, the gate assignments should be enough robust to the minor changes on the flight schedules [22]. For instance, a late arrival of one aircraft may result in a chain of delayed arrivals for other aircraft assigned to the same gate. This may make the current gate assignments inapplicable, leading to a complete rescheduling.

For GAP, maintaining the operational efficiency in airports and the passengers’ satisfaction simultaneously is a very complicated task. Under this circumstance, the GAP is modelled as a combinatorial multiobjective optimization problem (CMOP) with different and also possibly conflicting objectives [17], [19], [20], [23]–[25], [31], [36], [38]. Unlike a single-objective optimization problem where only one optimal solution exists, the multiobjective optimization problems have a set of tradeoff solutions among different
objectives. These solutions are called a Pareto set (PS), forming a Pareto front (PF) as their projection on the objective space.

Despite the multiobjective nature of the GAP, it has been widely handled as a single-objective optimization problem. For instance, in [23], the GAP contains three objectives, that is, 1) maximizing the total assignment preference score; 2) minimizing the number of ungated flights; and 3) minimizing the number of tows. These objectives were aggregated into a single objective via the weighted-sum approach. The weights are usually determined either by the airport or the airline manager.

In this article, the GAP is formulated using a constrained bi-objective robust gate assignment model (CBR-GAM). The walking distance of the passengers and the robust cost of the gate assignment are treated as the two objectives. The first objective is essentially used to evaluate the passenger satisfaction [22]. The latter objective relates to the delay cost on the idle time allocation [18].

For easy access, passenger bridges are usually equipped between terminal gates and aircraft. Yet, aprons need busses to transport passengers. For shortening walking distance of passengers, terminal gates are assigned to flights preferably over aprons [24]. The assignment of flights to aprons is likely to reduce the operational efficiency of the airport. Under this circumstance, the number of flights that can be assigned to the apron is limited as a constraint in the CBR-GAM.

To address the CBR-GAM efficiently, we propose a two-phase large neighborhood search (2PLNS). In the first phase, the bi-objective optimization problem is decomposed into a set of single-objective subproblems using the weighted-sum approach [40]. One solution is maintained for each subproblem. The collaborative local search (LS) is conducted among subproblems for the fast convergence toward the PF. After that, Pareto LS (PLS) is conducted on the resulting obtained solutions, for further approximating the whole PF.

Although large neighborhood search (LNS) is an effective solution strategy, it has poor scalability with the increasing neighborhood size [28]. To alleviate such an issue as well as avoid local optima, a greedy and stochastic strategy (GSS) is introduced for guiding the LS/PLS.

To validate the effectiveness and efficiency of 2PLNS on CBR-GAM, experimental studies have been conducted on both synthetic benchmarks of different scales and a set of real instances, which is constructed based on data obtained from the Baiyun airport (CAN) in Guangzhou, China. The same experiments have also been conducted by a group of state-of-the-art multiobjective optimization algorithms for comparison. The results show the effectiveness of 2PLNS in terms of both solution quality and scalability. In addition, solutions obtained by the proposed 2PLNS are analyzed through Gantt charts for the real instance.

In summary, the main contributions of this article can be listed as follows.

1) A constrained bi-objective model is proposed for the robust gate assignment problem. Differently from the existing works, more realistic aircraft-gate company, flight attribute and flight apron compatibility have been modeled as constraints (see Section III-C).

2) An objective function for measuring the robustness of the gate assignment is proposed. As a new addition to the literature, a weighted sum of the robust costs regarding all the flights is utilized since the importance of the flights can significantly differ from each other (see Section III-B).

3) A 2PLNS is designed for solving the aforementioned bi-objective model. A GSS-based LNS is proposed for effectively balancing between the exploitative and explorative search.

The rest of this article is organized as follows. Section II discusses the existing studies on the GAP as well as the relevant optimization approaches. Section III introduces the CBR-GAM in details. 2PLNS is elaborated in Section IV. The construction of real instances based on data obtained from CAN in China, as well as the synthetic instances, is given in Section V. The experimental setups are presented in Section VI. The experimental results are discussed and analyzed in Section VII. Finally, Section VIII concludes this article with a summary and the future research directions.

II. BACKGROUND

A. Gate Assignment Problem

Based on the number of objectives targeted, studies on the GAP can be primarily categorized into: the GAP with one objective, that is, single-objective, and the GAP with more than one objective, that is, multiobjective, as follows.

1) Single-Objective GAP: The majority of the studies try to minimize the passengers’ required walking distance as their sole objective [2], [5], [7], [14], [17], [20], [24], [29], [34], [36], [37]. This distance is usually calculated based on three types of walking area: 1) the distance from check-in to gates for embarking or originating passengers; 2) the distance from gates to the baggage claim areas (check-out) for disembarking or destination passengers; and 3) the distance from gate to gate for transfer or connecting passengers.

The robustness is another commonly used objective in GAP. The robust GAP has already been investigated in [6], [13], [18], [22], [23], [26], [37]. In [6], the idle period of time between two successive utilizations of one gate has been used for measuring the robustness. The robust gate assignments were obtained by minimizing the variance of such idle time. In [18], a utility function for measuring the robustness of the gate assignment was proposed by penalizing the shorter idle-time with much higher cost. In [23], the robustness of the gate scheduling is measured by the expected number of constraint violations, that is, ignoring gate closures, the violation of the shadow restrictions, and the gate conflicts in which two aircraft are assigned to the same gate. Although this approach can ensure that the corresponding constraints are satisfied, it does not provide any effective guidance for the construction of the robust gate assignment. A more comprehensive survey on the robustness in flight gate scheduling can be found in [22].

2) Multiobjective GAP: A considerable number of studies have been conducted on multiobjective GAP, which have
been summarized in Table I. The third column (objectives) in Table I lists the objectives to be optimized in GAP for the corresponding reference. The fourth column (constraints) displays the restrictions to be considered during the process of gate allocation. It is clear to see that most studies only consider two basic constraints: Const-1—only one aircraft can be assigned to a gate at a certain period of time and Const-2—the constraint that every flight must be assigned to exactly one gate.

Since the CBR-GAM is mainly modelled for a real-world scenario, an extensive set of constraints is taken into account. These constraints include the aircraft-gate type compatibility, the aircraft-gate company compatibility, the flight attribute compatibility and the maximum allowed number of flights assigned to the apron as well as two basic constraints. Table I also reports the type of approaches (single-objective or multiobjective) used to solve GAP. The studies that convert a multiobjective GAP into a single-objective GAP by aggregating all the objectives, dominate the literature. Only a limited number of research [14], [24] has adopted multiobjective approaches to approximate the whole PF. The more recent reviews on GAP can be referred to in [4], and [15].

### B. Multiobjective Optimization Problem

In this article, the GAP is formulated as a MOP, which can be defined as follows:

\[
\begin{align*}
\text{maximize} & \quad F(x) = (f_1(x), \ldots, f_t(x)) \\
\text{subject to} & \quad x \in \Omega
\end{align*}
\]  

where \( \Omega \) is the decision space, \( F : \Omega \rightarrow \mathbb{R}^t \) is composed of \( t \) objective functions. The attainable objective set is \( \{F(x) | x \in \Omega \} \). In the case when \( \Omega \) becomes a finite set, (1) is usually called a combinatorial MOP (CMOPs).

Let \( u, v \in \mathbb{R}^t \), \( u \) is said to dominate \( v \) (i.e., \( u < v \)), if and only if \( u_i \leq v_i \) for every \( i \in \{1, \ldots, t\} \) and \( u_j < v_j \) for at least one index \( j \in \{1, \ldots, t\} \).

A solution \( x^* \in \Omega \) is Pareto-optimal to (1) if no solution \( x \in \Omega \) exists such that \( F(x) \) dominates \( F(x^*) \). \( F(x^*) \) is then called a Pareto-optimal (objective) vector. Apparently, any improvement in one objective of a Pareto-optimal solution will lead to the deterioration of at least another objective.

### C. Decomposition Approaches

An MOP can be decomposed into a number of single-objective optimization subproblems to be solved simultaneously in a collaborative way. The representative approaches, such as [8], [9], [12], have adopted this decomposition idea. One of the most commonly used decomposition methods [30] is the weighted-sum approach, which can be stated as follows.

Weighted Sum (WS): The \( i \) th subproblem is

\[
\begin{align*}
\text{minimize} & \quad g^{ws}(x|\lambda^i) = \sum_{j=1}^{t} \lambda_j f_j(x) \\
\text{subject to} & \quad x \in \Omega
\end{align*}
\]

\(^1\)In the case of maximization, the inequality signs should be reversed.
D. Pareto Local Search and Large Neighborhood Search

As the GAP is an NP-hard combinatorial optimization problem [20], [26], the heuristic methods, such as LS and meta-heuristics [11], [20], have been widely used for addressing it. On the other hand, PLS [10], [27], [32] is a natural extension of the single-objective local search, thus it can be used with the same motivation for the multiobjective GAP. PLS works by exploring the neighbors of a nondominated solution set and update the population by recently generated nondominated solutions in an iterative way.

A key issue in LS or PLS is the choice of the neighborhood structure. As a rule of thumb, the larger the neighborhood, the better quality of the local optima are. However, it should be noted that the larger neighborhood is tended to be much slower to traverse compared to its small-sized alternatives. There exist a number of studies utilizing LNS for the GAP [33]. In [21], LNS was applied for improving a layered branch-and-bound algorithm on the GAP. An adaptive LNS (ALNS) algorithm was introduced in [37] for the robust GAP.

LNS can be used for the GAP, usually requiring two operators: 1) destroy and 2) repair. The destroy operator modifies the assignments of the partial flights. Then, the repair operator reallocates flights to the new available gates. For example, Xu and Bailey [34] devised three different operators (moves) for the neighborhood search, that is, insert, exchange_1 and exchange_2. Nevertheless, those three operators are unable to obtain high-quality solutions when the number of flights for assignment in a period of time is very large [19]. To overcome this shortcoming, an interval exchange move (IEM) [19], which partially swaps two flight sequences, was introduced. An apron swap operator was also developed to switch a flight originally assigned to the apron with a flight assigned to a terminal gate. More recently, an operator for neighborhood search called exchange and greedy move (EGM) was proposed in [14] for the GAP. In the exchange move, the gates of two successive flights are exchanged. In the greedy move, the selected flights are first assigned to the apron, then these flights are reassigned to the gates with the shortest walking distance if possible.

Nevertheless, the aforementioned operators used for LNS are computationally expensive and also ignore the feasibility constraints (see Section III-C). In this article, we propose a GSS. The greedy move prefers the gates having small possibility of time conflict, out of all the available gates for each flight constructed in advance. This strategy can significantly decrease the number of times that the feasibility constraints are matched during the allocation process, which eventually reduce the time complexity.

III. FORMULATION FOR CONSTRAINED BIOBJECTIVE ROBUST GAP

In this section, the mathematical model is provided for the constrained bi-objective robust gate assignment problem.

A. Notations

All the notations used in CBR-GAM are given in Table II. Two dummy gates have been adopted in our model, based on [20]: Gate 0 represents the entrance or exit of the airport. Gate \((m + 1)\) represents the apron when the terminal gates are all occupied. The binary variable \(x_{i,k}\) denotes that gate \(k\) is assigned to flight \(i\), \(1 \leq k \leq m + 1\), and \(x_{i,k} = 0\) otherwise.

A solution \(x\) is coded in real value, as shown in Fig. 1. Each value in \(x\) represents the assigned gate number of each flight in the order from left to right based on its arrival time.

B. Objective Functions in CBR-GAM

\[
\min z_1 = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{m+1} P_{i,j} w_{k,j} x_{i,k} x_{i,j} \quad + \sum_{i=1}^{n} \sum_{k=1}^{m+1} P_{0,i} w_{0,k} x_{i,k} + \sum_{i=1}^{n} \sum_{k=1}^{m+1} P_{k,0} w_{k,0} x_{i,k}.
\]

The first objective \(z_1\) is the total walking distance of the passengers, presented in (3)

\[
\min z_2 = \sum_{k=1}^{m} \sum_{i=2}^{n} C_{0,k} c(S_{i,k}) x_{i,k}.
\]

In this article, we also propose an effective objective function in (4), in terms of the total robust cost for the gate assignment, as the second objective \(z_2\), where the robust cost \(c(S_{i,k})\) can be evaluated as follows [18]:

\[
c(S_{i,k}) = \begin{cases} 1000\left(\arctan(0.21(5 - S_{i,k})) + \frac{\pi}{2}\right), & S_{i,k} \geq \eta \\ \infty, & \text{otherwise} \end{cases}
\]

Unlike [18], the objective of the robustness formulated in this article is the weighted sum of the robust costs of all the flights, due to the fact that the importance of different flights

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(n)</td>
<td>Total number of flights</td>
</tr>
<tr>
<td>(m)</td>
<td>Total number of gates</td>
</tr>
<tr>
<td>(a_i)</td>
<td>Arrival time of flight (i)</td>
</tr>
<tr>
<td>(d_i)</td>
<td>Departure time of flight (i) ((a_i &lt; d_i))</td>
</tr>
<tr>
<td>(w_{k,t})</td>
<td>Walking distance from gate (k) to gate (t)</td>
</tr>
<tr>
<td>(p_{i,j})</td>
<td>The number of passengers transferring from flight (i) to flight (j)</td>
</tr>
<tr>
<td>(S_{i,k})</td>
<td>The latest idle time of gate (k) before the arrival of flight (i)</td>
</tr>
<tr>
<td>(c(S_{i,k}))</td>
<td>The robust cost of idle time (S_{i,k})</td>
</tr>
<tr>
<td>(C_{0,k})</td>
<td>Weight coefficient for the robust cost of flight (k)</td>
</tr>
<tr>
<td>(DC_i)</td>
<td>The delay cost of the (i)-th flight</td>
</tr>
<tr>
<td>(DC_{min})</td>
<td>The minimum delay cost of all the flights</td>
</tr>
<tr>
<td>DOC</td>
<td>Delayed operating cost</td>
</tr>
<tr>
<td>(PEL)</td>
<td>Passenger economic lost</td>
</tr>
<tr>
<td>(\eta_i)</td>
<td>Maximum passengers of flight (i)</td>
</tr>
<tr>
<td>(PLR)</td>
<td>Passenger load factor</td>
</tr>
<tr>
<td>(APrice)</td>
<td>Average ticket price</td>
</tr>
<tr>
<td>ANPM</td>
<td>Average net profit margin</td>
</tr>
<tr>
<td>AFLY</td>
<td>Average flying time</td>
</tr>
<tr>
<td>(\eta)</td>
<td>The minimum allowed safety interval in minutes</td>
</tr>
</tbody>
</table>

Fig. 1. Solution coding example.
is quite different from each other. The weight coefficient $C_0_i$ of the $i$th flight can be computed as follows:

$$C_0_i = DC_i / \min(\text{DOC})$$

where $\min(\text{DOC})$ is the minimum delay cost of the $i$th flight. The delayed operating cost $DC_i$ is the profit loss of delayed flights, and $P_0_i$ is the passenger economic loss [35]. Each of them is further specified as follows.

1) Delayed Operating Cost: The delayed operating cost includes the cost of the parking fee, take-off and landing fees as well as the passenger service charges for the aircraft, varying with the aircraft type. The delayed operating cost for all types of aircraft is given in Table III [35].

As shown in Table III, we can get the corresponding delayed operating cost with different aircraft types.

2) Profit Loss of Delayed Flights: Profit loss of delayed flights is evaluated by the hourly profit earned by each aircraft type as follows [35]:

$$P_0_i = \gamma_i \times \text{PLR} \times \text{APrice} \times \text{ANPM} / \text{AFLY}$$

where $\gamma_i$ denotes the degree of aircraft utilization; $\text{PLR}$ indicates the maximum passengers of flight $i$; $\text{APrice}$ is the average ticket price; $\text{ANPM}$ is the profit factor (less than 1); and $\text{AFLY}$ indicates the average flight time (in hours) from the departure to the destination.

3) Passenger Economic Loss: The average delay cost per passenger is 50 per hour for the domestic flights and 100 per hour for the international flights [35]. It is worth noting that the economic loss for passengers can be evaluated by computing the average maximum number of passengers for the specific type of aircraft.

C. Constraints

1) Basic Constraints:

$$\sum_{k=1}^{m+1} x_{i,k} = 1, \quad 1 \leq i \leq n$$

Equation (9) ensures that one flight is only allowed to be assigned to one and only one gate. In the condition of none available terminal gate, the flight will be assigned to the apron. Equation (10) ensures there will not be any overlapped period of time for flights assigned to the same terminal gate. Equation (11) denotes if a flight has been assigned to a gate or not.

2) Feasibility Constraints: In contrast to the previous works, beside the aforementioned basic constraints, some other feasibility constraints are also taken into account in this article, for modeling more realistic scenarios.

Airplane Type Compatibility: Suppose there are two types of aircraft: 1) large aircraft and 2) small aircraft. Generally speaking, a large gate can park large, medium and small aircraft, and small gate can only park medium and small aircraft. The type constraint should be satisfied when assigning gates to the aircraft.

Aircraft-Gate Company Compatibility: Gates are resources that owned or leased by a particular airline company for a period of time based on a medium to long term contract. Flights can only be assigned to the area of its agent.

Flight Attribute Compatibility: According to the attribute of flights, flights can be broadly divided into two categories: domestic/international flights. Taking customs or border control checks into account, international flights should be assigned to international gates while domestic flights should be allocated to the domestic gates.

$$x_{i,k} \leq o_{i,k}, \quad 1 \leq i \leq n, \quad 1 \leq k \leq m$$

where $o_{i,k}$ is a binary variable, representing if a terminal gate $k$ can be allocated to flight $i$ or not. $o_{i,k} = 1$ indicates that a terminal gate $k$ can be allocated to flight $i$ when the following compatibility constraints are satisfied. A terminal gate is allowed to be assigned to a flight only if the airline of such a flight is permitted for the corresponding gate. Moreover, the flight attribute must be matched with the terminal gate attribute (domestic/international); and the aircraft type must be supported by the terminal gate type. In addition, a constraint has been added to limit the number of flights assigned to the apron, based on (13).

$$\sum_{i=1}^{n} x_{i,m+1} \leq \text{MAXApron}$$

MAXApron is a parameter determined by the airport manager, which can be revised if required.

IV. Two-Phase Large Neighborhood Search

In this section, a 2PLNS is designed for CBR-GAM. The weighted-sum approach is utilized to speed up the convergence of the population toward PFs by the collaborative LS in Phase 1, where PLS is further conducted on the obtained solutions for extending PFs in Phase 2.

Algorithm 1 specifies 2PLNS for the CBR-GAM. A set of weight vectors $W = \{\lambda^1, \ldots, \lambda^N\}$ is first generated uniformly, where $N$ is the number of subproblems/solutions. In addition, 2PLNS maintains two populations:

1) the working population WP for PLS;
2) the external archive EA, which contains the obtained nondominated solutions.
A. Initialization
Algorithm 2 presents the initialization procedure. First, a CBR-GAM is decomposed into $N$ subproblems by the weighted sum approach with $W = \{x^1, \ldots, x^N\}$. For each solution $x^i$, the $i$-th subproblem regarding $\lambda^i$ is initialized by a heuristic.

B. Phase 1
In Algorithm 3, the population WP seeks the fast convergence toward PF by the collaborative local search. For each solution $x \in WP$, we use LNS (Algorithm 5) to generate its neighboring solutions. Then, each solution $y \in LNS(x)$ is utilized for updating $x$’s neighborhood using the weighted-sum approach, as presented in (2).

C. Phase 2
PLS in Phase 2 is presented in Algorithm 4. PLS is conducted by searching the neighborhood $LNS(x)$ for each solution $x$ in WP. Then, $LNS(x)$ is utilized for updating EA. For each newly generated solution $y \in LNS(x)$, if it is not dominated by any solutions in EA, then $y$ will be added to EA while all the solutions dominated by it will be removed from EA. After that, the newly added solutions in EA are stored in WP for the next round of PLS. If WP is empty,
EA is assigned to WP. The variable count records the total number of neighborhood search [i.e., the times of calling LNS(x)]. When it reaches a predefined value, the algorithm is terminated.

D. Large Neighborhood Search

Algorithm 5 describes the process of LNS. The neighborhood of a solution x can be generated as follows. First, γ flights are removed from x and stored as a flight set Φ. After that, for each flight h ∈ Φ, its feasible gates that satisfied the feasibility constraints (see Section III-D) are stored as a gate set G(h). For each flight h ∈ Φ, if a random number r is not larger than a predefined probability P, a greedy strategy is applied by sorting all the gates in G(h) based on the number of flights that can be assigned to them in an ascending order. In this way, the gates with the less number of flights are preferred for reducing the risk of possible conflicts between different flights. A flag variable isAssigned, which denotes whether the flight h can be assigned to a new gate, is set to False. For each gate g ∈ G(h), if h is not time conflicting with the assigned flights on gate g, then g is assigned to h and the flag isAssigned is set to true. After traversing all the gates of G(h), if the flight h is still unassigned (i.e., flag isAssigned is false), then the apron is assigned to the flight h.

An example of LNS referring to Algorithm 5 is illustrated in Fig. 2, where the top subfigure shows the original gate schedule and the bottom one displays its neighbor. Assume that the neighborhood step size Φ is 2; all the gates are feasible for Flight 5 and only Gate1 and Gate2 are feasible for Flight 6. A neighbor of a solution x is generated by LNS as follows. First, Flight 5 and Flight 6 are removed from x to form the flight set Φ (line 1). This indicates that Gate1 and Gate2 are no longer assigned to Flight 5 and Flight 6. They are to be reassigned to the new gates. We further assume that the randomly generated number r is not larger than P for Flight 5 and it is larger than P for Flight 6. This means that the feasible gates of Flight 5 are to be sorted by a greedy strategy and that of Flight 6 keeps in a random order (lines 4–7). For Flight 5, its feasible gates are tracked one by one to find a feasible gate g that has no time conflict with the existing assigned flights on g. In the example, Gate3 is assigned to Flight 5 and the flag variable isAssigned is set to true (lines 9–15). If no available gate can be found for Flight 5 (i.e., flag isAssigned is false), then the apron is assigned to Flight 5 (lines 16–18). Similarly, Gate1 is assigned to Flight 6.

It is worth noting that this large neighborhood structure has been applied in both first and second phases for LS/PLS.

E. 2PLNS Versus TPLS+PLS

TPLS+PLS [14] is also a two-phase algorithm for the multiobjective gate assignment problem. However, 2PLNS and TPLS+PLS are fundamentally different in the following aspects.

1) In Phase one, both TPLS + PLS and 2PLNS adopt the weighted-sum approach to transform a MOP into a set of single-objective subproblems. However, TPLS + PLS conduct LS on each of these subproblems at a time without any collaboration among subproblems, while 2PLNS conducts the collaborative LS among these subproblems simultaneously. This can greatly help speed up the convergence in Phase 1 due to the collaboration among subproblems. In this sense, Phase 1 of 2PLNS is similar to MOEA/D [40].

2) More importantly, a LNS is tailored for the GAP in both phases of 2PLNS, which makes it fundamentally different from TPLS + PLS. More specifically, a GSS is designed for effectively balancing between the exploitive and explorative search in 2PLNS, leading to the satisfactory performance, as presented in Section III of the supplementary material.

V. INSTANCES

A. Construction of Real Instances

A set of real instances is constructed based on the data from the CAN, China. CAN is one of the three major international hub airports in China with 68 fixed terminal gates. As shown in Table IV, each flight has seven attributes: flight ID, entry time, departure time, the district, company, type, and passengers. The flight ID is an identity number of a flight. The entry time and departure time are the arrival time and leaving time of the flight. The district means whether the flight is domestic or
Fig. 3. Terminal map of Guangzhou Baiyun international airport [1].

international. Company represents the owner of the flight. Type represents the aircraft type. Passengers indicates the number of passengers on the aircraft. It is worth noting that the number of passengers is estimated based on the aircraft type and the seat occupancy rate randomly generated within a reasonable range [3]. It can be observed from Fig. 3 that the facility has 68 fixed terminal gates with boarding bridges, 33 boarding gates in area A and 35 boarding gates in area B. The boarding gates in area A start from A101 to A133 and the boarding gates in area B start from B201 to B235. Among them, the gates from A101 to A112 can only provide service for the international flights. In addition, as presented in Table IV, each gate has four attributes: gate ID, international or domestic gate, companies that the gate provides service for and the aircraft types. The distances are measured based on navigation map provided by the airport. Due to the page limit, more details of the real instance construction can be found in Section I of the supplementary material.

B. Test Instances

Eight days’ real instances and nine synthetic instances of different scales are used. As shown in Table V, the real instances are named from Day-1 to Day-8. They contain 402, 427, 400, 409, 438, 425, and 432 flights, respectively. The number of gates in the real instances are fixed to 68 and the value of MaxApron is set to 45 except Day-8 instance. The synthetic instances of different sizes are called S120-G16, S150-G18, S180-G18, M200-G20, M220-G20, M280-G24, L350-G28, and L400-G30, respectively. More details on instances can be found in Table V.

The synthetic instances are generated in a similar way to [24]. The feasibility constraint in the CBR-GAM is considered in our instance generation procedure unlike [24]. This means that the information on the aircraft-gate type compatibility, aircraft-gate company compatibility, aircraft-gate mission compatibility and flight attribute compatibility should be present. To simplify the instance generation process, it is assumed that the constraints of flight attribute compatibility and aircraft-gate mission compatibility are satisfied for all pairs of aircraft and gates. The number of passengers on the aircraft should be determined by the type of aircraft, so its capacity. Correspondingly, there are two types of gates: large gates and small gates. For aircraft-gate compatibility, large flights can only be allocated to large gate. The medium and small flights can be parked to any type of gates. For aircraft-gate company compatibility, we randomly generate the companies which the flights and gates affiliated to. The constraint is satisfied only if the flight airline is in the supported list regarding to the airport gate.

VI. EXPERIMENTAL SETUPS

A. Parameter Settings

In the experimental studies, 2PLNS is compared with a Pareto dominance-based approach (NSGA-II [16]), four decomposition-based approaches (MOEA/D-LS (WS, TCH, PBD [40] and Pareto simulated annealing (PSA) [24]) and a state-of-the-art TPLS+PLS [14], which is a hybridization
TABLE VI
VALUES OF C-METRIC (%) BETWEEN 2PLNS AND TPLS+PLS, NSGA-II-PLS, PSA AND MOEA/D-LS (WS, TCH, PBI) OVER 20 RUNS ON 17 TEST INSTANCES

<table>
<thead>
<tr>
<th>Instance</th>
<th>TPLS+PLS</th>
<th>NSGA-II-PLS</th>
<th>PSA</th>
<th>MOEA/D-LS(WS)</th>
<th>MOEA/D-LS(TCH)</th>
<th>MOEA/D-LS(PBI)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C(A,B)</td>
<td>C(A,B)</td>
<td>C(A,B)</td>
<td>C(A,B)</td>
<td>C(A,B)</td>
<td>C(A,B)</td>
</tr>
<tr>
<td>Day-1</td>
<td>70.23</td>
<td>17.93</td>
<td>100</td>
<td>0.00</td>
<td>100</td>
<td>0.00</td>
</tr>
<tr>
<td>Day-2</td>
<td>91.10</td>
<td>4.70</td>
<td>100</td>
<td>0.00</td>
<td>100</td>
<td>0.00</td>
</tr>
<tr>
<td>Day-3</td>
<td>46.48</td>
<td>38.45</td>
<td>82.56</td>
<td>2.83</td>
<td>100</td>
<td>0.00</td>
</tr>
<tr>
<td>Day-4</td>
<td>95.19</td>
<td>2.73</td>
<td>88.19</td>
<td>1.89</td>
<td>100</td>
<td>0.00</td>
</tr>
<tr>
<td>Day-5</td>
<td>44.02</td>
<td>37.61</td>
<td>89.02</td>
<td>1.78</td>
<td>100</td>
<td>0.00</td>
</tr>
<tr>
<td>Day-6</td>
<td>68.58</td>
<td>23.66</td>
<td>95.89</td>
<td>1.17</td>
<td>81.90</td>
<td>13.49</td>
</tr>
<tr>
<td>Day-7</td>
<td>53.20</td>
<td>35.71</td>
<td>100</td>
<td>0.00</td>
<td>100</td>
<td>0.00</td>
</tr>
<tr>
<td>Day-8</td>
<td>87.54</td>
<td>8.09</td>
<td>83.21</td>
<td>4.84</td>
<td>100</td>
<td>0.00</td>
</tr>
</tbody>
</table>

A corresponds to 2PLNS.
B corresponds to the compared algorithms.

Note that $C(A, B)$ is not necessarily equal to $1 - C(A, B)$. However, $C(A, B) = 0$ means that no solution in $B$ is dominated by a solution in $A$ while $C(A, B) = 1$ indicates that all solutions in $B$ are dominated by solutions in $A$.

The higher the Hypervolume value, the better the approximation is. In the experiments, $z^r$ is set to the 1.1 times of the maximum objective values of all the nondominated solutions delivered by all the algorithms.

VII. EXPERIMENTAL ANALYSIS
The following experimental studies have been conducted:
1) comparisons of 2PLNS with TPLS+PLS [14], NSGA-II-PLS [16], PSA [24] and MOEA/D-LS (WS, TCH, PBI) [40] for CBR-GAM;
2) investigation of the effect of GSS (see Section III of the supplementary material);
3) bi-objective optimization versus single-objective optimization;
4) sensitivity test of the parameters in 2PLNS;
5) selection for the solutions of interest (SOI) from a Pareto approximated set as well as their Gantt charts (see Section IV of the supplementary material).
A. Comparison of 2PLNS With Other Algorithms

To understand the convergence behavior of seven compared algorithms, all the nondominated solutions of 2PLNS over 20 runs are compared with those of TPLS+PLS, NSGA-II-PLS, PSA, and MOEA/D-LS (WS, TCH, and PBI), in terms of c-metric, on eight real instances and nine synthetic instances with different sizes, as shown in Table VI. It is clear to see that 2PLNS significantly outperforms all the other algorithms on all the real instances and most of the synthetic instances, in terms of convergence. The final average CPU time (in seconds) used by all the algorithms over 20 runs are also presented in Table VII. Obviously, 2PLNS is the fastest among all the compared algorithms on most instances, which indicates that 2PLNS is an efficient algorithm. It is worth noting that, although PSA runs much faster than other algorithms on the synthetic instances, its performance is much worse. This can be explained by the fact that the PSA may select dominated solutions for local search, leading to less efficient search with much less solutions entering the external population that costs much less CPU time. The final nondominated solutions and boxplots in terms of hypervolume obtained by the seven compared algorithms on both synthetic and real instances can be found in Section II of the supplementary material.

Furthermore, the performance of all the compared algorithms, in terms of hypervolume, on nine synthetic instances of different sizes, is presented in Table VIII. It can be observed that the performance of 2PLNS is significantly better than that of other compared algorithms in almost all the instances. Its performance is better that of TPLS+PLS on S120-G16 and MOEA/D-LS (WS) on L400-G30 without any statistical significance.

B. Bi-Objective Optimization Versus Single-Objective Optimization

Bi-objective optimization was used to solve the GAP model. Compared with single-objective optimization, bi-objective optimization is able to obtain a PF approximation, which can be of great help for the decision maker to understand the tradeoff between two conflicting objectives and select their preferred solutions.

In addition, the single-objective optimization for two different objectives (total robust cost and walking distance) can be considered as one subproblems with the weight vector (1, 0) and one with the weight vector (0, 1) in the bi-objective optimization respectively. Experimental studies on single-objective optimization have been conducted in this section. All the experimental setups are the same to the bi-objective optimization.
Table IX shows the mean objective values obtained by the single-objective optimization, compared with that at both ends of the PF approximation [i.e., the solution of the subproblem with the weight vector (1, 0) or (0, 1)] obtained by the bi-objective optimization, over 20 runs on eleven test instances.

It can be clearly observed that the mean objective values obtained by the single-objective approach are significantly lower (better) than those obtained by the single-objective optimization on most instances, which can be explained as follows. In the decomposition-based bi-objective optimization, the optimization for the subproblem with the weight vector (1, 0) or (0, 1) is conducted in a collaborative manner with their neighboring subproblems, which can be helpful for them to get out of the local optimal.

C. Parameter Sensitivity Test

As the neighborhood step size $\Upsilon$ and the probability of applying greedy strategy $P$ are two important parameters in 2PLNS, their sensitivity test is conducted in this section. Fig. 4 shows the performance of 2PLNS in terms of hypervolume with different $\Upsilon$ on Day-1 instance. It can be observed in Fig. 4 that $\Upsilon = 10$ when 2PLNS has the best performance. In addition, Fig. 5 shows the performance of 2PLNS in terms of hypervolume with different $P$ values on instances of different sizes. It can be observed in Fig. 5 that $P = 0.8$ or 0.9 when 2PLNS has the best performance. Another interesting observation is that optimal probability of applying greedy strategy (i.e., $P$ value) in the LNS increases with the increase of the problem size.

VIII. CONCLUSION

The present study proposed a CBR-GAM. Walking distance of passengers and robust cost of the gate assignment solution were treated as the two objectives. To address CBR-GAM efficiently, a 2PLNS was devised. A LNS called GSS was developed to accelerate convergence speed and avoid local optima. Nine synthetic instances of different scales, as well as eight real instances obtained from the CAN in Guangzhou, China, were constructed. The performance of 2PLNS has been compared against five classical multiobjective optimization algorithms, MOEA/D-LS (WS,TCH,PBI), NSGA-II-PLS, PSA; and a state-of-the-art algorithm (TPLS + PLS) on both real and synthetic instances. The experimental results indicate that 2PLNS has the best overall performance in terms of both convergence and diversity. In addition, the recursive computation of the expected marginal utility (EMU) has been adopted to identify the SOI out of a large number of Pareto approximated solutions. Gantt charts based on the selected SOI verify the effectiveness of the gate scheduling.

The uncertainty of the arrival and departure time has not yet been considered in the airport gate schedule. One of our future research direction includes formulating and addressing the GAP with stochastic arrival and departure time.
REFERENCES


Wenxue Sun received the master’s degree in computer science from the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2018. He is currently working as an Engineer with NetEase, Guangzhou, China. His research interests include multiobjective optimization and data mining.

Mustafa Misir received the Ph.D. degree in computer science from Katholieke Universiteit Leuven, Leuven, Belgium, in 2012. He worked as a Postdoctoral Researcher with INRIA Saclay, Université Paris Sud—XI, Palaiseau, France, Singapore Management University, Singapore, and the University of Freiburg, Freiburg im Breisgau, Germany. He was also a Visiting Researcher with Zurich University, Zürich, Switzerland, and the Universitat Politecnica de Catalunya, Barcelona, Spain. He was a Faculty Member with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China. He is a Faculty Member of computer engineering with Istinye University, Istanbul, Turkey. He published over 40 papers in various international conferences/journals. His main research interests include automated algorithm design (machine learning + algorithm design), data science, and operations research. Dr. Misir is a recipient of several prestigious academic awards.

Kay Chen Tan (SM’08–F’14) received the B.Eng. (First Class Hons.) degree in electronics and electrical engineering and the Ph.D. degree from the University of Glasgow, Glasgow, U.K., in 1994 and 1997, respectively. He is a Full Professor with the Department of Computer Science, City University of Hong Kong, Hong Kong. He has published over 200 refereed articles and six books. Dr. Tan is the Editor-in-Chief of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, was the Editor-in-Chief of the IEEE Computational Intelligence Magazine from 2010 to 2013, and currently serves as an Editorial Board Member of over ten journals. He is an Elected Member of the IEEE CIS AdCom from 2017 to 2019.

Xiaoping Li (M’09–SM’12) received the B.Sc. and M.Sc. degrees in applied computer science from the Harbin University of Science and Technology, Harbin, China, in 1993 and 1999, respectively, and the Ph.D. degree in applied computer science from the Harbin Institute of Technology, Harbin, in 2002. He is a Full Professor with the School of Computer Science and Engineering, Southeast University, Nanjing, China. He has authored or co-authored over more than 100 academic papers, some of which have been published in international journals, such as the IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON SERVICES COMPUTING, IEEE TRANSACTIONS ON CYBERNETICS, IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING, IEEE TRANSACTIONS ON CLOUD COMPUTING, IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS: SYSTEMS, INFORMATION SCIENCES, Omega, the European Journal of Operational Research, the International Journal of Production Research, Expert Systems With Applications, and the Journal of Network and Computer Applications. His research interests include scheduling in cloud computing, scheduling in cloud manufacturing, service computing, big data, and machine learning.

Tao Xu received the B.Sc. degree in computer engineering and the M.Sc. degree in computer science and application from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 1982 and 1989, respectively. He is a Professor with the College of Computer Science and Technology, Civil Aviation University of China, Tianjin, China. His current research interests include intelligence information processing, information system analysis, and their applications.

Zhun Fan (M’04–SM’12) received the B.S. and M.S. degrees in control engineering from the Huazhong University of Science and Technology, Wuhan, China, in 1995 and 2000, respectively, and the Ph.D. degree in electrical and computer engineering from Michigan State University, East Lansing, MI, USA, in 2004. He was an Assistant Professor, and then an Associate Professor with the Technical University of Denmark, Lyngby, Denmark. He is the Head of the Department of Electronic Engineering and a Full Professor with Shantou University, Shantou, China. He has also been working with the BEACON Center for Study of Evolution in Action, Michigan State University. His major research interests include intelligent control and robotic systems, robot vision and cognition, MEMS, computational intelligence, design automation, and optimization of mechatronic systems.