Hindawi
Journal of Advanced Transportation
Volume 2020, Article ID 8835201, 13 pages
https://doi.org/10.1155/2020/8835201

Research Article

A Tabu Search-Based Algorithm for Airport Gate Assignment: A Case Study in Kunming, China

Jun Bi,[1,2] Zhen Wu,[1] Lei Wang,[3] Dongfan Xie,[1,2] and Xiaomei Zhao[1,2]

[1]*School of Traffic and Transportation, Beijing Jiaotong University, Beijing 100044, China*
[2]*Key Laboratory of Transport Industry of Big Data Application Technologies for Comprehensive Transport, Beijing Jiaotong University, Beijing 100044, China*
[3]*Information Science & Technology Department, Beijing Capital International Airport Co., Ltd., Beijing, China*

Correspondence should be addressed to Zhen Wu; 18120767@bjtu.edu.cn

Received 16 July 2020; Revised 1 November 2020; Accepted 4 November 2020; Published 27 November 2020

Academic Editor: Luigi Dell'Olio

An airport gate is the core resource of an airport operation, which is an important place for passengers to get on and off the aircraft and for maintaining aircraft. It is the prerequisite for other related dispatch. Effective and reasonable allocation of gates can reduce airport operating costs and increase passenger satisfaction. Therefore, an airport gate assignment problem (AGAP) needs to be urgently solved in the actual operation of the airport. In this paper, considering the actual operation of the airport, we formulate an integer programming model for AGAP by considering multiple constraints. The model aims to maximize the number of passengers on flights parked at the gate. A tabu search-based algorithm is designed to solve the problem. In the process of algorithm design, an effective initial solution is obtained. A unique neighborhood structure and search strategy for tabu search are designed. The algorithm can adapt to the dynamic scheduling of airports. Finally, tests are performed using actual airport data selected from Kunming Changshui International Airport in China. The experimental results indicate that the proposed method can enhance the local search ability and global search ability and get satisfactory results in a limited time. These results provide an effective support for the actual gate assignment in airport operations.

1. Introduction

The gate is an important resource for the airport and an important place for aircraft to receive ground services. The aircraft can park at the gate and the apron. For aircraft parked at the gate, passengers can board and disembark directly through the covered bridge. For airplanes parked at the apron, passengers need to get on and off the plane by shuttle bus; passengers need to use the ferry to get on and off the plane. This way is not safe and convenient for passengers. However, the number of gates is limited, so not all flights can be parked at the gate. At present, there are two main ways to solve the shortage of airport gate resources. Firstly, we can directly increase the resources of hardware facilities such as expanding airports and aprons. Because the airport's various hardware facilities are also impossible to expand indefinitely, and the expansion of the airport requires a lot of capital, time, manpower, land, etc., this method is not practical. Secondly, we can optimize the gate assignment scheme, which can improve the utilization efficiency of airport resources and reduce airport operating costs. Most airport gate assignments are made manually and by simple computer-aided methods by dispatchers based on personal experience. This makes the gate assignment work under the condition of limited gate resources costly and inefficient.

The gate assignment plan is related to the operational safety of the airport. As a place to provide passenger and cargo transportation services for aircraft, there are usually multiple aircraft and ground working vehicles on the airport parking lot for ground operations at the same time, which is prone to safety accidents. Therefore, parking as many flights as possible at the gate can reduce the use of vehicles and reduce the probability of accidents. According to the regulations of the International Air Transport Association

(IATA) [1], the proportion of passengers who board and disembark the plane over the boarding bridge should be 90%–95%. Whether passengers board or disembark through the boarding bridge is an important factor affecting passenger satisfaction. Since the number of flights that can be parked at the gate is much larger than the number of gates, how to assign the gate to maximize the number of people boarding and disembarking through the bridge is an important point in AGAP. The gate assignment is mainly a process of allocating inbound and outbound flights to various gates by the dispatch center under the conditions of flight departure and arrival times, aircraft types, and the size of parking spaces. Ensure the normal operation of flights and maintain normal order at the airport.

The paper is organized as follows. A comprehensive analysis of the existing literature is given in Section 2. In Section 3, a model of aircraft allocation considering security constraints is constructed. In Section 4, a tabu search algorithm based on the new neighborhood search structure is designed as well as a dynamic scheduling algorithm. In Section 5, the actual airport data is used to verify and analyze the algorithm. Finally, conclusions are proposed in Section 6 and future research directions are discussed.

2. Literature Review

Due to the continuous development of the aviation field, research on boarding gates has continued to increase in recent years. There are also many research methods mainly using heuristic algorithms and precise algorithms.

For the exact solution methods, Yan and Huo [2] used the travel distance of the passenger as the objective function to establish a 0-1 integer programming model and then used the branch and bound method to solve the model. Bolat [3–5] used the gate free time equilibrium as the objective function to establish the gate allocation model and used the branch and bound method to solve this problem. Mangoubi and Mathaisel [6] take the walking distance of the transit passengers as the objective function and use the linear relaxation method of the mixed integer programming and the heuristic algorithm to solve the problems. Diepen et al. [7] proposed a new integer programming model and solved this linear relaxation problem through column generation.

As the scale of flights is getting larger and larger and cannot be solved with accurate solutions, heuristic algorithms are increasingly used to solve this problem. For the heuristic algorithms, Cheng and Ho [8] use multiple metaheuristic algorithms to solve the problem of shortest distance traveled by passengers. Deng and Zhao [9] established a multiobjective aircraft allocation model and solved it using an improved PSO algorithm. Drexl and Nikulin [10] established a multitarget gate assignment model that includes the smallest number of flights without assigned gates and the smallest total walking distance (or time) of passengers, and uses Perato's simulated annealing algorithm to solve. Marinelli et al. [11] aim to minimize the total walking distance of passengers and minimize the number of flights at parking apron, and uses a metaheuristic algorithm to solve it. Mokhtarimousavi et al. [12] used the second version of Nondominated Sorting Genetic Algorithm (NSGA-II) to solve the problem with the goal of minimizing the total walking distance of passengers. Liu and Chen [13] proposed an optimization model that considers operational safety constraints and solved them using genetic algorithms. Zhao and Cheng [14] proposed a mixed integer model and designed a colony algorithm to solve this problem. Wang et al. [15] proposed an optimization model based on the characteristics of flights and airport gates and solved it using immune genetic algorithms. Benlic et al. [16] established multiple objective functions and proposed a heuristic algorithm based on the breakthrough local search (BLS) framework to solve this problem. Yan et al. [17] solved AGAP by using a random boarding gate assignment model and real-time assignment rules, as well as two punishment adjustment methods. Lim and Rodrigues [18] proposed a model with a time window and then solved it using Tabu Search and Memetic Algorithms. Ding and Lim [19] designed a tabu search algorithm to solve the problem of the shortest distance traveled by passengers with assigned gate. Noyan and Seker [20] added random interference to establish a large-scale mixed integer programming problem and used a tabu search algorithm to solve the problem.

In addition to applications in airports, optimization algorithms are also used in other fields. Zuo et al. [21] took patients' flow cost and departments' closeness as the objective function and designed a tabu search algorithm, which contains a penalty function to deal with infeasible solutions. Wang et al. [22] proposed a competitive memetic algorithm which includes the search operator in the variable neighborhood search framework and the simulated annealing strategy to solve the capacitated green vehicle routing problem. Xu et al. [23] used the Delaunay-triangulation-based Variable Neighborhood Search algorithm to solve the large-scale A colored traveling salesman problem. Wang et al. [24] took the number of unserved requests, total traveling cost, and the workload deviation as objective functions and proposed a multiobjective iterated local search algorithm with adaptive neighborhood selection to solve this problem.

For the gate allocation problem, establishing an effective gate allocation model is crucial to the actual operation of the airport. The gate assignment model proposed by some scholars and experts mainly has the following three problems. First, some constraints of the actual operation of the airport have not been fully considered; for example, flights parked at two adjacent gates cannot enter and leave at the same time. This constraint is essential for the safe operation of the airport, but most of the literature did not take this constraint into account. Secondly, the number of test cases in most literatures is relatively small or simulation data is used, which is not enough for the actual situation. Third, most papers only considered the state where all the gates are initialized to empty and did not consider the real-time changes in the status of the gates. In this study, a new neighborhood search algorithm is proposed with the objective of maximizing the number of passengers on flights parked at the gate under the consideration of security constraints. The gate assignment model was tested using a large amount of data from Kunming Changshui International Airport in China.

3. Problem Formulation

The airport gate assignment has significant impacts on the quality of airport operations. Reasonable boarding gate assignments can improve the safety and efficiency. Because of the limited number of gates, it is difficult to assign all the flights in them. Furthermore, the size of the gates also limits the types of flights that can be parked. Two aircraft on the same gate need to have a safe time interval. Due to security considerations, gates in some areas cannot enter and leave flights at the same time. As remote stands are far away from the terminal, passengers on flights parked at remote stands need to take a shuttle bus to get on and off the plane, which will seriously affect passenger satisfaction. Therefore, the goal is to maximize the number of people boarding and disembarking using the boarding bridge, and to make full use of the gate to improve the utilization of airport resources.

3.1. Objective Function. Maximize the total number of passengers on flights parked at the gate:

$$\max \sum_{i=1}^{n_f} \sum_{j=1}^{n_g} x_{ij} * g_j * p_i, \tag{1}$$

where x_{ij} indicates whether flight i is assigned to gate j; when the flight is only assigned to a gate, the value of x_{ij} is equal to 1. Otherwise, the value of x_{ij} equals 0. g_j indicates whether there is a boarding bridge at gate j; when gate has a covered bridge, the value of g_j is equal to 1. Otherwise, the value of g_j equals 0. n_f represents the total number of flights. n_g represents the total number of seats. p_i represents the number of passengers on flight i.

3.2. Constraints. Constraints are mainly generated according to some restrictions in the actual operation of the airport and then described using mathematical expressions. The purpose is to make the calculation result meet the practical limit and improve the operation efficiency. The constraints in the model are listed as follows.

(1) Each flight is only assigned to one gate:

$$\sum_{j=1}^{n_g} x_{ij} = 1, \quad \forall i \in F,\ j \in G, \tag{2}$$

where F is the set of flights and G is the set of gates.

(2) There must be a safety interval between two adjacent flights at the same gate:

$$\sum_{k=1}^{n_{fc}} x_{kj} \leq 1, \quad k \in FC, \tag{3}$$

where FC represents a collection of flights that conflict with each other in time and n_{fc} represents the set size.

(3) Flights between two adjacent gates cannot enter and leave the gate at the same time, and a safety interval is required between them:

$$\sum_{m=1}^{n_{ng}} \sum_{k=1}^{n_{fc}} \left(x_{ij} + x_{km}\right) \leq 1, \quad \forall i \in F,\ k \in FC,\ m \in NG, \tag{4}$$

where NG is the set of gates adjacent to gate j and n_{ng} is the size of the gates.

(4) The attributes of the flight (domestic/international) need to match the gate:

$$x_{ij} = 0, \quad \forall i \in F,\ \forall j \in CG_i, \tag{5}$$

where CG_i represents the set of gates that cannot match flight i in the attributes of the flight (domestic/international).

(5) The type of the gate and the type of the aircraft need to match:

$$x_{ij} = 0, \quad \forall i \in F,\ \forall j \in CG_i, \tag{6}$$

where CG_i represents the set of gates that cannot match flight i in the attributes type.

(6) 0-1 variable constraints:

$$x_{ij} \in \{0, 1\}. \tag{7}$$

4. A Tabu Search Algorithm

AGAP is an NP-hard problem [25]. As the number of flights increases, it is difficult to obtain accurate solutions using some exact solution methods. Unlike conventional solution methods, the tabu search algorithm is an intelligent search algorithm that simulates the thinking of people. That is, people will not search for the places that they have searched for, but search for other places. If they are not found, they can search for the places they have already been. The tabu search algorithm starts from an initial feasible solution. It selects a series of specific search directions based on trial and selects the movement which could improve the objective value furthest. In order to avoid falling into the local optimal solution to the tabu search, a flexible "memory" technique is used, which records the optimization process that has been performed and guides the next search. These are achieved by establishing a tabu table. The tabu list stores the movements achieved in the last few iterations. In the current iteration, the movement in the tabu list is prohibited, so that the algorithm can be prevented from revisiting the solution that has been accessed in the previous iteration and avoid falling into the local optimal solution. In order to avoid missing the optimal solution, the tabu search also has a strategy of "aspiration criterion." Compare the candidate solution of one iteration with "best so far." If the target value is better than "best so far," you can unban the candidate solution, replace "best so far" with the current optimal solution, and then add the tabu list and update tabu list.

The most important factors affecting the performance of tabu search are the quality of the initial solution, the design of the neighborhood structure, and the processing of the tabu list. Therefore, getting a good initial solution, designing a suitable neighborhood structure, and designing an effective tabu list are crucial. The structure of the solution to the airport gate assignment problem is shown in Figure 1; that is, each boarding gate will be occupied by multiple non-conflicting flights under the condition that the constraints are met, where sp_{ij} represents the time interval between adjacent flights on the same gate and nsp_{ij} represents the time interval between flights on adjacent gates.

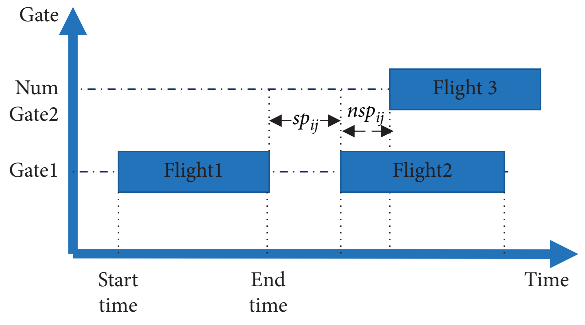

Figure 1: Structure of the solution.

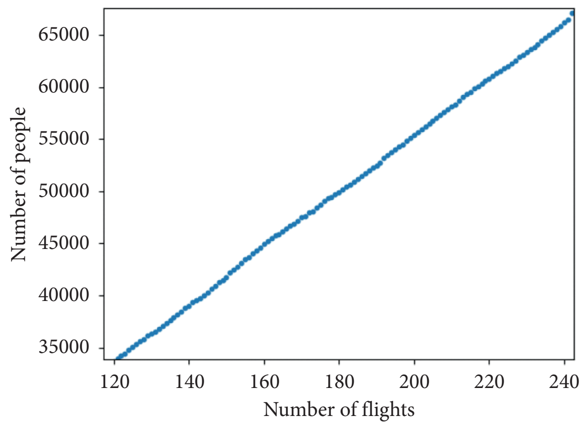

Figure 2: Correlation.

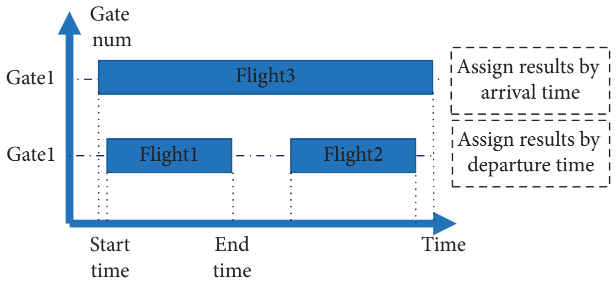
Gate num
Gate1
Gate1
Flight3
Flight1
Flight2
Assign results by arrival time
Assign results by departure time
Start time
End time
Time

Figure 3: Gate assignment in two ways.

4.1. Initial Solution. Figure 2 shows that the number of passengers on flights parked at the gate and the number of flights stopping at the gates are positively correlated. Assign gates to flights in accordance with certain rules, and make as few flights as possible to the apron, so that you can get a better initial solution. The following two rules are used to assign gates, and the literature [3] shows that, in the case of partial constraints, more flights can be parked on the gate through such rules.

(1) Sort the flights according to the corresponding departure time.

(2) For each flight, you need to retrieve each gate and then find a gate that can be put into the flight. Then, the corresponding time interval sp_{ij} is compared, where sp_{ij} is the time interval of adjacent flights of the same gate, the gate corresponding to the smallest sp_{ij} is selected, and the flight is placed on the gate.

For the first rule, manual assignment is usually based on the arrival time of the flight. If the assignment is based on the departure time of the flight, it can avoid the situation that only a few flights can be allocated when the gates are allocated according to the arrival time in Figure 3. Therefore, the number of flights docked at the apron can be reduced, and there is also a certain chance to increase the total number of passengers on flights parked at the gate. For the second rule, while ensuring a safe interval, the time interval between adjacent flights at the same gate can be made as small as possible, so that the idle time of the gate can be reduced, thereby improving the utilization rate of the gate.

4.2. New Neighborhood Search Methods. The objective function of the model is to maximize the total number of passengers on flights parked at the gate. Due to the initial solution obtained, only one-by-one allocation is performed in the order of flight departure time, and there are constraints between adjacent gates, so two problems arise. The first problem is as shown in Figure 2. Assume that the number of flight2 is greater than the number of flight1. This gate will be assigned flight1 directly on gate1 according to the time of arrival. This will cause flight2 not to be assigned to a gate, so it will as a result; the objective function is reduced. To solve this problem, this paper defines a "flight exchange" operation. Another problem is that, after the exchange of flights, there may be more idle time on the corresponding gate, and more flights may be put on this gate. This operation is defined as the "flight insert" operation.

4.2.1. Flight Exchange. The flights scheduled to be parked on the apron are screened for the gates that can be put in through various constraints, the flights that conflict with the flight at the corresponding gates are found, and the flights that conflict with them are exchanged. Define this exchange as $(f_a) \longleftrightarrow (g_i|(f_b, f_c, \ldots))$, where f_a is the flight assigned to the apron, g_i is the gate number, and $(f_b, f_c, \ldots)$ represents the set of conflicting flights at the corresponding gate. As shown in Figure 4, exchanging flight1 and flight2 can be expressed as (flight1)↔(gate1|(flight2)). Through this exchange, flights with a larger number of people can be allocated to the gate thereby obtaining a better solution.

4.2.2. Flight Insert. The result after the flight exchange may produce the result as shown in Figure 5. After flight1 and flight3 are exchanged, the free time sp_k will be generated. If there is a flight similar to flight2 on the apron, corresponding gates can be inserted under various constraints; then, the flight with the largest number of people from these flights needs to be inserted into this gate. Define this exchange as $(f_a) \longrightarrow (g_i)$. Through this exchange, you can increase the number of flights and the number of people on the gate and obtain a better solution.

4.3. Tabu Short-Term Memory. In order to prevent the loop and the local optimization from appearing in the search process, a tabu list is designed. The tabu objects in this article are mainly for the process of "flight exchange." If a neighborhood is generated by the transformation $(f_a) \longleftrightarrow (g_i|(f_b, f_c, \ldots))$, then this exchange will be added to the tabu table and a tabu length l will be added to the tabu object. This process has been forbidden until $l = 0$ and the tabu object is deleted from the tabu list, or when the "aspiration criterion" is met; the tabu object will not work.

4.4. Tabu Search Algorithm Steps

Step 1. Initialize the tabu search algorithm. Initialize the tabu list as empty, use the method in this paper to generate the initial feasible solution, and give the tabu search algorithm parameters.

Step 2. Get neighborhood solution. A neighborhood solution is generated for the current solution according to the above-mentioned method, and a part of the candidate solutions is selected from them, and the candidate solutions are sorted according to the objective function from large to small.

Step 3. Choose a better solution. Select the candidate solution corresponding to the largest objective function value, and first determine whether the amnesty criterion is met, that is, whether the objective function value is greater than the historical optimal value; if it is satisfied, then replace the historical optimal solution with the current solution, and then go to Step 2; or, go to Step 4.

Step 4. Update tabu list. Determine whether the exchange object corresponding to the candidate solution exists in the tabu list. If it does not exist, use the candidate solution as the new current solution, put the corresponding exchange object in the tabu list, and go to Step 5. Conversely, perform the $m = m - 1$ operation on the tabu length of the corresponding tabu object in the tabu list, and then reselect the suboptimal solution from the candidate set, and then perform this step again.

Step 5. Determine the end condition. Determine whether the algorithm has reached the maximum number of iterations, or the optimal solution of each iteration remains unchanged for several consecutive times; if so, the algorithm ends and the optimal solution is output; otherwise, go to Step 2.

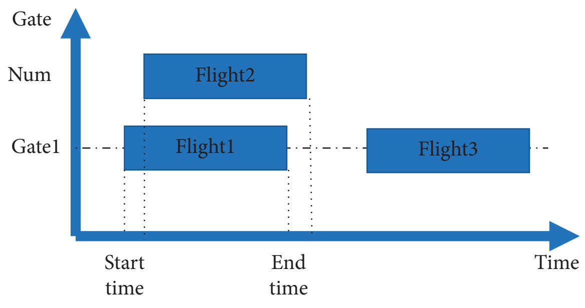

FIGURE 4: Status of flight exchange.

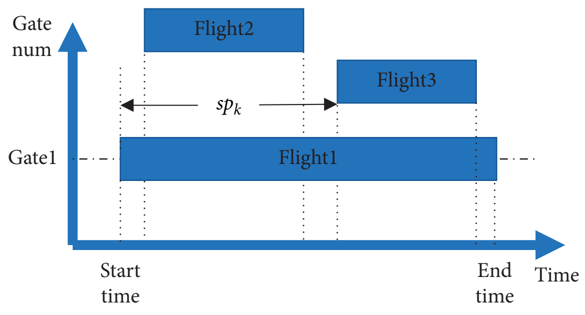

FIGURE 5: Status of flight insert.

4.5. Dynamic Allocation Algorithm. During the actual operation of the airport, the state of the gates must not be empty and will change in real time. The method mentioned in the existing literature does not apply to the actual state of all airports. Therefore, on the basis of the above tabu search algorithm, it is improved to apply to the actual operating status of the airport.

There are two statuses for the assigned flight in the actual operation of the airport. One state is that the gate assigned to the flight cannot be changed, and the other is that the gate has been assigned to the flight but the gate can be changed. In response to this situation, the improvements to the above tabu search are as follows:

(1) In the process of obtaining the initial solution, for the initial gates state, all assigned flights need to be added to the corresponding seats, and the corresponding state is added for each flight. There are three main statuses, including the inability to change the camera position, the possibility to change the camera position, and the unassigned camera position.

(2) During the flight exchange and flight insert process, it is necessary to determine the status of the flight. When the flight status cannot be changed, the exchange is stopped directly. When the flight status is exchangeable, exchange is performed when the exchange can improve the objective function. It can be exchanged when the flight status is unassigned.

5. Experimental Results and Analysis

5.1. Test Data. This paper uses actual operating data from Kunming Changshui International Airport, including 65 gates and 356 flights per day. The detailed information of the gate is shown in Table 1, where gateno indicates that the number of the gate is from 101 to 165; mdl indicates the types of aircraft that can be docked, including C, D, and E types. The three types are backward compatible. For example, E-type gates can park C and D aircraft. Nation indicates whether the type of parked flight is international or domestic. Similarly, the seat that can park international flights can also park domestic flights. Flight details are shown in Table 2, where planenumber is the unique identifier of the aircraft; arrivetime and leavetime indicate the time of flight in and out of the airport; arrivepeople and leavepeople indicate the number of flights in and out of the airport; and nation indicates the domestic and international attributes of the flight. I indicates an international flight, and D indicates a domestic flight. For example, I/D indicates that entering the airport is an international flight and leaving the airport is a domestic flight. As long as one of the flights is an international flight, the aircraft should be placed in a place where international flights can be parked. mdl indicates the type of aircraft.

5.2. Parameter Settings. The parameters of tabu search are set as follows. In order to ensure that there is no impact between flights, the time interval for the actual allocation of gates at the airport is adopted. Set the security time interval sp_{ij} between flights of the same gate to 20 minutes, and the security time interval nsp_{ij} between flights between adjacent gates to 20 minutes. In order to ensure the efficiency of time and that the algorithm can reach a better value within 200 times, the maximum number of iterations is $T_{\max} = 200$. The most important parameter is the tabu length l. If the tabu length is too short, it may fall into the local best advantage, which will cause a cycle. The length of the tabu is too long, and all the candidate solutions are tabu, which will cause the calculation time to increase. When 356 flights were selected for testing, 342 flights were assigned to gate. According to the size of the data, $l = \sqrt{m} = \sqrt{342} \approx 18$ was selected. The optimal value obtained near the tabu length of $l = 18$ is shown in Figure 6; the optimal value reaches the maximum when $l = 18$ and no longer increases with the increase of l.

5.3. Case Analysis. We selected 356 flights with arrival and departure times within one day, with a total of 98,606 people, and tested the algorithm. According to the actual allocation result of the airport, the proportion of passengers on the flight stopping at the gate on that day reached 85.63%. The initial solution obtained first includes 342 flights assigned to the gate, 14 flights assigned to the apron, and a total of 94430 people assigned to the gate. Set tabu length $l = \sqrt{m} = \sqrt{342} \approx 18$. The results obtained by the tabu search algorithm are shown in Figure 7 and Table 3. In the end, there were 96,197 people assigned to the gate at 346 flights, an increase of 1767 people compared to the initial solution. Figure 8 represents the iterative process of the algorithm. At the beginning of the algorithm, the number of people grew faster and finally reached the optimal value at the 128th time. The result made the proportion of passengers on flights parked at the gate on this day reach 97.56%. Through this case, it can be proved that the boarding gate assignment scheme obtained by the algorithm in this paper improves the total number of passengers on flights parked at the gate and reduces the airport's operating costs.

5.4. Case Analysis of Dynamic Allocation. For a day's flight, the flight that arrived yesterday and the flight that departed today and the flight that arrived and departed early need to be allocated in advance. The allocation of this flight to the new day is an initial state and needs to be considered separately. The allocation of these flights to the new day is an initial state and needs to be considered separately. Select 370 flights in one day, 41 of which are already allocated flights. The results obtained using the above dynamic allocation algorithm are shown in Figure 9. The green flight in the figure represents the assigned result. The yellow flight in the figure represents the result of algorithm allocation. In the end, 313 flights were allocated to the gate, and the result made the proportion of passengers on flights parked at the gate on this day reach 89.65%. The decrease in the bridge rate is mainly due to the initial state of the aircraft limiting the scope of understanding. Through this case, it can well represent the application of this algorithm in the actual dispatch of the airport and has a good reference for the actual dispatch of the airport.

5.5. Comparative Analysis. We implement these algorithms in Python and run them in Windows machines with i7-8700 CPU and 16G memory. They are tested with different data, and then the different test results are analyzed and researched. The algorithm is compared with the precise algorithm and genetic algorithm. The calculation time and the quality of the solution are analyzed to verify the rationality and effectiveness of the tabu search algorithm in the paper.

Test1 (363 Flights and 65 Gates). In order to test the performance of the algorithm when solving large-scale data, 65 gates of 363 flights were used for testing.

Table 1: Detailed information of gates in hub airport.

Gateno	mdl	Nation
101	C	I
102	C	I
103	D	I
104	D	I
…	…	…
163	E	D
164	E	D
165	D	D

Table 2: Detailed information of flights in hub airport.

Planenumber	Arrivetime	Leavetime	Arrivepeople	Leavepeople	Nation	mdl
B8395	2019-11-24 00 : 05 : 00	2019-11-24 01 : 10 : 00	122	143	I/D	C
B6142	2019-11-24 00 : 05 : 00	2019-11-24 14 : 30 : 00	77	111	D/I	C
B8318	2019-11-24 00 : 10 : 00	2019-11-24 09 : 10 : 00	214	208	D/D	D
…	…	…	…	…	…	…
B6371	2019-11-24 22 : 45 : 00	2019-11-24 23 : 50 : 00	120	48	D/I	C
B1461	2019-11-24 22 : 45 : 00	2019-11-24 23 : 45 : 00	114	112	D/D	C
B1565	2019-11-24 22 : 55 : 00	2019-11-24 23 : 50 : 00	131	147	D/D	C

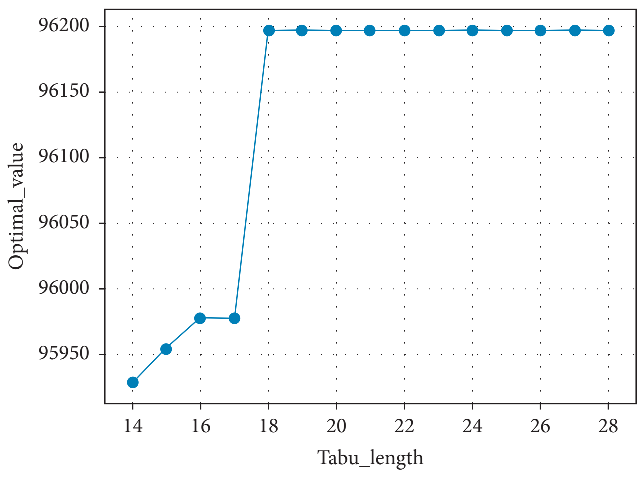

Figure 6: Tabu length selection process.

Test2 (363 Flights and 50 Gates). In order to test the efficiency and results of the algorithm when the number of gates is reduced so that more flights are allocated on the apron, 363 flights and 50 gates were used for testing.

Test3 (111 Flights and 65 Gates). In order to test the quality and efficiency of the solution for small-scale data, flights at peak hours of the day were selected for testing, which included 111 flights and 65 gates for testing.

Test4 (111 Flights and 50 Gates). In order to test the efficiency and quality of the solution of the algorithm when more flights are allocated on the apron under small-scale data, 111 flights and 50 gates were used for testing.

Compare the results of the four cases in Table 4. Since the initial solution obtained is directly obtained through certain rules, the number of flights and the number of gates will directly affect the initial solution. Comparing Test1 and Test2 and comparing Test3 and Test 4 show that when the number of gates decreases, the number of flights allocated to the gate decreases, which leads to an increase in calculation time. By analyzing the design of the domain structure in this paper, let n denote the total number of flights, and m is the number of flights assigned to the gate in the initial solution. It is necessary to compare the number of passengers on each flight parked at aprons with those on a near seat. So, the time complexity of the algorithm is $T(n) = m * (n - m)$. Based on the results obtained, this formula is also verified.

In order to further verify the effectiveness of the algorithm, the algorithm in this paper is compared with other algorithms. Analyze the differences between the algorithms and their respective advantages. The above four types of test data are solved by precise algorithms and genetic algorithms,

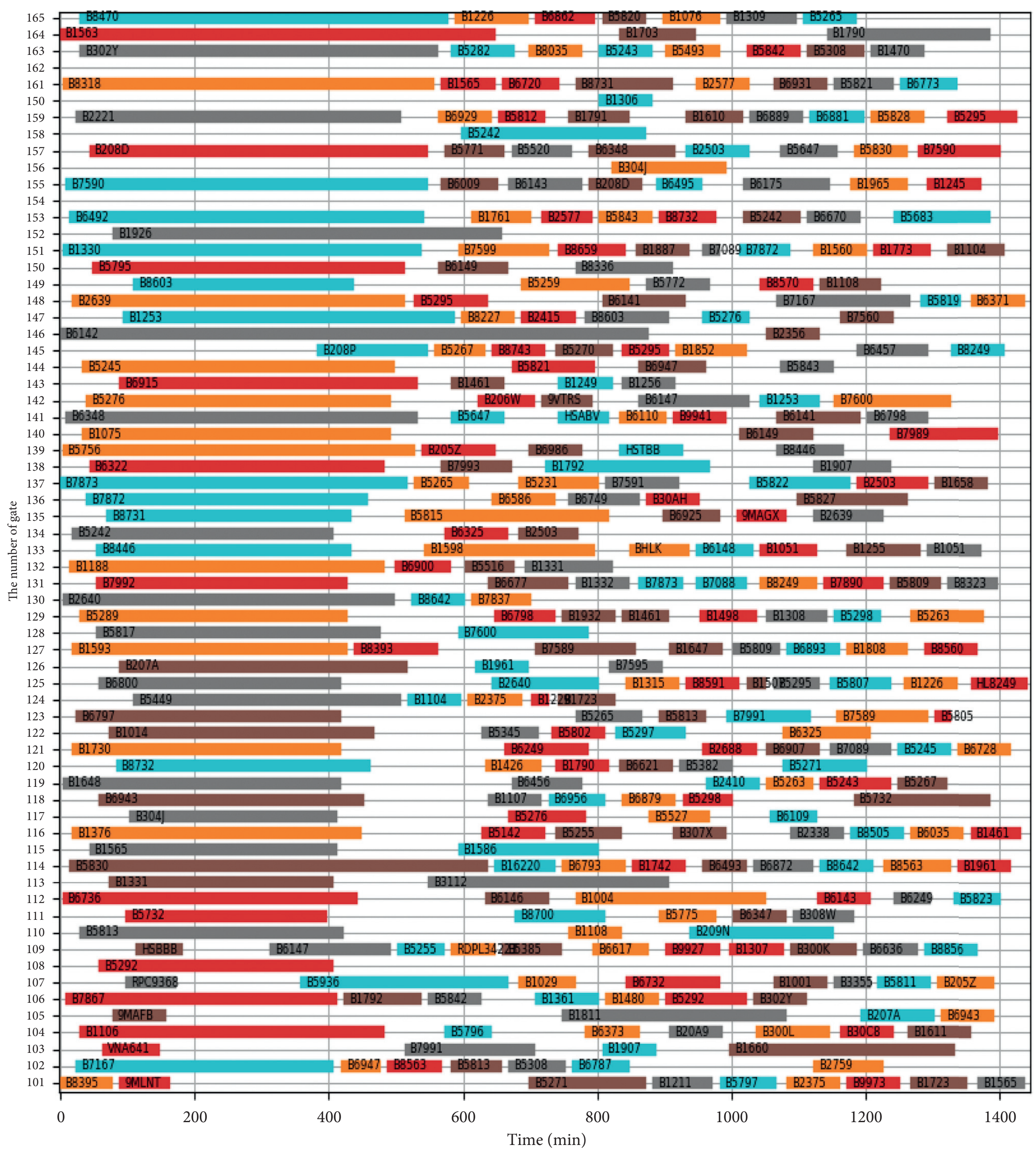
FIGURE 7: Gantt chart of airport gate assignment.

respectively. For the genetic algorithm, the maximum number of iterations is set as 200, and the results of the genetic algorithm solution are shown in Table 5. The exact algorithm uses the cutting plane method and the complex is used for the solution. The exact algorithm solution results are shown in Table 6. As the amount of data increases and

Table 3: Detailed information of gate assignment result.

Gate number	Flight number	People
101	B8395, 9MLNT, B5271, B1211, B5797, B2375, B9973, B1723, B1565	2420
102	B7167, B6947, B8563, B5813, B5308, B6787, B2759	1776
103	VNA641, B1660, B7991, B1907	1050
104	B300L, B30C8, B1611, B20A9, B5796, B6373, B1106	1971
105	9MAFB, B1811, B207A, B6943	1188
106	B7867, B1792, B5842, B1361, B1480, B5292, B302Y	2071
107	RPC9368, B5936, B1029, B6732, B1001, B3355, B5811, B205Z	2139
108	B5292	228
109	HSBBB, B6147, B5255, RDPL34223, B1307, B300K, B6636, B8856, B6617, B9927, B5385	2873
110	MU5813, B209N, B1108	866
111	B5732, B6347, B308W, B8700, B5775	1421
112	B6736, B6146, B6249, B5823, B1004, B6143	1875
...	...	...
163	B302Y, B5282, B8035, B5842, B5308, B1470, B5243, B5493	2332
164	B1563, B1703, B1790	810
165	B8470, B1226, B6862, B5820, B1309, B5265, B1076	1852

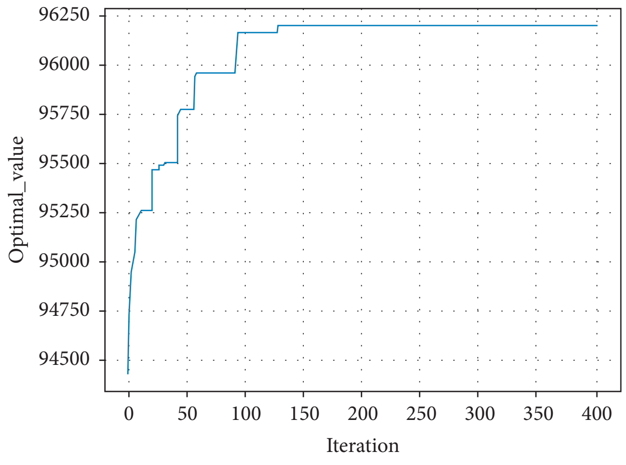

Figure 8: Iteration process.

the number of variables increases, the number of feasible solutions will increase exponentially. When solving this problem, there will be a dimensional disaster, and no feasible solution will be obtained.

According to the data in Tables 5–7 and Figures 10 and 11, when the amount of data is small, using an accurate algorithm to solve can get an accurate solution in a limited time. For example, in Test3, the number of flights parked at the gate is equal to the solution obtained by the method in the article. The number of passengers on flights parked at the gate has only increased by 21, but the calculation time required has nearly doubled. In Test4, the objective function value increased by 141, but the calculation time increased by 11 s. When the amount of data increases, it is difficult to find the optimal solution using accurate algorithms. The algorithm in this paper can get a solution with a high number of passengers on flights parked at the gate. Although it is not necessarily the optimal solution, it has certain value for practical applications. The genetic algorithm is used to solve the problem. Because the genetic algorithm is random, the results obtained are also random. For example, in Test1, the results obtained through multiple experiments are 63 less than the results of the algorithm in the paper, and the calculation time is 850 s longer. For Test2 when the gate is reduced, the solution time of the genetic algorithm is 590 s longer than the solution time of the algorithm in the paper. In short, whether it is an exact algorithm or a genetic algorithm, the algorithm in this paper has advantages for the actual gate assignment.

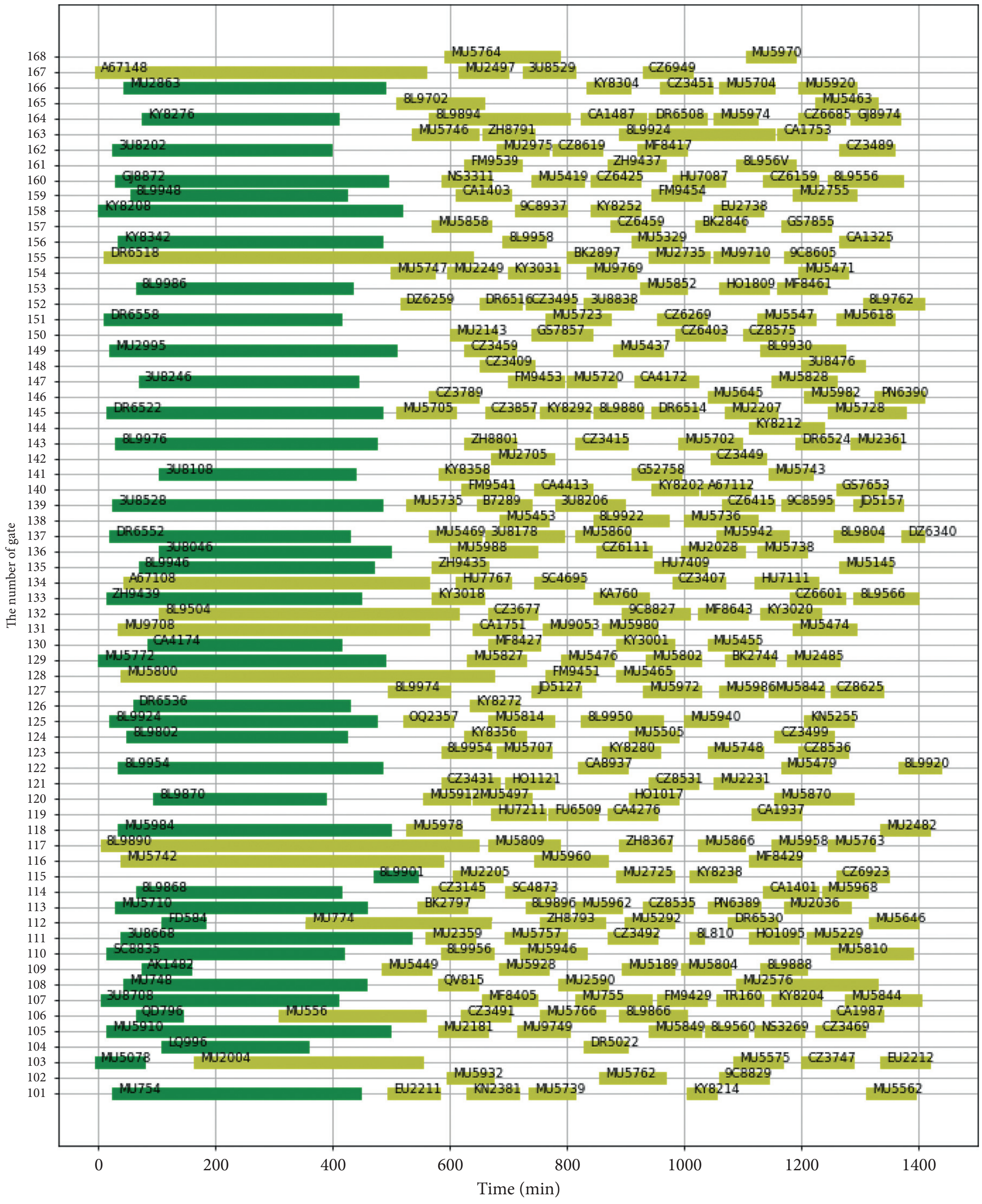

Figure 9: Gantt chart of airport gate dynamic assignment.

Table 4: Information for comparison of results.

Case	Initial solution (number of flights parked at the gate ∗ number of flights parked at the apron)	Running time (s)	The number of people added by the optimal solution
Test1	340 ∗ 23	131	1178
Test2	288 ∗ 79	360	2976
Test3	103 ∗ 8	20	1079
Test4	86 ∗ 35	41	1794

Table 5: Results of the genetic algorithm.

Case	Number of flights	Number of people	Running time (s)
Test1	342	95193	986
Test2	280	81796	950
Test3	103	24087	173
Test4	87	21368	158

Table 6: Results of the cutting plane method.

Case	Number of flights	Number of people	Running time (s)
Test1	—	—	—
Test2	—	—	—
Test3	103	24175	59
Test4	88	21576	52

Table 7: Results of individual cases.

Case	Number of people in the initial solution	Number of flights in the initial solution	Number of people in the final solution	Number of flights in the final solution
Test1	94078	340	95256	341
Test2	79465	288	82441	289
Test3	23075	101	24154	103
Test4	19641	86	21435	88

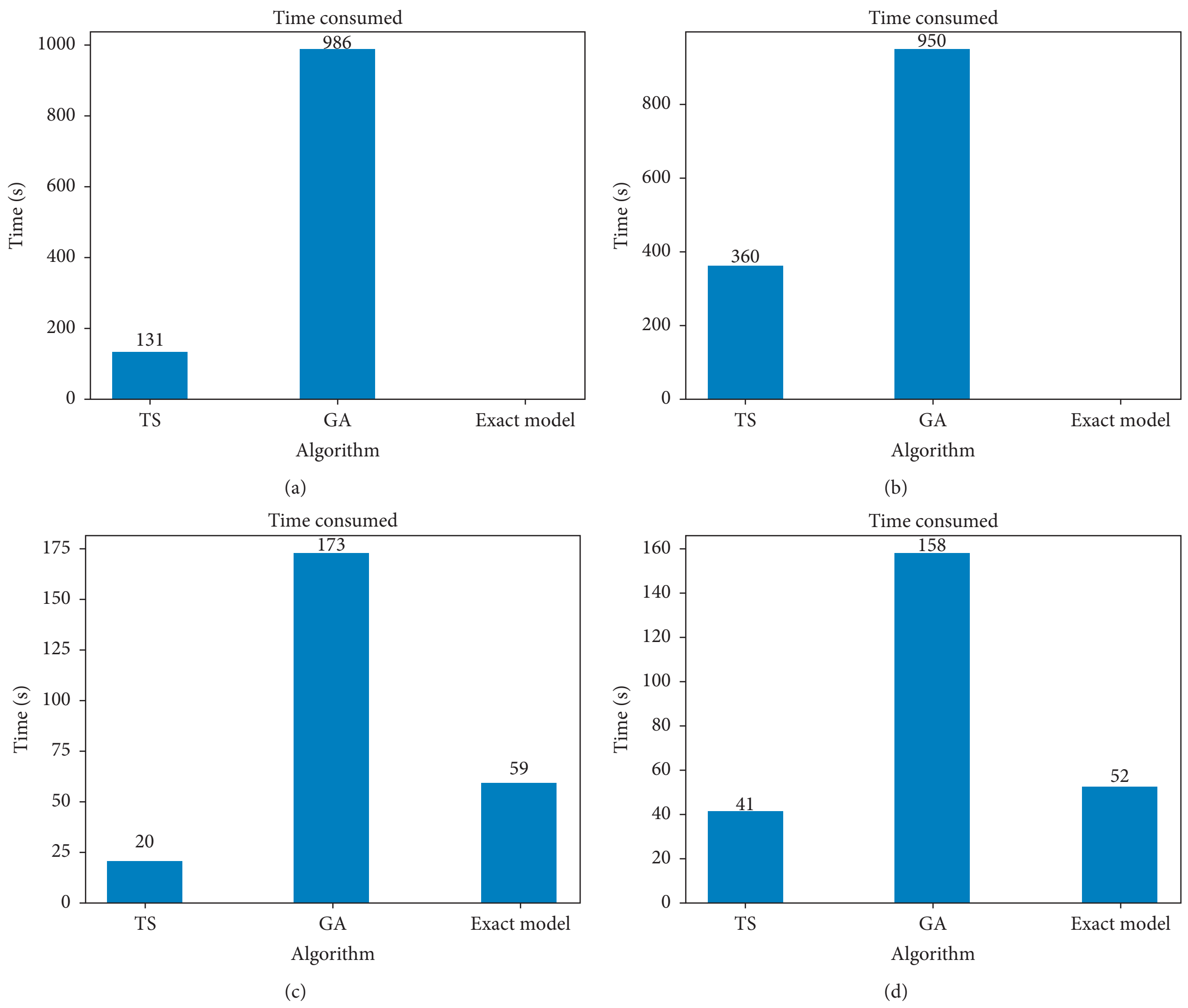

Figure 10: The comparison of time consumption. (a) Test1. (b) Test2. (c) Test3. (d) Test4.

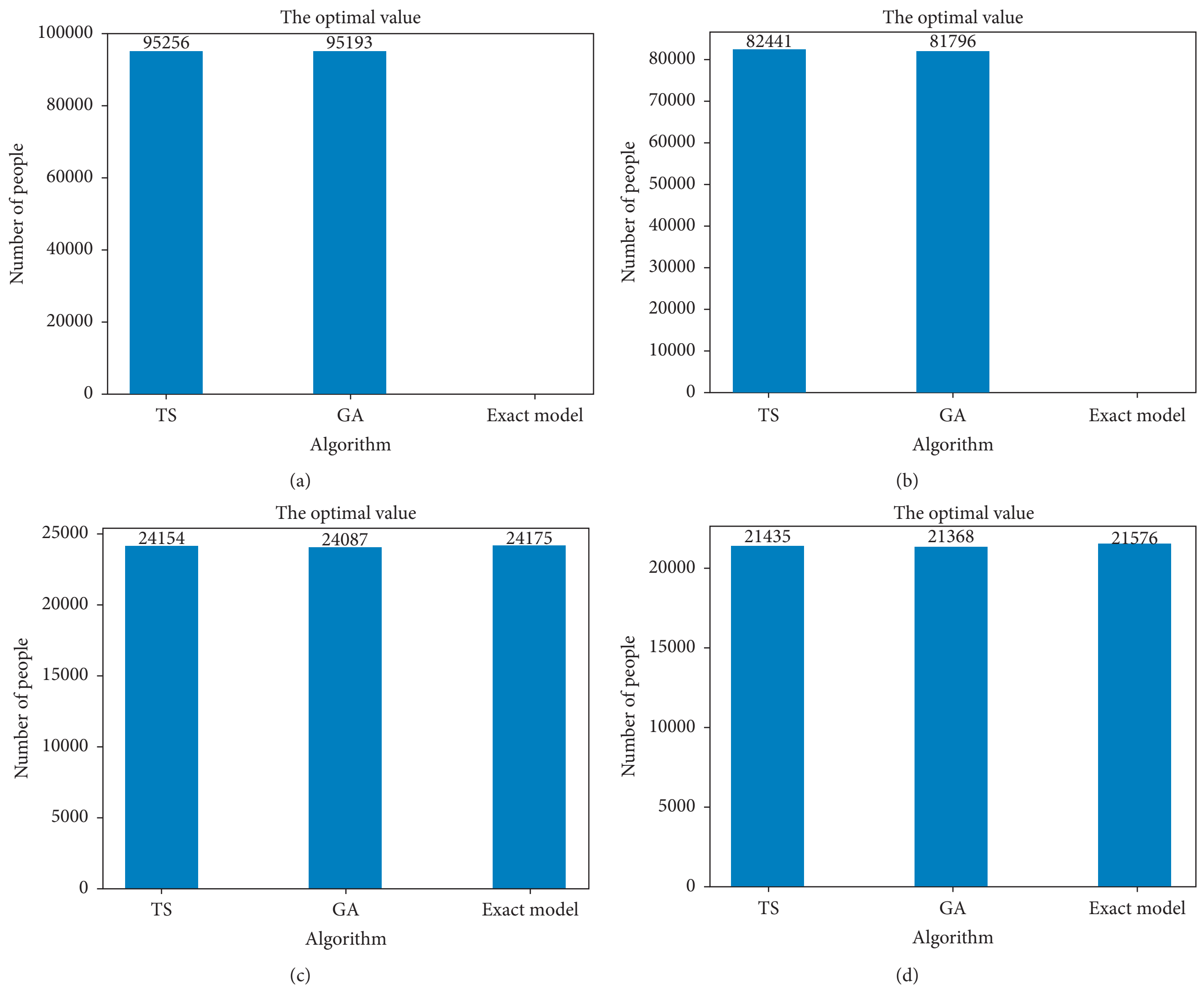

Figure 11: The comparison results of objective values. (a) Test1. (b) Test2. (c) Test3. (d) Test4.

6. Conclusion

In this paper, the total number of flights docked at the gate is used as the objective function. According to some restrictions in the actual operation of the airport, some constraints are constructed, and a mathematical model of gate assignment is established. We have designed a better initial solution algorithm and a special domain search algorithm which includes two operations: flight exchange and flight insertion. We have improved the related algorithm which has been applied to dynamic gate allocation. The tabu search algorithm designed in this way can improve the accuracy and efficiency of the solution. The practicality of the algorithm is analyzed through actual flight data and gate data of various airports, and then the genetic algorithm and the cutting plane method are used to solve the problem. The obtained solutions are compared to analyze the effectiveness of the algorithm. The experimental results show that the optimization performance of the algorithms proposed in this paper is generally better than genetic algorithms and precise algorithms and more practical than them, especially when the amount of data is relatively large. The constructed gate assignment model and algorithm can significantly increase the total number of passengers on flights parked at the gate, thereby improving airport service levels and passenger satisfaction. The proposed algorithm can provide an effective reference for airport gate allocation.

Since the flight data of the airport changes in real time, our allocation plan will also change. The algorithm proposed in this paper can only recalculate this problem, but in actual scheduling, it cannot change too many scheduling plans. It is necessary to further study the change of the flight and propose a more effective method to solve the gate reassigning problem.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported by an Open Fund of Key Laboratory of Airports Cluster Intelligent Operation under Grant no. KLAGIO20180801.

References

[1] E. Piazza, "Increasing airport efficiency: injecting new technology," *IEEE Intelligent Systems*, vol. 17, no. 3, pp. 10–13, 2002.

[2] S. Yan and C.-M. Huo, "Optimization of multiple objective gate assignments," *Transportation Research Part A: Policy and Practice*, vol. 35, no. 5, pp. 413–432, 2001.

[3] A. Bolat, "Assigning arriving flights at an airport to the available gates," *Journal of the Operational Research Society*, vol. 50, no. 1, pp. 23–34, 1999.

[4] A. Bolat, "Models and a genetic algorithm for static aircraft-gate assignment problem," *Journal of the Operational Research Society*, vol. 52, no. 10, pp. 1107–1120, 2001.

[5] A. Bolat, "Procedures for providing robust gate assignments for arriving aircrafts," *European Journal of Operational Research*, vol. 120, no. 1, pp. 63–80, 2000.

[6] R. Mangoubi and D. F. Mathaisel, "Optimizing gate assignments at airport terminals," *Transportation Science*, vol. 19, no. 2, pp. 173–188, 1985.

[7] G. Diepen, J. W. Smeltink, J. A. Hoogeveen, and J. M. v. d. Akker, "Finding a robust assignment of flights to gates at amsterdam airport schiphol," *Journal of Scheduling*, vol. 15, no. 6, pp. 703–715, 2012.

[8] C.-H. Cheng, S. C. Ho, and C.-L. Kwan, "The use of meta-heuristics for airport gate assignment," *Expert Systems with Application*, vol. 39, no. 16, pp. 12430–12437, 2012.

[9] W. Deng, H. Zhao, X. Yang, J. Xiong, M. Sun, and B. Li, "Study on an improved adaptive PSO algorithm for solving multi-objective gate assignment," *Applied Soft Computing*, vol. 59, pp. 288–302, 2017.

[10] A. Drexl and Y. Nikulin, "Multicriteria airport gate assignment and pareto simulated annealing," *IIE Transactions*, vol. 40, no. 4, pp. 385–397, 2008.

[11] M. Marinelli, M. Dell'Orco, and D. Sassanelli, "A metaheuristic approach to solve the flight gate assignment problem," *Transportation Research Procedia*, vol. 5, pp. 211–220, 2015.

[12] S. Mokhtarimousavi, D. Talebi, and H. Asgari, "A non-dominated sorting genetic algorithm approach for optimization of multi-objective airport gate assignment problem," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2672, no. 23, pp. 59–70, 2018.

[13] S. Liu, W.-H. Chen, and J. Liu, "Robust assignment of airport gates with operational safety constraints," *International Journal of Automation and Computing*, vol. 13, no. 1, pp. 31–41, 2016.

[14] H. Zhao and L. Cheng, "Ant colony algorithm and simulation for robust airport gate assignment," *Mathematical Problems in Engineering*, vol. 2014, Article ID 804310, 7 pages, 2014.

[15] L. Wang, Q.-L. Zhu, and X.-F. Xu, "Optimised assignment of airport gate configurations using an immune genetic algorithm," *Mathematical Structures in Computer Science*, vol. 24, no. 5, Article ID e240517, 2014.

[16] U. Benlic, E. K. Burke, and J. R. Woodward, "Breakout local search for the multi-objective gate allocation problem," *Computers & Operations Research*, vol. 78, pp. 80–93, 2017.

[17] S. Y. Yan and C.-H. Tang, "A heuristic approach for airport gate assignments for stochastic flight delays," *European Journal of Operational Research*, vol. 180, no. 2, pp. 547–567, 2007.

[18] A. Lim, B. Rodrigues, and Y. Zhu, "Airport gate scheduling with time windows," *Artificial Intelligence Review*, vol. 24, no. 1, pp. 5–31, 2005.

[19] H. Ding, A. Lim, B. Rodrigues, and Y. Zhu, "Aircraft and gate scheduling optimization at airports," in *Proceedings of the 37th Hawaii International Conference on System Sciences*, Big Island, HI, USA, January 2004.

[20] N. Noyan and M. Seker, "Stochastic optimization models for the airport gate assignment problem," *Transportation Research, Part E. Logistics and Transportation Review*, vol. 48, no. 2, pp. 438–459, 2012.

[21] X. Zuo, B. Li, X. Huang et al., "Optimizing hospital emergency department layout via multiobjective tabu search," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 3, pp. 1137–1147, 2019.

[22] L. Wang and J. Lu, "A memetic algorithm with competition for the capacitated green vehicle routing problem," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 2, pp. 516–526, 2019.

[23] X. Xu, J. Li, and M. Zhou, "Delaunay-triangulation-based variable neighborhood search to solve large-scale general colored traveling salesman problems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 99, pp. 1–11, 2020.

[24] J. Wang, Y. Sun, Z. Zhang, and S. Gao, "Solving multitrip pickup and delivery problem with time windows and manpower planning using multiobjective algorithms," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 4, pp. 1134–1153, 2020.

[25] A. Bouras, M. A. Ghaleb, U. S. Suryahatmaja, and A. M. Salem, "The airport gate assignment problem: a survey," *The Scientific World Journal*, vol. 2014, Article ID 923859, 27 pages, 2014.